Contents

List of Tables.................................................................................................................xiii

About this document.................................................................................................... xvii
Using this document........................................................................................................ xvii
z/OS information............................................................................................................ xvii
Discussion list............................................................................................................. xvii

How to send your comments to IBM............................................................................ xix
If you have a technical problem........................................................................................ xix

Summary of changes................................................................................................... xx
Summary of changes for z/OS UNIX for Version 2 Release 3 (V2R3)................................... xx
Summary of changes for z/OS UNIX for Version 2 Release 2 (V2R2)............................... xxii
Summary of changes for z/OS UNIX for Version 2 Release 1 (V2R1) as updated February, 2015......................... xxiii
z/OS Version 2 Release 1 summary of changes................................................................ xxiii

Chapter 1. Introduction to shell commands and DBCS.............................................. 1
Reading the command descriptions.................................................................................. 1
Using the double-byte character set (DBCS).................................................................... 7

Chapter 2. Shell command descriptions................................................................. 11
alias - Display or create a command alias...................................................................... 11
amblist - Display formatted information from object and executable files for diagnostic purposes...... 14
ar - Create or maintain library archives........................................................................ 15
as - Use the HLASM assembler to produce object files.................................................. 19
asa - Interpret ASA/FORTRAN carriage control............................................................... 23
at - Run a command at a specified time.......................................................................... 24
autolOAD - Indicate function name not defined............................................................... 27
automount - Configure the automount facility............................................................... 27
awk - Process programs written in the awk language..................................................... 34
basename - Return the nondirectory components of a path name................................... 50
batch - Run commands when the system is not busy......................................................... 51
bc - Use the arbitrary-precision arithmetic calculation language.................................... 52
bg - Move a job to the background.................................................................................. 66
bpxmtext - Display reason code text............................................................................. 67
bpxtrace - Activate or deactivate traces for processes..................................................... 68
bpxwmigf - Migrate HFS file systems to zFS................................................................. 74
break - Exit from a loop in a shell script....................................................................... 77
c++ - Compile C and C++ source code, link-edit and create an executable file................. 78
c99 - Compiler invocation using host environment variables....................................... 78
cal - Display a calendar for a month or year................................................................. 112
calendar - Display all current appointments.................................................................. 112
cancel - Cancel print queue requests (stub command).................................................... 113
captoinfo - Print the terminal entries in the terminfo database.................................... 114
cat - Concatenate or display text files......................................................................... 115
c - Compile C source code, link-edit, and create an executable file.............................. 118
cd - Change the working directory.............................................................................. 118
ceebldtx - Transform message source files into assembler source files....................... 121
chtag - Change audit flags for a file ............................................................... 124
chcp - Set or query ASCII/EBCDIC code pages for the terminal .................. 126
chgrp - Change the group owner of a file or directory .................................. 129
chlabel - Set the security label of files and directories .................................. 130
chmod - Change the mode of a file or directory ............................................. 132
chmount - Change the mount attributes of a file system ............................... 135
chown - Change the owner or group of a file or directory ............................. 137
chroot - Change the root directory for the execution of a command ................ 138
cksum - Calculate and display checksums and byte counts .......................... 143
clear - Clear the screen of all previous output ............................................. 145
cmp - Compare two files ............................................................................. 146
col - Remove reverse line feeds ................................................................... 149
: (colon) - Do nothing, successfully ............................................................. 150
comm - Show and select or reject lines common to two files ......................... 151
command - Run a simple command ............................................................... 153
compress - Lempel-Ziv file compression ....................................................... 155
conffigs - Invoke the vfs_pf I wsctl function for HFS file systems .............. 157
configstk - Configure the AF_UNIX stack ..................................................... 159
configstrm - Set and query the STREAMS physical file system configuration ... 160
continue - Skip to the next iteration of a loop in a shell script ....................... 161
copytree - Make a copy of a file hierarchy while preserving all file attributes ... 162
cp - Copy a file ............................................................................................. 163
cpio - Copy in/out file archives ...................................................................... 176
cron daemon - Run commands at specified dates and times .......................... 180
crontab - Schedule regular background jobs .................................................. 182
csplit - Split text files .................................................................................. 184
ctags - Create tag files for ex, more, and vi .................................................. 187
cu - Call up another system (stub only) ......................................................... 188
cut - Cut out selected fields from each line of a file ...................................... 189
cxx - Compile C and C++ source code, link-edit and create an executable file 192
date - Display the date and time .................................................................... 192
dbgld - Create a module map for debugging .................................................. 195
dbx - Use the debugger .................................................................................. 198
? subcommand for dbx: Search backward for a pattern .................................. 203
delete subcommand for dbx: Remove traces and stops .................................... 210
detach subcommand for dbx: Continue program execution without dbx control 210
display memory subcommand for dbx: Display the contents of memory .......... 211
down subcommand for dbx: Move the current function down the stack ........ 213
dump subcommand for dbx: Display the names and values of variables in a procedure 214
dump subcommand for dbx: Display the list of source files .......................... 215
edit subcommand for dbx: Invoke an editor .................................................... 215
goto subcommand for dbx: Run a specified source line .................................. 216
gotoi subcommand for dbx: Change the program counter address .................. 217
help subcommand for dbx: Display a subcommand synopsis .......................... 217
history subcommand for dbx: Display commands in a history list ................. 218
ignore subcommand for dbx: Stop trapping a signal ....................................... 218
list filessubcommand for dbx: Display lines of the current source file ............. 219
list files subcommand for dbx: Display the list of source files ...................... 219
list func subcommand for dbx: Display the list of functions ........................... 220
list instructions subcommand for dbx: List instructions from the program ....... 220
load characteristic subcommand for dbx: Display load characteristics .......... 221
move subcommand for dbx: Display or change the next line to be shown with the list command 222
mutex subcommand for dbx: Display a list of active mutex objects ................. 223
next subcommand for dbx: Run the program up to the next source line ......... 224
nexti subcommand for dbx: Run the program up to the next machine instruction 225
nm - Display symbol table of object, library, or executable files.................................................. 489
nohup - Start a process that is immune to hangups........................................................................ 492
obrowse - Browse a z/OS UNIX file.................................................................................................. 493
od - Dump a file in a specified format............................................................................................... 494
oedit - Edit files in a z/OS UNIX file system.................................................................................... 498
pack - Compress files by Huffman coding....................................................................................... 500
passwd - Change user passwords or password phrases................................................................. 502
paste - Merge corresponding or subsequent lines of a file............................................................ 502
patch - Change a file using diff output.............................................................................................. 506
pathchk - Check a path name............................................................................................................ 509
pax - Interchange portable archives.................................................................................................. 510
pcat - Unpack and display Huffman packed files............................................................................ 538
pg - Display files interactively........................................................................................................... 539
pr - Format a file in paginated form and send it to standard output.............................................. 543
print - Return arguments from the shell........................................................................................ 546
printf - Display the values of environment variables....................................................................... 547
ps - Return the status of a process.................................................................................................... 550
pwd - Return the working directory name........................................................................................ 559
r - Process a command history list.................................................................................................. 559
read - Read a line from standard input............................................................................................ 560
readonly - Mark a variable as read-only......................................................................................... 562
renice - Change priorities of a running process.............................................................................. 563
return - Return from a shell function or . (dot) script.................................................................. 565
rlogin - Validate rlogin requests....................................................................................................... 565
rm - Remove a directory entry.......................................................................................................... 567
rmdir - Remove a directory............................................................................................................... 567
runcat - Pipe output from mkcatdefs to gencat.............................................................................. 570
script - Makes a typescript of a terminal session............................................................................ 570
sed - Start the sed noninteractive stream editor............................................................................ 572
set - Set or unset command options and positional parameters.................................................... 579
setfacl - Set, remove, and change access control lists (ACLs)......................................................... 583
sh: Invoke a shell.................................................................................................................................. 588
shedit - Interactive command and history editing in the shell......................................................... 615
shift - Shift positional parameters..................................................................................................... 623
sleep - Suspend execution of a process for an interval of time........................................................ 624
skel - Remove old files from a directory........................................................................................... 625
sort - Start the sort-merge utility....................................................................................................... 628
spell - Detect spelling errors in files................................................................................................ 633
split - Split a file into manageable pieces.......................................................................................... 635
stop - Suspend a process or job.......................................................................................................... 637
strings - Display printable strings in binary files............................................................................ 637
strip - Remove unnecessary information from an executable file.................................................. 640
stty - Set or display terminal options............................................................................................... 641
su - Change the user ID associated with a session......................................................................... 648
submit - Submit a batch job for background processing.................................................................. 651
sum - Calculate and display checksums and block counts.............................................................. 653
suspend - Send a SIGSTOP to the current shell.............................................................................. 655
sysvar - Display static system symbols............................................................................................ 655
tabs - Set tab stops............................................................................................................................ 656
tail: Display the last part of a file..................................................................................................... 658
talk - Talk to another user............................................................................................................... 661
tar: Manipulate the tar archive files to copy or back up a file......................................................... 663
tcsh: Invoke a C shell......................................................................................................................... 671
       @ (at) built-in command for tcsh: Print the value of tcsh shell variables.................................. 714
       % (percent) built-in command for tcsh: Move jobs to the foreground or background.............. 715
       alloc built-in command for tcsh: Show the amount of dynamic memory acquired.................... 716
       bindkey built-in command for tcsh: List all bound keys.............................................................. 716
builtns built-in command for tcsh: Prints the names of all built-in commands...........................717
by built-in command for tcsh: Terminate the login shell..........................................................718
chdir built-in shell command for tcsh: Change the working directory.................................718
compl built-in command for tcsh: List completions.................................................................718
dirs built-in command for tcsh: Print the directory stack.........................................................722
echotc built-in command for tcsh: Exercise the terminal capabilities in args.............................723
filetest built-in command for tcsh: Apply the op file inquiry operator to a file..........................724
glob built-in command for tcsh: Write each word to standa output.........................................724
hashstat built-in command for tcsh: Print a statistic line on hash table effectiveness...............724
hup built-in command for tcsh: Run command so it exits on a hang-up signal...........................725
limit built-in command for tcsh: Limit consumption of processes........................................725
log built-in command for tcsh: Print the watch tcsh shell variable........................................726
login built-in command for tcsh: Terminate a login shell.......................................................726
logout built-in command for tcsh: Terminate a login shell...................................................727
ls-F built-in command for tcsh: List files....................................................................................727
notify built-in command for tcsh: Notify user of job status changes........................................728
onintr built-in command for tcsh: Control the action of the tcsh shell on interrupts...............729
popd built-in command for tcsh: Pop the directory stack..........................................................729
pushd built-in command for tcsh: Make exchanges within directory stack...............................729
rehash built-in command for tcsh: Recompute internal hash table........................................730
repeat built-in command for tcsh: Execute command count times.........................................731
sched built-in command for tcsh: Print scheduled event list.................................................731
setenv built-in command for tcsh: Set environment variable name to value.............................732
settc built-in command for tcsh: Tell tcsh shell the terminal capability cap value.................732
setty built-in command for tcsh: Control tty mode changes....................................................732
source built-in command for tcsh: Read and execute commands from name.............................733
telltc built-in command for tcsh: List terminal capability values..........................................733
uncomplete built-in command for tcsh: Remove completions whose names match pattern........733
unhash built-in command for tcsh: Disable use of internal hash table.....................................733
unlimit built-in command for tcsh: Remove resource limitations.............................................734
unsetenv built-in command for tcsh: Remove environmental variables that match pattern........734
watchlog built-in command for tcsh: Print the watch shell variable........................................734
where built-in command for tcsh: Report all instances of command.......................................735
which built-in command for tcsh: Display next executed command.......................................735

tee - Duplicate the output stream............................................................................................735
test - Test for a condition..........................................................................................................736
tic - Put terminal entries in the terminfo database.................................................................740
time - Display processor and elapsed times for a command.....................................................741
times - Get process and child process times.............................................................................742
touch - Change the file access and modification times............................................................743
tput - Change characteristics of terminals................................................................................746
tr: Translate characters.............................................................................................................747
trap - Intercept abnormal conditions and interrupts.................................................................750
true - Return a value of 0............................................................................................................752
tso - Run a TSO/E command from the shell.............................................................................752
tsocmd - Run a TSO/E command from the shell (including authorized commands)................757
tsort: Sort files topologically.....................................................................................................759
tty - Return the user's terminal name........................................................................................760
type - Tell how the shell interprets a name................................................................................760	ypeset - Assign attributes and values to variables....................................................................761
unconvdef - Create binary conversion tables............................................................................763
ulimit - Set process limits..........................................................................................................764
umask - Set or return the file mode creation mask.....................................................................766
unalias - Remove alias definitions.............................................................................................767
uname - Display the name of the current operating system.......................................................768
uncompress - Undo Lempel-Ziv compression of a file...............................................................770
unexpand - Compress spaces into tabs.......................................................................................771
uniq - Report or filter out repeated lines in a file.......................................................................773
## List of Tables

1. Locales that are supplied by z/OS UNIX....................................................................................................... 4
2. Allocation-spec keywords for allocany and allocuser................................................................................30
3. The order of operations for awk..................................................................................................................38
4. Reference documentation for programs invoked by c89, cc, and c++ commands................................. 79
5. Possible txtflag and CCSID combinations................................................................................................ 143
6. Copying files to files by using the default (without the -0 option)...........................................................168
7. Automatic conversion and file tagging behavior for cp: Copying files to files ........................................169
8. Automatic conversion and file tagging behavior for cp: Copying MVS data sets to files ...................... 169
9. Automatic conversion and file tagging behavior for cp: Copying files to MVS data sets...................... 170
10. Options that are allowed for cp: File to File and File (multiple files) to directory...............................172
11. Copies that are allowed for cp: File to File.............................................................................................172
12. Copies allowed for cp: File... (multiple files) to directory....................................................................173
13. Rules for testing files.............................................................................................................................. 309
14. Output messages of the file utility......................................................................................................... 310
15. Fields in the configuration file (inetd daemon)...................................................................................... 347
16. Explanation of the ipcs command listing............................................................................................... 354
17. Internal table sizes (lex command)........................................................................................................379
18. Output of the mount -q and -v options..................................................................................................470
19. Automatic conversion and file tagging behavior for mv: Moving files to files....................................476
20. Automatic conversion and file tagging behavior for mv: Moving files to MVS data sets....................477
21. Options allowed for mv: File to File and File ... (multiple files) to directory........................................479
22. Moves allowed for mv: File to File........................................................................................................ 479
23. Moves allowed for mv: File... (multiple files) to directory..................................................................480
**About this document**

This document presents the information you need to use a z/OS® UNIX system with the shell and utilities feature as well as TSO/E (Time Sharing Option Extensions) commands for using z/OS UNIX System Services (z/OS UNIX). Also included are REXX system commands. These features provide an application program interface (API) and a shell interface based on open systems standards.

z/OS UNIX gives the z/OS operating system an open standards interface. It consists of two features:

- Shell and Utilities, which you can use to enter shell commands, write shell scripts, and work with the file system.
- Debugger, which an application programmer can use to debug a z/OS UNIX application program written in the C or C++ languages.

This document describes how to use the shell commands, utilities, and TSO/E commands.

**Using this document**

This document is for application programmers, system programmers, and end users working on a z/OS system and using the shell. It contains information about z/OS UNIX commands.

This document also assumes that you are using Security Server for z/OS. RACF® is a component of the Security Server for z/OS. Instead of RACF, you could use an equivalent security product if it supports the system authorization facility (SAF) interfaces required by z/OS UNIX, which are documented in z/OS Security Server RACF Callable Services.

**z/OS information**

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see z/OS Information Roadmap.

To find the complete z/OS library, go to IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

**Discussion list**

Customers and IBM® participants also discuss z/OS UNIX on the mvs-oe discussion list. This list is not operated or sponsored by IBM.

To subscribe to the mvs-oe discussion, send a note to:

listserv@vm.marist.edu

Include the following line in the body of the note, substituting your given name and family name as indicated:

subscribe mvs-oe given_name family_name

After you have been subscribed, you will receive further instructions on how to use the mailing list.
How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

**Important:** If your comment regards a technical question or problem, see instead “If you have a technical problem” on page xix.

Submit your feedback by using the appropriate method for your type of comment or question:

**Feedback on z/OS function**
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

**Feedback on IBM Knowledge Center function**
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

**Feedback on the z/OS product documentation and content**
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS UNIX System Services Command Reference, SA23-2280-30
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

**If you have a technical problem**

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.
Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS UNIX for Version 2 Release 3 (V2R3)

The most recent updates are listed at the top of each section.

New

• With APAR OA56781, a new allocation specification keyword, ucat, is available to be used in conjunction with the allocany and allocuser configuration keywords for automount. This keyword restricts the allocation of a new zFS automount file system from being cataloged in the master catalog. See “automount - Configure the automount facility” on page 27.

• With APAR OA54559, the -B option was added to the iconv command. When the -B option is specified, the byte order mark (BOM) at the beginning of the Unicode data is removed during the conversion. See “iconv - Convert characters from one code set to another” on page 342.

• With APAR OA55299, the -I option was added to the cp command. When the specified file has an alias and the file is copied from UNIX to MVS, the alias information is also copied. See “cp - Copy a file” on page 163.

• With APAR OA55246, the -json and -x options were added to the zlsof command. See “zlsof - Display information about open files, sockets, and pipes” on page 877.

• A note about the METAL compiler option was added to the as command. See “as - Use the HLASM assembler to produce object files” on page 19.

• As part of HFS to zFS migration, the following commands are available:
  – “bpxwmigf - Migrate HFS file systems to zFS” on page 74 (shell command)
  – “BPXWMIGF - Migrate HFS file systems to zFS” on page 886 (TSO/E command)
  – “bpxwmigf: Migrate HFS file systems to zFS” on page 931 (system REXX command)

• The logger shell command can be used to send messages that are longer than 126 characters. Messages can also be sent to the syslogd daemon. See “logger - Log messages” on page 393.

• Warnings that the rlogind and uucpd daemons are not secure have been added to these sections:
  – “rlogind - Validate rlogin requests” on page 565
  – “uucpd daemon - Invoke uucico for TCP/IP connections from remote UUCP systems” on page 790

• Two new dbx subcommands, which can be used in dump processing mode, were added.
  – “asid subcommand for dbx: Display a list of address spaces” on page 205
  – “pid subcommand for dbx: Display a list of processes” on page 229

• Users can display a specified number of lines before or after matching lines. These commands have been enhanced to reflect the added feature:
  – “egrep - Search a file for a specified pattern” on page 283
  – “fgrep - Search a file for a specified pattern” on page 305
  – “grep - Search a file for a specified pattern” on page 333

• Support was added for the standard GNU TAR format as well as for the long pathname and long linkname entries in the GNU TAR archives. See “pax - Interchange portable archives” on page 510 and “tar: Manipulate the tar archive files to copy or back up a file” on page 663.
Changed

- For APAR OA56443, a usage note was added to the `tsocmd` shell command indicating that the command cannot be issued from the OMVS shell. See “`tsocmd - Run a TSO/E command from the shell (including authorized commands)`” on page 757.

- Clarification was added that the automount facility will not allocate a new file system for specific entries. It only allocates new file systems for generic (*) entries. See “`automount - Configure the automount facility`” on page 27.

- A usage note was added to the `tsocmd` command to clarify that the standard input stream can also be queued as input to the TSO command. See “`tsocmd - Run a TSO/E command from the shell (including authorized commands)`” on page 757.

- For APAR OA55684, a usage note was added to the following commands:
  - “`cp - Copy a file`” on page 163
  - “`mv - Rename or move a file or directory`” on page 472

- With APAR OA53128, the usage notes in “`bpwmxmigf - Migrate HFS file systems to zFS`” on page 74 were updated with additional information.

- The usage notes in “`automount - Configure the automount facility`” on page 27 was updated with clarifications about the cataloging of automount file systems.

- The `df` command was updated with a usage note about displaying a NFS-mounted file system that was mounted as version NFSv2. See “`df - Display the amount of free space in the file system`” on page 255.

- Applicable shell commands were updated to process time values beyond January 19, 2038. Usage notes were added to the following commands:
  - “`cpio - Copy in/out file archives`” on page 176
  - “`pax - Interchange portable archives`” on page 510
  - “`tar: Manipulate the tar archive files to copy or back up a file`” on page 663

- Information was added about using the 11d conversion specifier character for the long data type. See “`pax - Interchange portable archives`” on page 510. “Specifications for list mode format” on page 531 was updated with information about specifying the conversion specifier character s for ZOS.taginfo.

- Restrictions were clarified regarding `pax` support of generation data groups (GDGs) and tape data sets. See “`pax - Interchange portable archives`” on page 510.

- Also for the `pax` command, the extended header keywords atime and mtime, the stat64 structure is now used instead of stat. See “Extended header keywords” on page 525.

- Missing information about substring support for the `sysvar` command was added. See “`sysvar - Display static system symbols`” on page 655.

- “`tar: Manipulate the tar archive files to copy or back up a file`” on page 663 was updated to indicate that it supports the use of generation data groups (GDGs) and tape data sets.

- Previously, the BPXWRFD environment variable could only have a value of YES, which allowed file descriptors in the range 10-99 to be inherited. Now, with APAR OA47376, the high end of the range can be a user-specified value up to the limit of OPEN_MAX. See “`tsocmd - Run a TSO/E command from the shell (including authorized commands)`” on page 757.

- A new usage note for OEDIT indicates that all directories in the path name, starting from the root, must be searchable. See “`OEDIT - Edit a z/OS UNIX file`” on page 901.

- Appendix K, “Specifying MVS data set names in the shell environment,” on page 991 was updated with an additional restriction.

Deleted

- Because the z/OS UNIX System Services website is no longer available, information about it was deleted. Information about Tools and Toys was also deleted.
Summary of changes for z/OS UNIX for Version 2 Release 2 (V2R2)

New

• The _LD_DEBUG_TRACE environment variable can be used to export trace output to a UNIX file. See “ld - Link object files” on page 368.
• A new usage note was added to the file command. See “file - Determine file type” on page 305.
• A new file, /usr/man/%L/man[0-9]/*.stfpt, is available for the man command. It is a single-tagged preformatted plain text file that contains man pages. See “man - Display sections of the online reference manual” on page 453.

Changed

• Various updates were made to the automount command section (“automount - Configure the automount facility” on page 27).
  – Support for multi-line continuation was added. See the description of the MapName file in “Files” on page 29.
  – An example for the allocany keyword was added. See “Files” on page 29.
  – Table 2 on page 30 was updated with corrections.
  – A usage note was corrected. See “Usage notes” on page 33.
• An example was corrected in the chown command section. See “chown - Change the owner or group of a file or directory” on page 137.
• Corrections were made to the explanation of the subcommand variable of the dbx help subcommand. See “help subcommand for dbx: Display a subcommand synopsis” on page 217.
• An example was corrected in the env command section. See “env - Display or set environment variables for a process” on page 283.
• APAR OA46726 updated one of the usage notes in “extattr - Set, reset, and display extended attributes for files” on page 299.
• The description of the max value of the wait_flag field in the inetd command was corrected. See “inetd daemon - Provide service management for networks” on page 346.
• A restriction was added to the join command. See “join - Join two sorted textual relational databases” on page 362.
• The passwd description was updated to indicate that the login name associated with the current terminal is used if userid is omitted or if the -u option is omitted. See “passwd - Change user passwords or password phrases” on page 502.
• For the pax command:
  – Clarification was added to “Exit values for pax” on page 538.
  – With APAR OA46639, clarification was added that socket-type files are now treated as socket-type files. Previously, they were treated as character-special files. See “Options for the pax command” on page 512.
  – A new example was added. See “Examples for the pax command” on page 535.
• A restriction was added to the ps command. See “ps - Return the status of a process” on page 550.
• Clarification was added to explain when <backslash> is used as an escape character. See “read - Read a line from standard input” on page 560.
• APAR OA46726 clarified the supported file types in “setfacl - Set, remove, and change access control lists (ACLs)” on page 583. Also, APAR OA47215 clarified error conditions.
• An example was updated in the sort command section. See “sort - Start the sort-merge utility” on page 628.
• An example was updated in a section of the tcsh command; see Completion and listing” on page 673.
• A clarification was added for the **tso** command. See “**tso - Run a TSO/E command from the shell**” on page 752.

• APAR OA46726 clarified the file system status when returning to the unmount caller. See “**Usage notes for unmount**” on page 779. Another usage note was added about the completion of the unmount.

• APAR OA46726 clarified the file system status when returning to the unmount caller. See “**Usage notes for UNMOUNT**” on page 929.

• APAR OA47398 added clarifications were added to the BPXBATCH documentation:
  – “**General rules for setting return codes**” on page 956.
  – “**Circumstances that cause the internal and real return code to be multiplied by 256 before being issued by BPXBATCH**” on page 957.
  – “**Examples of interpreting BPXBATCH return codes in different environments**” on page 957.

• BPXCOPY was updated to clarify that it can handle any file systems. See Appendix E, “**BPXCOPY: Copying a sequential or partitioned data set or PDSE member into files,**” on page 959.

**Deleted**

• The `/var/man/%L/entry.[0-9]/*.bookname` (cached man pages that are extracted from book files) was deleted from the **man** command.

---

**Summary of changes for z/OS UNIX for Version 2 Release 1 (V2R1) as updated February, 2015**

**New**

• The **dbx** debugger now supports machine-level debugging of vector-enabled programs. A new variable, `$novregs`, was added to the set subcommand of **dbx**. See “**set subcommand for dbx: Define a value for a dbx variable**” on page 234. It was also added to “**variables: “Set” variables**” on page 1001. Information about vector registers was added to “**Usage notes for the registers subcommand of dbx**” on page 232.

The usage note for the registers subcommand of **dbx** was also updated with information about displaying and assigning vector registers. See “**Usage notes for the registers subcommand of dbx**” on page 232.

• With APAR OA46568, a usage note was added to the **ls** command about issuing the **ls** command against a large directory structure. See “**Usage notes for the ls command**” on page 404.

• With APAR OA46394, to comply with UNIX specifications, a certain arithmetic substitution was changed. See “**Arithmetic substitution**” on page 600.

**Changed**

• For the **cd** command, too many arguments could also result in exit value 2. See “**cd - Change the working directory**” on page 118.

---

**z/OS Version 2 Release 1 summary of changes**

See the Version 2 Release 1 (V2R1) versions of the following publications for all enhancements related to z/OS V2R1:

• **z/OS Migration**

• **z/OS Planning for Installation**

• **z/OS Summary of Message and Interface Changes**

• **z/OS Introduction and Release Guide**
Chapter 1. Introduction to shell commands and DBCS

This section is an introduction to the shell commands and the double-byte character set (DBCS).

Reading the command descriptions

Each shell command appears in alphabetic order. The description for each command is divided into several topics, which are explained in the following paragraphs. Some of these topics apply only to a few command descriptions. Also, some command descriptions include special topics that are not explained here.

Format

The Format topic provides a quick summary of the command's format, or syntax. The syntax was chosen to conform to general UNIX usage. For example, here is the format of the `ls` command:

```
ls [-AabCcdFfgiLlmnopqRrstuWx1] [pathname ...]
```

The format takes the form of a command line as you might type it into the system; it shows what you can type in and the order in which you should do it. The parts enclosed in square brackets are optional; you can omit them if you choose. Parts outside the square brackets must be present for the command to be correct.

The format begins with the name of the command itself. Command names always appear in bold font.

After the command name comes a list of options, if there are any. A typical z/OS shell command option consists of a dash (–) followed by a single character, typically an uppercase or lowercase letter. For example, you might have –A or –a.

**Note:** The case of letters is important. For example, in the format of `ls`, –a and –A are different options, with different effects.

If you are going to specify several options for the same command, you can put all the option characters after the same dash. Or you can put each option after its own dash. Or you can rearrange the order of options. For example, the following formats are all equivalent:

```
l s -Aa
l s -a -A
l s -aA
```

The format line shows options like -a in bold font. In the description of `ls`, all options are shown in one long string after the single dash. But another common option form is:

```
-x value
```

where –x is a dash followed by a character, and value provides extra information for using that option. For example, here is the format for the `sort` command, which takes unsorted input and sorts it:

```
s ort [-cmu]
[-o outfile]
[-t char]
[-yn] [-zn]
[-bdflMnr]
[-k startpos[,endpos]] ...
```

© Copyright IBM Corp. 1996, 2019
You can see that there are two possibilities here; you would need to choose which of the two versions of `sort` met your requirements. In either possibility, however, we have the option:

```
-o outfile
```

This option tells the `sort` command where to save its sorted output. The form of the option is `-o`, followed by a space, followed by `outfile`. In a command format, anything appearing in italic serif font is a placeholder for information that you are expected to supply. Sometimes after the format, the kind of information expected in place of the placeholder is explained. In our `sort` example, `outfile` stands for the name of a file where you want `sort` to store its output. For example, if you wanted to store the output in the file `sorted.dat`, you would specify:

```
sort -o sorted.dat
```

(followed by the rest of the command).

The format for `sort` also contains an option of the form:

```
-t char
```

This is similar to the option form we were just discussing, except that there is no space between the `-t` and `char`. `char` in italics is a placeholder; in this case, it stands for any single character. If you want to use the `-t` option for `sort`, you just type `-t` followed immediately by another character, as in:

```
sort -t:
```

In this case, we use a colon (:) in the position of the placeholder `char`.

The end of the `sort` format is:

```
[file ...]
```

This means a list of one or more file names; the ellipsis (…) stands for repetitions of whatever immediately precedes it. Since there are square brackets around the previous list, you can omit the list if you like.

The format of `ls` ended in:

```
[pathname ...]
```

As you might guess, this means that an `ls` command can end with an optional list of one or more path names. What's the difference between this and our `sort` example? A path name (specified with `pathname`) can be the name of either a file or a directory; a file name (specified with `file`) is always the name of a file.

The order of items on the command line is important. When you type a command line, you should specify its parts in the order they appear in the command format. The exceptions to this are options marked with a dash (`-`); they do not have to be given in the exact order shown in the format. However, all the `-` options must appear in the correct area of the command line. For example, you can specify:

```
ls -l -t myfiles
ls -t -l myfiles
```

but you will not get correct results if you specify:
ls myfiles -l -t ***incorrect***

or:

ls -l myfiles -t ***incorrect***

and so on. If you enter the last example, for instance, *ls* interprets –t as the path name of a file or directory, and the command will try to list the characteristics of that item.

As a special notation, most *z/OS* shell commands let you specify two dashes (--) to separate the options from the nonoption arguments; -- means that there are no more options. Thus, if you really have a directory named –t, you could specify:

ls -- -t

to list the contents of that directory.

**The Description topic**

The Description topic describes what the command does. For a particularly complex command, this topic may be divided into a large number of subtopics, each dealing with a particular aspect of the command.

The Description topic often mentions the *standard input* (stdin) and the *standard output* (stdout). The standard input is typically the workstation keyboard; the standard output is typically the display screen. The process of redirection can change this. Redirection is explained in *z/OS UNIX System Services User's Guide*.

The shell differentiates between hexadecimal, octal, and decimal format as follows:

- Any number that starts with 0x is in hexadecimal format.
- Any number that starts with 0 is in octal format.
- Any number that does not start with 0x or 0 is in decimal format.

**Options**

The Options topic describes each of the options used by the command.

**Examples**

The Examples topic is present in many command descriptions, giving examples of how the *z/OS* shell can be used.

Before you try to run any of the provided examples, you need to know that the *z/OS* shell uses the EBCDIC code page 01047 (Latin-1). Characters entered on a workstation keyboard and passed to the shell by *z/OS* do not have the same hexadecimal encoding as the code page used by the shell. You may need to customize your keyboard so that those characters have the encoding that the shell uses. See *z/OS UNIX System Services User's Guide* for more information about code page conversion.

**Environment variables**

The Environment Variables topic lists the environment variables that affect the command, if any, and describes the purposes that those variables serve. For example, the *ls* command description lists two environment variables (COLUMNS and TZ). It also explains that COLUMNS is the terminal width and that TZ contains information about the local time zone.

**Localization**

All shell commands are affected by the following special localization variables:
• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH

The Localization topic describes how the locale-related environment variables affect the behavior of the command. These environment variables allow you to access locale information, including alternate character sets; alternate numeric, monetary, and date and time formats; and foreign language translations of common messages. Locales make it easier for users around the world to use the shell and utilities.

z/OS UNIX supports the IBM-supplied locales listed in the following table. User-generated locales using IBM code page 1047 are also supported.

<table>
<thead>
<tr>
<th>Country or region</th>
<th>Language</th>
<th>Locale name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgaria</td>
<td>Bulgarian</td>
<td>Bg_BG.IBM-1025</td>
</tr>
<tr>
<td>Czech Republic</td>
<td>Czech</td>
<td>Cs_CZ.IBM-870</td>
</tr>
<tr>
<td>Denmark</td>
<td>Danish</td>
<td>Da_DK.IBM-277</td>
</tr>
<tr>
<td>Denmark</td>
<td>Danish</td>
<td>Da_DK.IBM-1047</td>
</tr>
<tr>
<td>Switzerland</td>
<td>German</td>
<td>De_CH.IBM-500</td>
</tr>
<tr>
<td>Switzerland</td>
<td>German</td>
<td>De_CH.IBM-1047</td>
</tr>
<tr>
<td>Germany</td>
<td>German</td>
<td>De_DE.IBM-273</td>
</tr>
<tr>
<td>Germany</td>
<td>German</td>
<td>De_DE.IBM-1047</td>
</tr>
<tr>
<td>Greece</td>
<td>Ellinika</td>
<td>El_GR.IBM-875</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>English</td>
<td>En_GB.IBM-285</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>English</td>
<td>En_GB.IBM-1047</td>
</tr>
<tr>
<td>Japan</td>
<td>English</td>
<td>En_JP.IBM-1027</td>
</tr>
<tr>
<td>United States</td>
<td>English</td>
<td>En_US.IBM-037</td>
</tr>
<tr>
<td>United States</td>
<td>English</td>
<td>En_US.IBM-1047</td>
</tr>
<tr>
<td>Spain</td>
<td>Spanish</td>
<td>Es_ES.IBM-284</td>
</tr>
<tr>
<td>Spain</td>
<td>Spanish</td>
<td>Es_ES.IBM-1047</td>
</tr>
<tr>
<td>Finland</td>
<td>Finnish</td>
<td>FiFI.IBM-278</td>
</tr>
<tr>
<td>Finland</td>
<td>Finnish</td>
<td>FiFI.IBM-1047</td>
</tr>
<tr>
<td>Belgium</td>
<td>French</td>
<td>Fr_BE.IBM-500</td>
</tr>
<tr>
<td>Belgium</td>
<td>French</td>
<td>Fr_BE.IBM-1047</td>
</tr>
<tr>
<td>Canada</td>
<td>French</td>
<td>Fr_CA.IBM-037</td>
</tr>
<tr>
<td>Canada</td>
<td>French</td>
<td>Fr_CA.IBM-1047</td>
</tr>
<tr>
<td>Switzerland</td>
<td>French</td>
<td>Fr_CH.IBM-500</td>
</tr>
<tr>
<td>Switzerland</td>
<td>French</td>
<td>Fr_CH.IBM-1047</td>
</tr>
<tr>
<td>France</td>
<td>French</td>
<td>Fr_FR.IBM-297</td>
</tr>
<tr>
<td>France</td>
<td>French</td>
<td>Fr_FR.IBM-1047</td>
</tr>
<tr>
<td>Croatia</td>
<td>Croatian</td>
<td>Hr_HR.IBM-870</td>
</tr>
<tr>
<td>Hungary</td>
<td>Hungarian</td>
<td>Hu_HU.IBM-870</td>
</tr>
<tr>
<td>Country or region</td>
<td>Language</td>
<td>Locale name</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Iceland</td>
<td>Icelandic</td>
<td>Is_IS.IBM-871</td>
</tr>
<tr>
<td>Iceland</td>
<td>Icelandic</td>
<td>Is_IS.IBM-1047</td>
</tr>
<tr>
<td>Italy</td>
<td>Italian</td>
<td>It_IT.IBM-280</td>
</tr>
<tr>
<td>Italy</td>
<td>Italian</td>
<td>It_IT.IBM-1047</td>
</tr>
<tr>
<td>Israel</td>
<td>Hebrew</td>
<td>Iw_IL.IBM-424</td>
</tr>
<tr>
<td>Japan</td>
<td>Japanese</td>
<td>Ja_JP.IBM-939</td>
</tr>
<tr>
<td>Japan</td>
<td>Japanese</td>
<td>Ja_JP.IBM-1027</td>
</tr>
<tr>
<td>Korea</td>
<td>Korean</td>
<td>Ko_KR.IBM-933</td>
</tr>
<tr>
<td>Belgium</td>
<td>Dutch</td>
<td>NL_BE.IBM-500</td>
</tr>
<tr>
<td>Belgium</td>
<td>Dutch</td>
<td>NL_BE.IBM-1047</td>
</tr>
<tr>
<td>Netherlands</td>
<td>Dutch</td>
<td>NL_NL.IBM-037</td>
</tr>
<tr>
<td>Netherlands</td>
<td>Dutch</td>
<td>NL_NL.IBM-1047</td>
</tr>
<tr>
<td>Norway</td>
<td>Norwegian</td>
<td>No_NO.IBM-277</td>
</tr>
<tr>
<td>Norway</td>
<td>Norwegian</td>
<td>No_NO.IBM-1047</td>
</tr>
<tr>
<td>Poland</td>
<td>Polish</td>
<td>PL_PL.IBM-870</td>
</tr>
<tr>
<td>Brazil</td>
<td>Brazilian</td>
<td>Pt_BR.IBM-037</td>
</tr>
<tr>
<td>Brazil</td>
<td>Brazilian</td>
<td>Pt_BR.IBM-1047</td>
</tr>
<tr>
<td>Portugal</td>
<td>Portuguese</td>
<td>Pt_PT.IBM-037</td>
</tr>
<tr>
<td>Portugal</td>
<td>Portuguese</td>
<td>Pt_PT.IBM-1047</td>
</tr>
<tr>
<td>Romania</td>
<td>Romanian</td>
<td>Ro_RO.IBM-870</td>
</tr>
<tr>
<td>Russia</td>
<td>Russian</td>
<td>Ru_RU.IBM-1025</td>
</tr>
<tr>
<td>Serbia</td>
<td>Serbian (Latin)</td>
<td>Sh_SP.IBM-870</td>
</tr>
<tr>
<td>Slovakia</td>
<td>Slovak</td>
<td>Sk_SK.IBM-870</td>
</tr>
<tr>
<td>Slovenia</td>
<td>Slovenian</td>
<td>SL_SI.IBM-870</td>
</tr>
<tr>
<td>Serbia</td>
<td>Serbian (Cyrillic)</td>
<td>Sr_SP.IBM-1025</td>
</tr>
<tr>
<td>Sweden</td>
<td>Swedish</td>
<td>Sv_SE.IBM-278</td>
</tr>
<tr>
<td>Sweden</td>
<td>Swedish</td>
<td>Sv_SE.IBM-1047</td>
</tr>
<tr>
<td>Turkey</td>
<td>Turkish</td>
<td>Tr_TR.IBM-1026</td>
</tr>
<tr>
<td>People's Republic of China</td>
<td>Simplified Chinese</td>
<td>Zh_CN.IBM-935</td>
</tr>
<tr>
<td>Taiwan</td>
<td>Traditional Chinese</td>
<td>Zh_TW.IBM-937</td>
</tr>
</tbody>
</table>

For more information about locales, see Appendix F, “Localization,” on page 965.

Files

The Files topic lists any supplementary files (files not specified on the command line) that the command refers to. Such files typically provide information the command needs; the command accesses these files during its operation. If the files cannot be found, the command issues a message to this effect.

Files documented in this topic might be temporary files, output files, databases, configuration files, and so on.
The z/OS XL C/C++ runtime library supports a file naming convention of // (the file name can begin with exactly two slashes). However, z/OS UNIX System Services does not support this convention. Do not use this convention (//) unless it is specifically indicated (as in the description for the c89 command). z/OS UNIX System Services does support the POSIX file naming convention, where the file name can be selected from the set of character values excluding the slash and the null character.

Usage notes

The usage notes section gives additional notes for those using the shell. Its purpose is similar to that of the Caution topic. That is, it provides important information that the reader should not overlook. However, it typically deals with issues that are more benign than what the Caution topic deals with.

Exit values

The Exit Values topic presents the error messages that the shell might display, along with a description of what caused the message and a possible action you can take to avoid getting that message. Occasionally, this topic refers you to another command description for more information about an error message.

This topic also contains information about the exit status returned by the command. You can test this status to determine the result of the operation that the command was asked to perform.

Limits

The Limits topic lists any limits on the operation of the shell. Some limits are implicit rather than explicit and might be lower than the explicitly stated limit.

Portability

The Portability topic includes two types of information:
• Availability of a version of the command on existing UNIX systems (System V, BSD)
• Compatibility with industry standards, for example, the POSIX.2 Draft Standard or the X/Open Portability Guide, Issue 4 (XPG**).

Caution

The Caution topic contains important advice for users. In z/OS shell documentation, the Caution topic is often aimed at those who are familiar with UNIX systems. Since the z/OS shell primarily conforms to the emerging POSIX standards, its behavior might not precisely match the corresponding UNIX commands. The Caution topic might point out discrepancies in behavior that might catch experienced POSIX or UNIX users by surprise.

Related information

The Related Information topic refers to other command descriptions that might contain information relevant to the command description you have just read. For example, consider the head command; by default, head displays the first 10 lines of each file given on the command line. Its Related Information topic refers you to tail, the command that displays the last 10 lines of a file.
Using the double-byte character set (DBCS)

z/OS UNIX supports the double-byte character set (DBCS). It also supports a DBCS locale. The name of the IBM-supplied DBCS locale is Ja_JP. This locale uses the IBM-939 coded-character set, which is a double-byte character set.

This topic discusses the following:

- Requirements for using DBCS
- When you must use SBCS characters and not DBCS characters
- When you can use DBCS characters
- Byte sequences that are not permitted in DBCS strings
- Displaying DBCS characters
- Switching locales
- Problems with DBCS file names containing DBCS characters

Requirements for using DBCS

If you plan to use DBCS interactively, you must work at a terminal that supports DBCS, such as a PS/55, and follow the procedures for the terminal emulator being used. It is not necessary, however, to be at a terminal that supports DBCS if you just want to use files that contain DBCS data.

To use DBCS, you need to do the following:

1. Specify special logmodes to access TSO/E and VTAM® support for DBCS. Typically, the system programmer has already set these up and provided you with instructions.

2. Issue the TSO/E PROFILE PLANGUAGE(JPN) command, if required, to receive TSO/E messages in the Japanese language.

3. On the OMVS command, use the null character conversion table (the default) for character conversion. You do not need to specify the CONVERT operand on the OCOPY, OGETX, OPUT, and OPUTX commands.

4. Access the shell using the OMVS command with the DBCS operand (which is the default setting).

You can also access the shell by using the rlogin program. The default conversion is from ISO8859-1 to IBM-1047; users can change their conversion to use different code pages by using the chcp command.

5. Define single-byte escape characters for typing escape sequences.

6. Enable the shell and utilities for the DBCS locale, including having all shell and utility messages in Japanese, by entering the these commands:

```bash
export LC_ALL=Ja_JP
exec sh
```

To receive shell and utility messages in Japanese, but not put your terminal in DBCS mode, enter the this command:

```bash
export LC_MESSAGES=Ja_JP
```

When you must use SBCS and not DBCS characters

You must use the single-byte character set (SBCS) when specifying the following:

- User names.
- System, device, group, and terminal names.
- User names, passwords, and password phrases
• Shell command-line options.
• Shell commands and their operands.
• Environment variables (DBCS characters are not exportable).
• Delimiters such as space, slash (/), braces ({}), tab, parentheses, dot (.), and any other shell special characters.
• Encoding for newline or null cannot be embedded in a DBCS character’s code. There are other rules that define valid DBCS data:
  – The DBCS blank is 0x4040.
  – The first byte of the code defining the DBCS character must be in the range 0x41 to 0xFE.
  – The second byte must be in the range 0x41 to 0xFE.
All others are not valid. This effectively covers the newline and null escape sequences, because they cannot be part of a valid DBCS character.

For more information about invalid DBCS characters, see “Byte sequences that are not permitted in DBCS strings” on page 8.

• Although file names with DBCS characters are tolerated, you should not create file names with DBCS characters. Doing so makes the file nonportable across locales, and problems may occur if file names are subsequently used in a single-byte locale. Instead, use the portable character set specified by POSIX and single-byte file names.

IBM will not support any customer problems with DBCS file names.

For more information about DBCS file names, see “Problems with file names containing DBCS characters” on page 9.

When you can use DBCS characters

When in the DBCS locale, you can use DBCS to specify the following:

• sh command-line arguments, although arguments expressed as numeric values must use SBCS characters.
• Text in data files. Files containing DBCS text are processed correctly by the shell and the utilities (such as ed and grep) if the DBCS locale is active. These files can be either DBCS text or mixed text (combinations of SBCS and DBCS). Both types of file can exist in the file system along with files that contain only single-byte text.

Byte sequences that are not permitted in DBCS strings

If you create invalid DBCS text, you may see an “illegal byte sequence” message when processing that text. The shell or command issues this error message, and the command stops processing in most cases.

Valid DBCS strings must start with "shift out" (SO [0x0E]) and end with "shift in" (SI [0x0F]). The first byte of the code defining the DBCS character must be in the range 0x41 to 0xFE. The second byte must be in the range 0x41 to 0xFE. The exception is that DBCS blank is 0x4040. All others codes are invalid.

Normal terminal operations do not produce incorrect DBCS strings. To prevent incorrect DBCS characters and strings:

• Do not use commands that operate on the data as byte strings instead of character strings. For example, head is a utility that could truncate a DBCS string or character in an inappropriate place, thus creating an incorrect DBCS string. Using pipes between utilities can also result in incorrect DBCS strings unless you pay attention to how each command handles the data.
• Do not edit text in nontext mode such as having the TSO/E editor in HEX ON mode.

If the shell command is operating on a character string and not on a byte string, and the shell is in a locale that supports DBCS, and if the utility encounters an invalid DBCS string, such as the ones described in this topic—you get an “illegal byte sequence” message and the utility may fail.
Note: newline (\n [0x15]) causes the shift state of any subsequent character sequence to start in the initial state (shifted into the SBCS mode). This may apply when a command is processing a DBCS string and encounters newline before a "shift in".


Displaying DBCS characters

In a double-byte environment, column positions are always based on the width of narrow characters. Normally, characters are “thin”; they take up only one column position when displayed. In contrast, some DBCS characters are “thick”; they take up two column positions when displayed.

The number of actual characters that are displayed by the command in the column area depends on the thickness of the characters. This applies to such commands as ls, fold, and pr, which display DBCS characters in column positions.

Switching locales

By default, the shell starts in the POSIX locale and cannot handle DBCS text until the locale is changed, typically with the shell command export LC_ALL=Ja_JP. This export command affects the current shell environment with the following exception: if you change the locale to DBCS, the shell's LC_CTYPE locale category remains in the locale until it is replaced by means of the exec command (exec /bin/sh).

Even if you change the locale to DBCS by using export LC_ALL=Ja_JP, the shell's LC_CTYPE variable remains in the previous locale (initially POSIX) until the shell is exec'd again with exec sh.

Always follow the export LC_ALL=your_locale with exec sh to be sure the shell and utilities are running in the desired locale. This is true even if you place the export LC_ALL=your_locale in your login profile.

Problems with file names containing DBCS characters

The file system treats all file names as if they contained SBCS characters. However, when you use the shell in the DBCS locale, file name and path name comparison is performed in wide mode. That is, all the characters in the name are converted to wide characters before comparison. By doing this, the shift codes are removed from the comparison and, therefore, a match can be found with the file names.

For example, if you have such DBCS file names as:

| db/so dbfile1 si  
| db/so dbfile2 si |

where so and si are the shift codes that shift out to DBCS and back to SBCS, then when in the DBCS locale (Ja_JP),

```
ls db/so file si *
```

lists both files.

When in the POSIX locale, DBCS strings are treated as byte strings. Comparison is performed byte by byte. For example:

```
ls db/so file si *
```

shows the comparison string ending with an "e si". The files in the directory would have to end with an "e si" in order to find a match. Neither of the file names in the example would be found.
Chapter 2. Shell command descriptions

Following are the descriptions of all the commands for the z/OS shell. The descriptions are listed in alphabetic order. For instructions on how to read the command descriptions, see “Reading the command descriptions” on page 1.

The z/OS shell is based on the KornShell that originated on a UNIX system. As implemented for z/OS UNIX System Services, this shell conforms to POSIX standard 1003.2-1992.

Restriction: z/OS UNIX shell commands can only read a large format sequential data set that has no more than 65,535 tracks of data on any single volume.

This information assumes that your z/OS system includes Resource Access Control Facility (RACF). Instead of RACF, your system could have an equivalent security product.

alias - Display or create a command alias

Format

```
alias [-tx] [name[=value] ...]
alias -r
```

For the tcsh shell:

```
alias [name [wordlist ]]
```

Description

When the first word of a shell command line is not a shell keyword, **alias** causes the shell to check for the word in the list of currently defined aliases. If it finds a match, the shell replaces the alias with its associated string value. The result is a new command line that might begin with a shell function name, a built-in command, an external command, or another alias.

When the shell performs alias substitution, it checks to see if *value* ends with a blank. If so, the shell also checks the next word of the command line for aliases. The shell then checks the new command line for aliases and expands them, following these same rules. This process continues until there are no aliases left on the command line, or recursion occurs in the expansion of aliases.

Calling **alias** without parameters displays all the currently defined aliases and their associated values. Values appear with appropriate quoting so that they are suitable for reinput to the shell.

Calling **alias** with parameters of the form *name=value* creates an alias for each *name* with the string *value*.

If you are defining an alias where *value* contains a backslash character, you must precede it with another backslash. The shell interprets the backslash as the escape character when it performs the expansion. If you use double quotation marks to enclose *value*, you must precede each of the two backslashes with an additional backslash. The additional backslash is needed because the shell escapes characters (that is, the shell does not interpret the character as it normally does) both when assigning the alias and again when expanding it.

To avoid using four backslashes to represent a single backslash, use single quotation marks rather than double quotation marks to enclose *value*, because the shell does not escape characters that are enclosed in single quotation marks during assignment. As a result, the shell escapes characters in single quotation marks only when expanding the alias.

Calling **alias** with *name* without any value assignment displays the function name (*name*) and its associated string value (*value*) with appropriate quoting.
**DBCS recommendation:** Use single-byte characters when specifying an alias name, because the POSIX standard states that alias names must contain only characters in the POSIX portable character set.

In the tcsh shell:

- Without arguments, `alias` in the tcsh shell prints all aliases.
- With `name`, `alias` prints the alias for name. With `name` and `wordlist`, `alias` assigns `wordlist` as the alias of `name`. `wordlist` is command and filename substituted. `name` cannot be `alias` or `unalias`.

See also the information about `unalias` in the tcsh shell in “unalias - Remove alias definitions” on page 767.

**Options**

- **-r**
  Removes all tracked aliases.

- **-t**
  Makes each `name` on the command line a tracked alias. Each tracked alias resolves to its full path name; the shell thus avoids searching the PATH directories whenever you run the command. The shell assigns the full path name of a tracked alias to the alias the first time you invoke it; the shell reassigns a path name the first time you use the alias after changing the PATH variable.

  When you enter the command:

  ```
  set -h
  ```

  each subsequent command you use in the shell automatically becomes a tracked alias. Running `alias` with the `-t` option, but without any specified names, displays all currently defined tracked aliases with appropriate quoting.

- **-x**
  Marks each alias `name` on the command line for export. If you specify `-x` without any names on the command line, `alias` displays all exported aliases. Only exported aliases are passed to a shell that runs a shell script.

Several aliases are built into the shell. Some of them are:

```
alias autoload="typeset -fu"
alias functions="typeset -f"
alias hash="alias -t"
alias history="fc -l"
alias integer="typeset -i"
alias nohup="nohup "
alias r="fc -s"
alias stop="kill -STOP"
alias suspend="stop \$\$"
```

You can change or remove any of these aliases. These changes remain in effect for the current shell and any shell scripts or child shells invoked implicitly from the command. These aliases are reset to their default built-in values each time a new shell is invoked from the command line.

**Examples**

1. The command:

   ```
   alias ls="ls -C"
   ```

   defines `ls` as an alias. From this point on, when you issue an `ls` command, it produces multicolumn output by default.

2. For the tcsh shell, to alias the `!!` `history` command, use `\!-1` instead of `\!\!`. For example:

   ```
   alias mf 'more \!-1$'
   ```
creates an alias for looking at the file named by the final argument of the previously-entered command. For example:

```bash
alias mf 'more \!-1$'
echo "We love tcsh." > file1
mf
```

```
We love tcsh. "file1" (EOF)
```

where `mf` pulls the last argument of the previous command (`file1`), and then displays that file using the `more` command.

**Localization**

**alias** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Usage notes**

1. **alias** is a built-in shell command.

2. Because exported aliases are only available in the current shell environment and to the child processes of this environment, they are not available to any new shell environments that are started (via the `exec sh` command, for example). To make an alias available to all shell environments, define it as a nonexported alias in the ENV file, which is executed whenever a new shell is run.

**Exit values**

0  
   Successful completion.

1  
   Failure because an alias could not be set.

2  
   Failure because of an incorrect command-line option.

If you define **alias** to determine the values of a set of names, the exit value is the number of those names that are not currently defined as aliases.

**Portability**

POSIX.2 User Portability Extension, UNIX KornShell.

The `-t` and `-x` options are extensions to the POSIX standard.

**Related information**

`fc`, `hash`, `nohup`, `set`, `sh`, `typeset`, `unalias`, `tcsh`
amblist - Display formatted information from object and executable files for diagnostic purposes

Format

amblist file...

Description

The amblist utility provides a UNIX interface to the AMBLIST program. With AMBLIST, you can obtain information about object modules and executable modules, and diagnose problems with them. Output is written to standard output and errors to standard error.

amblist reads control statements from standard input. One or more control statements that identify the processing to be performed must be specified. Each control statement line must begin with one or more blanks. Keywords are case-sensitive and must be uppercase. Each control statement line can be up to 80 bytes long, but only the first 70 bytes can contain control information. Control statement lines longer than 70 bytes might be ignored or might cause an error to be reported.

Options

file

The file argument can be either a path name or a data set name. You cannot specify both a path name and a data set name at the same time. If you do, amblist ends with an error message and a nonzero return code.

- If a path name is specified, it can be either a UNIX file or a UNIX directory. You can use only one path name at a time. If a UNIX directory is used for the path name, MEMBER must be specified on the amblist control statement to specify the file name.
- If a data set name is specified, or more data sets can be listed to indicate a concatenation of data sets to be searched. If a member name cannot be specified on the data set name (such as for LISTLOAD), it can either be specified on a control statement or omitted completely. If the member name is omitted, then all members are processed.

Examples

1. Control statement from a pipe, output redirected to a file:

   ```
   echo " LISTLOAD" | amblist a.out > a.amblist
   ```

2. Control statement read interactively, output sent to terminal:

   ```
   amblist hello.o
   ```
   The user must then type " LISTOBJ" (leading blank, no quotation marks), then press CTRL-D to end the amblist processing.

3. Control statement from a file, with output sent to terminal:

   ```
   amblist hello.o < hello.ambctl
   ```
   where the file hello.ambctl contains a single line " LISTOBJ" (leading blank, no quotation marks).

4. Control statement from pipe, process an object data set, output redirected to a file: contains the single line " LISTOBJ" (leading blank, no quotation marks).

   ```
   echo " LISTOBJ" | amblist "/binder.obj(hello)" > hello.amblist
   ```

For examples of output that is created when running amblist, see z/OS MVS Diagnosis: Tools and Service Aids.
Localization

`amblist` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values

0
- Successful completion.

1
- No operands were specified, or -? was used as an option

2
- An incorrect option was specified.

3
- The AMBLIST program ended with a nonzero return code. The return code is in message AMBU2003. Some output might have been written to standard output.

4
- UNIX path names and MVS™ data set names cannot be mixed.

9
- A file could not be opened. The error number is in message AMBU2010.

10
- Dynamic allocation failure. Message AMBU2010 indicates the return code, error code, and the information code from dynamic allocation (SVC 99). Additional messages that describe the error might have been written to standard error.

Usage notes

1. You cannot use the DNN= control statement keyword except when you specify the default value of SYSLIB because `amblist` does not provide a facility for the use of any other data definition name.

2. You must specify at least one library when you invoke `amblist`. Do not use LISTLPA because it is obsolete.

Related information

`z/OS MVS Diagnosis: Tools and Service Aids` contains detailed reference information about the AMBLIST program. `z/OS MVS Program Management: User’s Guide and Reference` also has information about AMBLIST.

ar - Create or maintain library archives

Format

```
ar -d[-Ilv] archive member ... ...
ar -m[-abIlsv] [posname] archive member ...
ar -p[-Ilsv] archive member ...
ar -q[-cls]v [-F format] archive member ...
ar -r[abcIlsv] [-F format] [posname] archive member ...
```

Shell command descriptions 15
ar -t[Ilsv] archive member ...
ar -u[-abcIilsv] [-Fformat] [posname] archive member ...
ar -x[-CIlsTv] archive member...]

Description

ar maintains archive libraries. The archive library is a collection of files, typically object files. Using ar, you can create a new library, add members to an existing library, delete members from a library, extract members from a library, and print a table of contents for a library.

A library member is an arbitrary file. Typically, these files are object files or side files, suitable for use by a linkage editor.

If any members of a library are object files, ar creates and maintains an external symbol index for link-editing.

Member names in an archive are only the final component of any path name. When creating a new library member (member) as given on the command line, ar uses the full path name given. When storing the member name in the library, or comparing a member name, ar uses only the final component.

Options

The format shows the main functions of ar, which are defined as follows:

-d
Deletes each named member from the archive and regenerates the symbol table.

-m
Moves the named archive member in the archive. The new position is specified by -a, -b, i, or posname. If a location is not specified, the member is moved to the end of the archive.

-p
Displays each member specified to the standard output. If you did not specify any members, ar displays all members.

-q
Quickly appends the specified file to the archive. With this option, ar does not check to see if file is already a member of the archive.

-r
Replaces or adds file to archive. If archive does not exist, ar creates it and prints a message. When ar replaces an existing member, the archive order is not changed. If file is not replacing a member, it is added to the end of the archive unless -a, -b, or -i is used. This option regenerates the symbol table.

-t
Displays a table of contents that lists members, or every member if member is not specified. ar prints a message for each member it does not find. By default, ar prints the member name for all selected members. With the verbose (-v) option, ar prints more information for all selected members.

-x
Extracts each specified member from the archive and copies it to a file. If member is specified as a full path name, it is copied to that path name. If no member is specified, all members are extracted. The archive remains unchanged.

The following options change the behavior of the main functions:

-a
Placed file in the archive after the member specified by posname. If no member is named, file is added to the end of the archive.

-b
Placed file in the archive before the member specified by posname. If no member is named, file is placed at the beginning of the archive.
-C Prevents ar from overwriting existing files with extracted files. This option is used only with extraction (-x).

-C Suppresses the message that is normally printed when ar creates a new archive file. You can use this only in conjunction with the -r and -q options.

-F format Specifies the archive format to be used by a new archive. You can use this option only when creating a new archive with the -r and -q options.

-I Ignores the case of letters when searching the archive for specified member names. Normally, the case is significant.

-i Inserts file into the archive before the member specified by posname. If posname isn't specified, ar inserts file at the beginning of the archive. This option is the same as -b.

-l This option is ignored. It requests that temporary files that are generated by ar be put in the directory rather than in the default temporary file directory. It is provided for compatibility with earlier versions of ar

-s Regenerates the external symbol table regardless of whether the command modifies the archive.

-t When used with -x, allows extraction of members with names longer than the file system supports. Normally this is an error and ar does not extract the file. Most file systems truncate the file name to the appropriate length.

-u Replaces the archive member only if the member file's modification time is more recent than the archive member time. -u implies -r, so it is not necessary to specify -r also.

-v Gives verbose output. With -d, -q, -r, and -x, this option prints the command letter and the member name affected before performing each operation. With -t, ar prints more information about archive members using a format similar to ls -l. With -p, ar writes the name of the member to standard output, before displaying the contents of the file.

Operands

archive Specifies the path name of the archive file.

member Specifies the path name of the file that is to be acted upon (placed, deleted, searched for, and so on) in the archive library.

Examples

1. To add a member fioacc.o to the archive file /u/turner/bin/cliserpgm.a, specify:

   ```
ar -rc /u/turner/bin/cliserpgm.a fioacc.o
   ```

2. To display the members of the archive file /u/turner/bin/cliserpgm.a, specify:

   ```
ar -tv /u/turner/bin/cliserpgm.a
   ```

3. To delete the member repgen.o from the archive file /u/turner/bin/cliserpgm.a and regenerate the external symbol table for the archive, specify:

   ```
ar -ds /u/turner/bin/cliserpgm.a repgen.o
   ```
Environment variables

ar uses the following environment variable:

TMPDIR

The path name of the directory that is used for temporary files. If it is not set, z/OS UNIX uses /tmp.

Localization

ar uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_TIME
- NLSPATH


Files

ar creates temporary files in the directory of the archive file and in the directory named by the TMPDIR environment variable. These files are intermediate versions of the archive file being created or updated. Consequently, they require approximately the same file size as the archive file being manipulated.

Usage notes

ar can be used to store multiple versions of the same object file within one archive library. This is useful if you are providing an archive library which might be used to resolve references from code that is compiled with various compiler options. These options cause differences in the object files which must be matched with the archive library member attributes. Attributes for ar are: AMODE, XPLINK, and IPA.

ar will store the attribute information for every entry in the symbol table. The linkage editor will use the attribute information to resolve external references with the appropriate archive library member. Because archive library member names are only the final component of the path name, these member names must be unique for the different object file versions.

Side files (normally those created when link-editing a DLL) can be made members of an archive file. When the linkage editor processes such an archive file, it will normally read in all such side-files so that archives can be used for resolving symbol references in DLLs. For more information about resolving external references, see z/OS MVS Program Management: User's Guide and Reference.

You will want to establish a naming convention for the object files, and change your build procedures to generate the correct names. For example, if your archive contains three versions of myfuncs.o, you could generate names

| myfuncs.o | AMODE(31), non-XPLINK |
| myfuncsX.o | AMODE(31), XPLINK |
| myfuncs64.o | AMODE(64) (AMODE(64) always forces XPLINK) |

Your make file might generate commands such as these:

c89 -c myfuncs.c
c89 -Wc,xplink -o myfuncsX.o -c myfuncs.c
c89 -Wc,LP64 -o myfuncs64.o -c myfuncs.c
ar -ruv libmyfuncs.a myfuncs.o myfuncsX.o myfuncs64.o

To display the attributes of the symbols within an object file or an archive library of object files, use “nm - Display symbol table of object, library, or executable files” on page 489.
Exit values

0
  Successful completion.

1
  Failure due to any of the following reasons:
  • Inability to create the extracted file.
  • An error writing to the extracted file.
  • The requested module not found on appending.
  • An error opening the module on appending.
  • An incorrect module on appending.
  • Inability to access the module on appending.
  • A module not found on table or extraction.

2
  Incorrect command-line arguments or options.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
For compatibility with earlier versions, you can omit the dash that precedes the options if the options appear only as the first argument after the command name.
The following options are XPG extensions to the POSIX standard: -a, -b, -C, -i, -l, -m, -q, -s, and -t.
The -F and the -i options are extensions to the POSIX and XPG standards.

Related information
  c89, make, nm

as - Use the HLASM assembler to produce object files

Format

as
  [--option, option] ...]
  [-a|egimrsx][=file]] ...
  [-g]
  [--[no]gadata[=file]]
  [--[no]gdwarf4[=file]]
  [--moption]
  [-I name]
  [-o objectfile]
  [-d textfile]
  [-v]
  [--[no]help]
  [--[no]verbose]
  file
Description
The `as` command processes assembler source files and invokes the HLASM assembler to produce object files.

Options

```
--
```

Accepts all options that are accepted by HLASM. Multiple options can be specified by separating them with a comma. This style of option specification is designed to provide smooth migration for users who are accustomed to specifying options in JCL. For example:

```
-- "FLAG(ALIGN),RENT"
```

```
-a[egimrsx][=file]
```

Instructs the assembler to produce a listing.

```
-an
```

Instructs the assembler to produce the External Symbol Dictionary section of the assembler listing. This is equivalent to specifying: `--ESD`.

```
-ag
```

Instructs the assembly language to produce the General Purpose Register Cross Reference section of the assembler listing. This is equivalent to specifying: `--RXREF`.

```
-ai
```

Instructs the assembler to copy all product information to the list data set. This is equivalent to specifying: `--INFO`.

```
-am
```

Instructs the assembler to produce the Macro and Copy Code Source Summary section of the assembler listing. This is equivalent to specifying: `--MXREF`.

```
-ar
```

Instructs the assembler to produce the Relocation Dictionary (RLD) section of the assembler listing. This is equivalent to specifying: `--RLD`.

```
-as
```

Instructs the assembler to produce the Ordinary Symbol and Literal Cross Reference section of the assembler listing. It also instructs the assembler to produce the unreferenced symbols that are defined in the CSECTs section of the assembler listing. This is equivalent to specifying: `--XREF(SHORT,UNREFS)`.

```
-ax
```

Instructs the assembler to produce the DSECT Cross Reference section of the assembler listing. This is equivalent to specifying: `--DXREF`.

```
=file
```

Specifies the file name of the listing output. If you do not specify a file name, the output goes to standard output.

You can combine these options; for example, use `-ams` for an assembly listing with expanded macro and symbol output. The `=file` option, if used, must be specified last.

```
-g
```

Instructs the assembler to collect debug information. By default, the debug information is produced in DWARF Version 4 format (or `--gdwarf4`).

```
--[no]gadata=[file]
```

Instructs the assembler to collect associated data and write it to the associated data file. You can optionally specify the name of the output debug file. The specified name cannot be a PDS or z/OS UNIX file system directory name. If you do not specify a file name, the default name is created as follows:
• If you are compiling a data set, the `as` command uses the source file name to form the name of the output data set. The high-level qualifier is replaced with the user ID under which the `as` command is running, and .ADATA is appended as the low-level qualifier. For example, if TS12345 is compiling TSMYID.MYSOURCE(src) with this option, the produced debug file name will be TS12345.MYSOURCE.ADATA(src).

• If you are compiling a z/OS UNIX file, the `as` command stores the debug information in a file that has the name of the source file with an .ad extension. For example, if you are compiling src.a with this option, the compiler creates a debug file named src.ad.

`--[no]gdwarf4=file`  
Instructs the assembler to generate debug information conforming to the DWARF Version 4 format. Debugging tools (for example, `dbx`) can take advantage of this debug information. You can optionally specify the name of the output debug file. The file name of the output debug file must be a PDS member, a sequential data set, or z/OS UNIX file; it cannot be a PDS directory or z/OS UNIX System Services file system directory name. If you do not specify a file name, the default name is created as follows:

• If you are compiling a data set, the `as` command uses the source file name to form the name of the output data set. The high-level qualifier is replaced with the user ID under which the `as` command is running, and .DBG is appended as the low-level qualifier. For example, if TS12345 is compiling TSMYID.MYSOURCE(src) with the `-g` option, the produced debug file name will be TS12345.MYSOURCE.DBG(src). If TS12345 is compiling TSMYID.SEQSRC with the `-g` option, the produced debug file name will be TS12345.SEQSRC.DBG.

• If you are compiling a z/OS UNIX file, the `as` command stores the debug information in a file that has the name of the source file with a .dbg extension. For example, if you are compiling src.a with the `-g` option, the produced debug file name will be src.dbg.

`-m` option  
HLASM keyword options are specified using the following syntax:

```
-m<option>[=<parm>[=<value>]][:<parm>[=<value>]]...
```

where `<option>` is an option name, `<parm>` is a suboption name, and `<value>` is the suboption value. Keyword options with no parameters represent switches that can be either on or off. The keyword by itself turns on the switch, and the keyword preceded by the letters NO turns the switch off. For example, `-mLIST` tells the HLASM assembler to produce a listing and `-mNOLIST` tells the HLASM assembler not to produce a listing. If an option that represents a switch is set more than once, the HLASM assembler uses the last setting.

Keyword option and parameter names may appear in mixed case letters in the invocation command.

`-I name`  
Instructs HLASM to look for assembler macro invocation in the specified location. The `name` can be either a PDS name or z/OS UNIX file system directory name. If a PDS data set is specified, it must be fully qualified. The specified locations are then prepended to a default set of macro libraries. The `as` command assumes a default set of macro libraries that is compatible with the defaults for the C/C++ compilers. The default data sets used are: `-I CEE.SCEEMAC`, `-I SYS1.MACLIB`, and `-I SYS1.MODGEN`. The default data sets can be changed via the environment variable `_AS_MACLIB`, for example:

```
export _AS_MACLIB="FIRST.PDS:SECOND.PDS"
```

`-o objectfile`  
Specifies the name of the object file. If the name specified is a PDS or z/OS UNIX System Services directory name, a default file name is created in the PDS or z/OS UNIX directory specified as follows:

• If the source file is a sequential data set, the second last part of the data set name is used. If the data set name only contains one part after the high-level qualifier, then the last part is used.

• If the source file is a PDS member, the member name is used.

• If the source file is a z/OS UNIX file, the suffix is removed if applicable.
If the object file is going into a PDS, the first eight characters of the name is used. If there is a dot, anything after the first dot is removed.

If the object file is going into a z/OS UNIX directory, .o is appended to the name.

For example:

```
Source file: //abc.hello.source
Output file in PDS: HELLO
Output file in UNIX directory: hello.o

Source file: //ABC.HELLO'
Output file in PDS: HELLO
Output file in UNIX directory: HELLO.o

Source file: //SOURCE(hello)
Output file in PDS: HELLO
Output file in UNIX directory: hello.o

Source file: /abc/hello.s
Output file in PDS: HELLO
Output file in UNIX directory: hello.o

Source file: /abc/hellothere.s
Output file in PDS: HELLOTHE
Output file in UNIX directory: hellothere.o
```

**-d textfile**

Specifies the name of the object file output in text mode. If the name specified is a PDS or z/OS UNIX System Services directory name, a default file name is created in the PDS or z/OS UNIX directory with the same rule as **-o**.

**-v**

Writes the version of the as command to standard error.

**--[no]help**

Help menu. Displays the syntax of the as command.

**--[no]verbose**

Specifies verbose mode, which writes additional information messages to standard error.

**file** may be:

- An MVS data set (for example, //somename).
- An absolute z/OS UNIX file (for example, /somename).
- A relative z/OS UNIX file (for example, ./somename or somename).

The output of the as command is an object file. If you do not specify a file name via the **-o** option, the default name is created as follows:

- If you are compiling a data set, the as command uses the source file name to form the name of the output data set. The high-level qualifier is replaced with the user ID under which the as command is running, and .OBJ is appended as the low-level qualifier. For example, if TS12345 is compiling TSMYID.MYSOURCE(src), the compiler will create an object file named TS12345.MYSOURCE.OBJ(src).
- If you are compiling a z/OS UNIX file, the as command names the object file with the name of the source file with an .o extension. For example, if you are compiling src.a, the object file name will be src.o.

**Notes:**

- The as command does not accept standard input as a file.
- The as command invokes the HLASM assembler to produce the object file. The HLASM assembler is invoked with the default options ASA and TERM. The ASA option instructs HLASM to use American National Standard printer control characters in records written to the listing file, thus making the listing file more readable in the z/OS UNIX System Services environment. The TERM option instructs HLASM to write error messages to standard error. These defaults can be changed by using the **-m** option or **--** option.
• HLASM messages and as error messages are directed to standard error. Verbose option output is directed to standard output.
• When invoking as from the shell, any option arguments or operands specified that contain characters with special meaning to the shell must be escaped. For example, source files specified as PDS member names contain parentheses; if they are specified as fully qualified names, they contain single quotation marks. To escape these special characters, either enclose the option argument or operand in double quotation marks, or precede each character with a backslash.
• When the METAL compiler option is specified, the compiler might put a debug data block in the generated assembly file. The as utility gets the MD5 signature from the debug data block, if the block exists, and puts the signature in the debug side file. In addition, the Metal C compiler generates a placeholder for the debug side file name in the debug data block. If the as utility has the write permission to the assembly file, it will update the assembly file by replacing the debug side file name in the debug data block with the user provided name or a default debug side file name. Otherwise, the as utility will fail to update the debug side file name in the debug data block.

asa - Interpret ASA/FORTRAN carriage control

Format
asa [file ...]

Description
Historically, printouts created by programs use the first character of each line to control the spacing between that line and the previous one. For example, if the first character is a space, the rest of that line immediately follows the previous line; if it is a 1, that line should begin on a new page, and so on.

asa reads input in this format and writes it out in a normal text format, using newlines, formfeeds, and carriage returns to achieve the same effects as the carriage control characters.

If you specify files on the command line, asa reads input from these files; otherwise, it reads the standard input. asa writes output to the standard output.

It does not copy newline characters in the input to the output. Instead, it uses the first character of each line to determine how to print the rest of the line. asa interprets the first character as follows:

Space
  Outputs the rest of the line without change.

0
  Outputs a newline character before printing file.

1
  Outputs a formfeed (start a new page) sequence before printing file.

+
  Outputs a carriage return sequence so that file is output over the previous file. If + starts the first line, it is treated as a space.

Localization
asa uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH
Exit values
0  Successful completion.
1  Failure due to any of the following reasons:
   • Write error on standard output.
   • Inability to open the input file.
2  Unknown command-line option.

Portability
POSIX.2, X/Open Portability Guide, UNIX System V.

at - Run a command at a specified time

Format
at [-m] [-f file] [-q queue] -t time
at [-m] [-f file] [-q queue] timespec
at -r [-q queue] at_job...
at -l [-q queue] [at_job...]

Description
at lets you set up a series of commands to be run later. It reads the commands from the standard input (stdin) or from a file specified with the -f option. When the commands run, they have the same environment variables, working directory, file creation mask, and so on that are set up when you run the at command; however, at does not typically preserve open file descriptors, traps, or priority inherited from the working environment.

Typically, you redirect the standard output (stdout) from these commands to files so you can read the files after the system runs the commands. at mails the standard output (stdout) and standard error output (stderr) to you if you do not redirect them.

The at command displays an at_job identifier when you submit commands, along with the time that the system is to run the commands.

at, batch, and crontab submit jobs to cron; the data in these jobs may contain double-byte characters. When the jobs are run, the data in the jobs are interpreted in the locale that cron is using. Since it is strongly recommended that cron be started in the POSIX locale, double-byte characters in the job may not be interpreted correctly. You can get around this by calling setlocale() in the job itself.

Options
-f file  Reads commands from file rather than from standard input (stdin).
-l Reports on standard output (stdout) all jobs you have scheduled and when the system is to run them if you do not specify at_job. Specifying at_job reports information about those jobs only.
-m Sends you mail after your job has finished running. If you did not redirect the stdout and stderr, at also mails these to you. If stdout or stderr is non-null, at mails this output to you even if you do not specify -m.
-q queue
   Specifies the queue your at job is to be recorded in or removed from. queue can be any single-byte character except a space, a tab, a null character, or a number sign (#). By default, at stores all its jobs in a queue called a, and batch stores all its jobs in a queue called b. If used with this option, -l only reports information about at jobs in queue.

-r at_job
   Removes previously scheduled at jobs. The at_job arguments must be the identifiers assigned to the jobs when you set them up with at.

-t time
   Specifies the time for the system to run the job. Specify time in the same format as the time argument for touch.

When you do not use the -t option, you can use a timespec argument to specify the time. A timespec argument consists of three parts: a time, a date, and an increment (in that order). You must always specify the time, but you can omit the date, the increment, or both. Following are possible time formats:

Format
   Meaning

hh:mm   hh hours, mm minutes, 24-hour clock
hh:mm   hh hours, mm minutes, 24-hour clock
h:mm    h hours, mm minutes, 24-hour clock
h:m     h hours, m minutes, 24-hour clock
hh:mm zone
   zone is time zone
hh:mmam  Morning, 12-hour clock
hh:mmam zone
   Morning, 12-hour clock in given time zone
hh:mmpm  Afternoon, 12-hour clock
hh:mmpm zone
   Afternoon, 12-hour clock in given time zone
noon    Noon
midnight Midnight
varnextvar
   The current time, next day that meets date and increment
varnowvar
   The current time today

All minute specifications are optional. For example, to specify an at job to run at 1:00 p.m., you can enter

at 1pm

Currently, the z/OS shell only supports the time zones GMT, CUT, UTC, and ZULU, all of which stand for Coordinated Universal Time (often called Greenwich Mean Time). If you do not specify a zone, at interprets times with respect to the TZ environment variable.

Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.
Possible date formats are shown in the following list:

<table>
<thead>
<tr>
<th>Format</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>month day</td>
<td>The full name, or the three-letter abbreviation (as in January or Jan)</td>
</tr>
<tr>
<td>month day, year</td>
<td>Day and year given as appropriate numbers</td>
</tr>
<tr>
<td>weekday</td>
<td>Weekday is the full name or the three-letter abbreviation (as in Monday or Mon)</td>
</tr>
<tr>
<td>vartodayvar</td>
<td>The current day</td>
</tr>
<tr>
<td>vartomorrowvar</td>
<td>Next day</td>
</tr>
</tbody>
</table>

The increment is added to the time and date you specify with the preceding parts of timespec. It has the format + n units where n is a number and units is one of the following:

<table>
<thead>
<tr>
<th>Units</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>minute</td>
<td>minutes</td>
</tr>
<tr>
<td>day</td>
<td>days</td>
</tr>
<tr>
<td>month</td>
<td>months</td>
</tr>
<tr>
<td>hour</td>
<td>hours</td>
</tr>
<tr>
<td>week</td>
<td>weeks</td>
</tr>
<tr>
<td>month</td>
<td>months</td>
</tr>
<tr>
<td>year</td>
<td>years</td>
</tr>
</tbody>
</table>

Here are some sample time specifications:

- 06:55
- 18:55
- 18:55
- 6:55pm
- 6:55 pm Jan 10
- now + 3 hours
- noon tomorrow
- midnight Friday

Environment variables

*at* uses the following environment variables:

- **SHELL**
  - Contains the name of the shell used to invoke the *at* job.

- **TZ**
  - Specifies the default time zone for all times given on the command line. If you include a time zone as part of *time* or *timespec*, it overrides the value of *TZ*. Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the *TZ* environment variable.

Usage notes

*at* jobs that contain a line consisting of just the string "!!!ATEOF!!!" fail with unexpected results.

Localization

*at* uses the following localization variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_TIME
- NLSPATH

The keywords midnight, noon, today, and tomorrow are valid only in the POSIX locale.

**Exit values**

0
   Successful completion.

>0
   Returned if the command fails for any reason.

If an error occurs, **at** does not schedule, remove, or list the job.

**Portability**

POSIX.2 User Portability Extension, UNIX systems.

**Related information**

**batch, bg, cron, crontab, touch, tcsh**

---

**autoload - Indicate function name not defined**

**Format**

`autoload name ...`

**Description**

`autoload` is an alias for `typeset -fu`. Like `typeset -fu`, `autoload` indicates that the functions named in the command line are not yet defined.

See “`typeset - Assign attributes and values to variables`” on page 761 and Command execution for more information.

**Related information**

`typeset, functions, sh`

---

**automount - Configure the automount facility**

**Format**

`automount [-aeqs] [master_file_name]`

`automount -f filesystem_name`

**Description**

`automount` is used to configure the automount facility. The automount facility can automatically mount file systems at the time they are accessed, and also unmount them later. You can use a single automount policy to manage both HFS and zFS file systems. For information about setting up the automount facility, see z/OS UNIX System Services Planning.

**Important:** The automount facility will not allocate a new file system for specific entries. It only allocates new file systems for generic (*) entries.

`automount` requires superuser authority.

Run `automount` from the `/etc/rc` script with no arguments. This action processes the installation's default `automount` configuration file. When run with no arguments, `automount` reads the `/etc/`
auto.master file to determine all directories that are to be configured for the automount and the file names that contain their configuration specifications.

The /etc/auto.master file contains the directory or directories that the automount facility will monitor. It also contains an associated MapName file that contains the mount parameters. The name of the map file can be specified as an MVS data set name. The data set name must be specified as a fully qualified name and can be uppercase or lowercase. Single quotation marks are not needed.

If the automount policy is loaded, you will get a return code of 0. A nonzero return code indicates that the policy was not loaded.

The automount file system (*AMD/) is mounted with an automove attribute of either AUTOMOVE or UNMOUNT. The automove attribute is set to UNMOUNT only when its parent file system has its automove attribute set to UNMOUNT. When the automove attribute is set to UNMOUNT, the owning system of the automount file system is identical to the owning system of the parent.

If you run automount with the [master filename] argument, that file name is used instead of /etc/auto.master.

Tip: zFS is the preferred file system and continued use of HFS is discouraged. New file systems should be created as zFS file systems.

Options

-a

Indicates that the policy being loaded is to be appended to the existing policy rather than replace the existing policy. For example:

```
/usr/sbin/automount -a
```

-a is mutually exclusive with -q.

-e

Displays recent error information from automount attempting to create a new zFS or HFS file system. Typically, one allocation error value and reason code is displayed for the last allocation error, if there was one. If a zFS file system could not be created, you will see message text or error and reason codes (or both) for each automount-managed directory where the zFS file system was to be created.

-f

Displays the information of the job that last accessed the specified file system. The file system name must be specified and it is treated as case-insensitive. All automount-managed file systems that match is reported. The information includes file system name, mount point, state, timer, UID, PID, and job name. The state has two values: duration and delay. The timer is the minutes that remain for the specified file system to be in this state.

-q

Displays the current automount policy. -q is mutually exclusive with -a.

-s

Checks the syntax of the configuration file. No automount is performed.

Examples

1. The following example shows how automatic unmount can be avoided for a directory:

<table>
<thead>
<tr>
<th>name</th>
<th>duration</th>
<th>wjs</th>
<th>nolimit</th>
</tr>
</thead>
</table>

Keywords that are not specified on a specific entry are inherited from the generic entry, if present. If the generic entry is not present, or if keys are not specified, the defaults are used. If the file system key cannot be resolved, the entry is considered invalid. The filesystem attribute for a specific entry must exist, and will never be created using the inherited allocany values.
2. The following example is a /etc/auto.master file that is used to specify /u as automount-managed and the specifications for that directory in /etc/u.map:

```
/u   /etc/u.map
```

Files

`automount` uses these files:

`/etc/auto.master`

Specifies a list of directories to be configured, along with their MapName files.

Each line in this file contains two path names that are separated by at least one space: the directory name to be managed and the path name of the MapName file. Both of these path names must be absolute.

The path name of the managed directory is used as a file system name prefixed with *AMD/. This restricts the length of the path name of a managed directory to 40 characters. If path names need to be longer, you can use symbolic links to resolve all or part of the path name.

Blank lines and lines beginning with the characters */ are considered comments and are ignored. Line comments are not tolerated.

Tip: While MVS system symbols can be used in master files such as &ZOSREL, only use static system symbols in order to avoid unexpected results. The symbols are resolved when the automount policy is loaded. If the symbol is dynamically changed after the policy is loaded, the policy must be reloaded in order to have the symbol resolved again. To display the symbol substitution, use the `automount -q` option.

MapName

The MapName file contains the mapping between a subdirectory of a directory that is managed by `automount` and the mount parameters.

The file is organized as a set of specifications. Each specification contains one or more lines. Each line is of the form `keyword argument`. Each specification must begin with the keyword `name`.

Blank lines and lines that begin with the characters * are considered comments and are ignored. Line comments are not tolerated.

A line can be continued by having a backslash character (\) at the end. The leading tabs and blanks in the continuation lines are ignored. The tabs and blanks before the continuation character (\) in the continued lines are not ignored. For example:

```
parm mynfs.ibm.com:/SY1/tmp/fs1/abcd\
efgh/myfs,\nXLAT(Y)
```

A generic entry can be specified as the first specification by using the name of *. The generic specification provides defaults for subsequent specific specifications. When the automount facility tries to resolve a lookup request, it attempts to find a specific entry. If a specific entry does not exist for the name that is being looked up, it attempts to use the generic entry.

The following example shows a generic entry:

```
name   *
type   HFS
filesystem  OMVS.HFS.USER.<uc_name>
mode    rdwr
duration 30
delay   10
parm    SYNC(60)
tag     text,819
```

These special symbols provide name substitution:
• `<asis_name>` used to represent the name exactly, as is.
• `<uc_name>` used to represent the name in uppercase characters.
• `<sysname>` or `&SYSNAME`. used to substitute the system name.

Use `&SYSNAME`. because `<sysname>` is only temporarily supported for compatibility.

You can use these symbols when specifying a file system name or file system parameter that has a specific form with the name inserted as a qualifier.

Following is a list of supported keywords. You can enter keywords using mixed case letters. Some arguments require mixed case. The allocany, allocuser, and lowercase keywords are valid on any specification, but are meaningful only on the generic entry. In other words, `automount` will not allocate a new filesystem for specific entries, only for generic (*) entries.

**Restriction:** The filesystem attribute for a specific entry must exist, and will never be created using the inherited allocany values.

**allocany allocation-spec**
Specifies the allocation parameters when using `automount` to allocate HFS or zFS file systems, keeping in mind that zFS is the preferred file system. Specifying the allocany keyword causes an allocation if the data set does not exist for any name looked up in the automount-managed directory.

The automount facility creates a new zFS file system as an HFS-compatible file system if the file system that was specified in the automount policy does not exist. Space for zFS file systems is always assumed to be in units of cylinders regardless of other specifications. All other allocation keywords that can be used for HFS can be specified but will be ignored. However, the syntax must be correct. These restrictions are in place so that migration to zFS or back to HFS will require minimal changes to the automount policy. See usage note “5” on page 33.

**allocation-spec**
A string that specifies allocation keywords. The keywords in Table 2 on page 30 can be specified in the string.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>zFS</th>
<th>HFS</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataclas</td>
<td>Applied</td>
<td>Applied</td>
<td>Specifies the data class for the data set.</td>
</tr>
<tr>
<td>euid</td>
<td>Applied</td>
<td>Applied</td>
<td>The new data set owner is set to the effective UID and GID.</td>
</tr>
<tr>
<td>maxvol</td>
<td>Ignored</td>
<td>Applied</td>
<td>Number of volumes for a multivolume data set.</td>
</tr>
<tr>
<td>mgmtclas</td>
<td>Applied</td>
<td>Applied</td>
<td>Specifies the management class for the data set.</td>
</tr>
<tr>
<td>pathperm</td>
<td>Applied</td>
<td>Not supported</td>
<td>Specifies permission to the root directory. In order to use the pathperm keyword, all systems in a shared file system configuration must be at least the z/OS V2R1 level.</td>
</tr>
<tr>
<td>space</td>
<td>Applied</td>
<td>Applied</td>
<td>Specifies primary and optional secondary space allocations.</td>
</tr>
<tr>
<td>storclas</td>
<td>Applied</td>
<td>Applied</td>
<td>Specifies the storage class for the data set.</td>
</tr>
<tr>
<td>cyl</td>
<td>tracks</td>
<td>block</td>
<td>Applied</td>
</tr>
<tr>
<td>unit</td>
<td>Ignored</td>
<td>Applied</td>
<td>Specifies the unit name, device type, or unit address.</td>
</tr>
<tr>
<td>vol</td>
<td>Applied</td>
<td>Applied</td>
<td>Specifies the serial numbers for eligible direct access volumes where the data set is to reside.</td>
</tr>
<tr>
<td>ucat</td>
<td>Applied</td>
<td>Not supported</td>
<td>The high-level qualifier for the zFS data set must be an alias in the master catalog.</td>
</tr>
</tbody>
</table>

Table 2: Allocation-spec keywords for allocany and allocuser
Example: An example of allocany is as follows:

```
allocany   space(100,20)  cyl  vol(MYDISK)  pathperm(777)  euid
```

**allocuser allocation-spec**

Specifies the allocation parameters when using automount to allocate HFS or zFS file systems, keeping in mind that zFS is the preferred file system. Allocation occurs only if the name being looked up matches the user ID of the current user.

The automount facility creates a new zFS file system as an HFS-compatible file system if the file system that was specified in the automount policy does not already exist. Space for zFS file systems is always assumed to be in units of cylinders regardless of other specifications. All other allocation keywords that can be used for HFS can be specified but are ignored. However, the syntax must be correct. These restrictions are in place so that migration to zFS or back to HFS will require minimal changes to the automount policy. See usage note “5” on page 33.

**allocation-spec**

A string that specifies allocation keywords. The keywords in Table 2 on page 30 can be specified in the string.

**charcase [lower|upper|asis]**

Indicates the case for names that can match the * specification. This keyword is valid on any specification but is only meaningful on the generic entry. This keyword is mutually exclusive with the lowercase keyword.

**lower**

Only names that are composed of lowercase characters can match the * specification. Numbers and special characters can also be used. When this keyword is specified, uppercase characters are not allowed. This option is equivalent to lowercase yes.

**upper**

Only names that are composed of uppercase characters can match the * specification. Numbers and special characters can also be used. When this keyword is specified, lowercase characters are not allowed.

**asis**

Any name can match the * specification. This keyword is the default and is equivalent to lowercase no.

**delay**

The minimum amount of time in minutes to leave the file system mounted after the duration expires and the file system is no longer in use. The default is 10.

**Tip:** In a shared file system environment, specify a delay time of at least 10.

**duration**

The minimum amount of time in minutes to leave the file system mounted. The default is nolimit.

**filesystem**

The name of the file system to mount. This argument is case-sensitive. For the HFS file system, this argument must be specified in uppercase.

**Restriction:** Symbol symbolics that are used by the file system name template cannot be more than 44 characters long. Symbolics that are used for the automount (<sysname>, <asis_name>, <uc_name>) are resolved within automount as part of checking the length of the file system name template.

**lowercase [Yes|No]**

Indicates the case for names that can match the * specification. This keyword is valid on any specification, but is only meaningful on the generic entry. It is also mutually exclusive with the charcase keyword.
Yes
Only names that are composed of lowercase characters can match the * specification (numbers and special characters can also be used). When this is specified, uppercase characters are not allowed. Yes is equivalent to charcase lower.

No
Any names can match the * specification. This is the default and is equivalent to charcase asis.

mode
The mount mode for the file system (rdwr or read). The default is rdwr.

name
The name of the directory to be mounted. This key is required and must be the first key that is specified for the entry. If the first entry specifies name *, it is treated as the generic entry for the automount-managed directory.

parm
The file system-specific parameter. This argument is case-sensitive. For example, the following parameters can be specified for an HFS file system:

| parm   | SYNC(t),NOWRITEPROTECT |

security [Yes|No]
Specifies security checking for files in the file system. You can specify these values:

Yes
Normal security checking is done. Yes is the default.

No
Specifies that security checks will not be enforced for files in this file system. Any user can access or change any file or directory in any way.

Security auditing will still be performed if the installation is auditing successes.

The SETUID, SETGID, APF, and Program Control mode bits can be turned on in files from this file system, but are not honored while it is mounted with NOSECURITY. When a file system is mounted with the NOSECURITY option enabled, any new files or directories that are created are assigned an owner of UID 0, no matter what UID issued the request.

Tip: The installation should normally take the default (Yes).

For more information about mounting with no security and on the MOUNT statement in BPXPRMxx, see z/OS UNIX System Services Planning. Security keywords on the TSO MOUNT command are also discussed in "mount - Logically mount a file system" on page 468.

setuid [Yes|No]
Specifies whether the setuid and setgid mode bits are to be respected for executables run from this file system. You can specify these values:

Yes
The setuid and setgid modes are respected. Yes is the default.

No
The setuid and setgid modes are ignored.

tag (text|notext,ccsid)
Specifies whether file tags for untagged files in the mounted file system are implicitly set. Either text or notext, and ccid (coded character set identifier) must be specified when tag is specified:

text
Specifies that each untagged file is implicitly marked as containing pure text data that can be converted.

notext
Specifies that none of the untagged files in the file system are automatically converted during file reading and writing.
ccsid
Identifies the coded character set identifier to be implicitly set for the untagged file. ccsid is specified as a decimal value from 0 to 65535. However, when text is specified, the value must be between 0 and 65535. Other than this, the value is not checked as being valid and the corresponding code page is not checked as being installed.

For more information about file tagging, see z/OS UNIX System Services Planning. More information about the TAG parameter can be found in "mount - Logically mount a file system" on page 468.

type
The type of the file system (such as HFS, zFS, and NFS). The default is HFS.

Usage notes
1. When a new file system of the type HFS is created and allocated to a new user, the owner UID and GID are based on that user. The setting of the permission bits is 700. By default, automount uses the UID and GID of the user ID that owns the process. If the euid keyword is specified for allocany or allocuser, the thread-level UID and GID are used instead.

2. When a new file system of the type zFS is created and allocated to a new user, the owner UID and GID are based on that user. The permission is set to the value of pathperm (the default is 750). If permission is not specified, or if the value is 000, the default is used. To display the pathperm value, whether or not it is specified for allocany and allocuser, use the automount -q option. By default, automount uses the UID and GID of the user ID owning the process. If the euid keyword is specified for allocany or allocuser, the thread-level UID and GID are used instead.

3. The syntax of the automount master file is extended to optionally include the name of the filter utility. Each line contains:
   • The path name of the directory that is to be managed.
   • The path name of the map file.
   • An optional path name of the conversion utility.

   If a conversion utility is specified, automount runs that utility and provide the specified map file as the standard input for the utility. It processes the standard output from the utility as the automount map file and list it on its standard output. Errors that are detected by the automount facility are flagged the same as before, but line numbers will refer to the line as output from the conversion utility rather than the original map file that the utility processes.

4. automount recognizes the type specification in the automount map files of HFS and zFS as potentially interchangeable file system types. At the time automount applies the specification for the mount, it determines if the file system is the name of either an zFS or HFS file system and alters the type as appropriate. If the data set does not exist and if allocany or allocuser is specified, a new file system is allocated as the file system type as specified in type. Allocation is only done if allocany or allocuser is specified. If it is preferred to have new file systems allocated as zFS file systems, the automount policy should be changed to specify type zFS.

   This allows automount-managed file systems to be changed from HFS to zFS without changing the file system name and without changing the automount policy. If the file system name must be changed, it is necessary to add a specific entry in the automount policy for this file system or manage it on another managed directory.

5. When the allocation-spec keyword TRACKS or BLOCK is specified in either the allocany or allocuser option for zFS file systems, the specified SPACE(\) units are converted to approximate CYL equivalent units before the zFS file system is allocated.

   The following formulas are used to do the conversion into CYL units:

   1 TRACKS Unit = 1/15 CYL Unit
   1 BLOCK Unit = 1/180 CYL Unit
The conversion that is used does not consider the device type.

6. The // placeholder is supported when used with the allocany or allocuser keywords in automount policy files to create new file systems.

7. The steps for preparing RACF in z/OS UNIX System Services Planning include a suggestion to make the kernel address space trusted. If you did not make the local address space trusted, you must give the kernel access to the local data sets.

If allocany and allocuser are used, file systems are created and cataloged under the authority of the OMVS address space. The kernel address space must be trusted or mount failures will occur. Also, if zFS is being used, then the zFS started task address space must also be trusted in order to properly identify the data set as ZFS.

If ucat is specified and an alias does not exist for a new zFS file system allocation, the automount will fail with the following reason code:

jrNoUserCat Rsn0695. Keyword UCAT is ignored for the HFS file system.

Before you use the MVS user ID as the high level qualifier for the allocation of automount-managed file systems, first review Security considerations for the automount policy in z/OS UNIX System Services Planning.

Related information
chmount, mount, unmount

awk - Process programs written in the awk language

Format

awk [-F ere] [-v var=value ...] [program] [var=value ...] [file ...]
awk [-F ere] [-F prog] [-v var=value ...] [var = value ...] [file ...]

Description

awk is a file-processing language that is well suited to data manipulation and retrieval of information from text files. If you are unfamiliar with the language, you might find it helpful to read the awk information in z/OS UNIX System Services User’s Guide first.

An awk program consists of any number of user-defined functions and rules of the form:

pattern {action}

There are two ways to specify the awk program:

- Directly on the command line. In this case, program is a single command-line argument, typically enclosed in single quotation marks (‘’) to prevent the shell from attempting to expand it.
- By using the -F prog option.

You can specify program directly on the command line only if you do not use any -F prog arguments.

For a summary of the UNIX03 changes to this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Options

awk recognizes the following options:

-F ere

Is an extended regular expression to use as the field separator.
-F prog

Runs the awk program contained in the file prog. When more than one -F option appears on the command line, the resulting program is a concatenation of all programs you specify.

-v var=value

Assigns value to var before running the program.

Files that you specify on the command line with the file argument provide the input data for awk to manipulate. If you specify no files or you specify a dash as a file, awk reads data from standard input (stdin).

You can initialize variables on the command line using:

\[ var=value \]

You can intersperse such initializations with the names of input files on the command line. awk processes initializations and input files in the order they appear on the command line. For example, the command:

\[ awk -f progfile a=1 f1 f2 a=2 f3 \]

sets a to 1 before reading input from f1 and sets a to 2 before reading input from f3.

Variable initializations that appear before the first file on the command line are performed immediately after the BEGIN action. Initializations appearing after the last file are performed immediately before the END action. For more information about BEGIN and END, see “Patterns” on page 43.

Use the -v option to assign a value to a variable before the awk program begins execution (that is, before the BEGIN action). For example, in:

\[ awk -v v1=10 -f prog datafile \]

awk assigns the variable v1 its value before the BEGIN action of the program (but after default assignments made to such built-in variables as FS and OFMT; these built-in variables have special meaning to awk).

awk divides input into records. By default, newline characters separate records; however, you can specify a different record separator if you want.

One at a time, and in order, awk compares each input record with the pattern of every rule in the program. When a pattern matches, awk performs the action part of the rule on that input record. Patterns and actions often refer to separate fields within a record. By default, white space (usually blanks, newlines, or horizontal tab characters) separates fields; however, you can specify a different field separator string using the -F ere option).

You can omit the pattern or action part of an awk rule (but not both). If you omit pattern, awk performs the action on every input record (that is, every record matches). If you omit action, awk's default action is equivalent to: \{print\}.

awk considers everything after a # in a program line to be a comment. For example:

\[ # This is a comment \]

To continue program lines on the next line, add a backslash (\) to the end of the line. Statement lines ending with a comma (,), double or-bars (||), or double ampersands (&&) continue automatically on the next line.

Variables and expressions

There are three types of variables in awk: identifiers, fields, and array elements.

An identifier is a sequence of letters, digits, and underscores beginning with a letter or an underscore. These characters must be from the POSIX portable character set. (Data can come from other character sets.)
For a description of fields, see “Input” on page 39.

Arrays are associative collections of values called the elements of the array. Constructs of the form:

```
identifier[subscript]
```

where subscript has the form expr or expr,expr,..., refer to array elements. Each such expr can have any string value. For multiple expr subscripts, awk concatenates the string values of all expr arguments with a separate character SUBSEP between each. The initial value of SUBSEP is set to \042 (code page 01047 field separator).

Fields and identifiers are sometimes referred to as scalar variables to distinguish them from arrays.

You do not declare awk variables, and you do not need to initialize them. The value of an uninitialized variable is the empty string in a string context and the number 0 in a numeric context.

Expressions consist of constants, variables, functions, regular expressions, and subscript-in-array conditions combined with operators. (Subscript-in-array conditions are described in Subscript in array.) Each variable and expression has a string value and a corresponding numeric value; awk uses the value appropriate to the context.

When converting a numeric value to its corresponding string value, awk performs the equivalent of a call to the sprintf() function where the one and only expr argument is the numeric value and the fmt argument is either %d (if the numeric value is an integer) or the value of the variable CONVFMT (if the numeric value is not an integer). The default value of CONVFMT is % .6g. If you use a string in a numeric context, and awk cannot interpret the contents of the string as a number, it treats the value of the string as zero.

Numeric constants are sequences of decimal digits.

String constants are quoted, as in "a literal string". Literal strings can contain the following escape sequences:

<table>
<thead>
<tr>
<th>Escape character</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>\a</td>
<td>Audible bell</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace</td>
</tr>
<tr>
<td>\f</td>
<td>Form feed</td>
</tr>
<tr>
<td>\n</td>
<td>Newline</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage return</td>
</tr>
<tr>
<td>\t</td>
<td>Horizontal tab</td>
</tr>
<tr>
<td>\v</td>
<td>Vertical tab</td>
</tr>
<tr>
<td>\ooo</td>
<td>Octal value ooo</td>
</tr>
<tr>
<td>\xdd</td>
<td>Hexadecimal value dd</td>
</tr>
<tr>
<td>/</td>
<td>Slash</td>
</tr>
<tr>
<td>&quot;</td>
<td>Quote</td>
</tr>
<tr>
<td>\c</td>
<td>Any other character c</td>
</tr>
</tbody>
</table>

awk supports full regular expressions. (See Appendix C, “Regular expressions (regexp),” on page 945 for more information.) When awk reads a program, it compiles characters that are enclosed in slash characters (/) as regular expressions. In addition, when literal strings and variables appear on the right side of a ~ or!~ operator, or as certain arguments to built-in matching and substitution functions, awk interprets them as dynamic regular expressions.
**Note:** When you use literal strings as regular expressions, you need extra backslashes to escape regular expression metacharacters, because the backslash is also the literal string escape character. For example, the regular expression:

```
/e\.g\./
```

should be written as:

```
"e\.g\\."
```

**Subscript in array**

**awk** defines the subscript-in-array condition as:

```
index in array
```

where `index` looks like `expr` or `(expr,...,expr)`. This condition evaluates to 1 if the string value of `index` is a subscript of `array`, and to 0 otherwise. This is a way to determine if an array element exists. When the element does not exist, the subscript-in-array condition does not create it.

**Symbol table**

You can access the symbol table through the built-in array SYMTAB. SYMTAB[`expr`] is equivalent to the variable named by the evaluation of `expr`.

For example, SYMTAB["var"] is a synonym for the variable `var`.

**Environment**

An **awk** program can determine its initial environment by examining the ENVIRON array. If the environment consists of entries of the form `name=value`, then ENVIRON[`name`] has string value "value". For example, the following program is equivalent to the default output of `env`:

```
BEGIN {        for (i in ENVIRON)            printf("%s=%s\n", i, ENVIRON[i])            exit }    
```

**Operators**

**awk** follows the usual precedence order of arithmetic operations, unless overridden with parentheses; a table giving the order of operations appears later in this topic.

The unary operators are `+`, `-`, `++`, and `- -`, where you can use the `++` and `- -` operators as either postfix or prefix operators, as in C. The binary arithmetic operators are `+`, `-`, `*`, `/`, `%`, and `^`.

The conditional operator

```
expr ? expr1 : expr2
```

evaluates to `expr1` if the value of `expr` is nonzero, and to `expr2` otherwise.

If two expressions are not separated by an operator, **awk** concatenates their string values.
The tilde operator ~ yields 1 (true) if the regular expression on the right side matches the string on the left side. The operator !~ yields 1 when the right side has no match on the left. To illustrate:

\$2 ~ /[0-9]/

selects any line where the second field contains at least one digit. awk interprets any string or variable on the right side of ~ or !~ as a dynamic regular expression.

The relational operators are <, <=, >, >=, ==, and !=. When both operands in a comparison are numeric, or if one is numeric and the other is not initialized, awk compares their values numerically; otherwise, it compares them as strings. An operator is considered to be numeric if it consists of any of the following:

- An integer or floating-point number.
- A field, FILENAME, ARGV array element, or ENVIRON array element that looks like a number.
- A variable created by a command-line assignment that looks like a number.
- Input from a getline() function that looks like a number.
- An array element created by the split() function that looks like a number.
- A variable assignment from another number variable that looks like a number.

The Boolean operators are || (or), && (and), and ! (not). awk uses short-circuit evaluation when evaluating expressions. With an && expression, if the first operator is false, the entire expression is false and it is not necessary to evaluate the second operator. With an || expression, a similar situation exists if the first operator is true.

You can assign values to a variable with:

\[ \text{var} = \text{expr} \]

If op is a binary arithmetic operator, var op= expr is equivalent to var = var op expr, except that var is evaluated only once.

See Table 3 on page 38 for the precedence rules of the operators.

**Table 3: The order of operations for awk**

<table>
<thead>
<tr>
<th>Operators</th>
<th>Order of operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>Grouping</td>
</tr>
<tr>
<td>$i V[a]</td>
<td>Field, array element</td>
</tr>
<tr>
<td>V++ V-- ++V --V</td>
<td>Increment, decrement</td>
</tr>
<tr>
<td>A^B</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>+A -A !A</td>
<td>Unary plus, unary minus, logical NOT</td>
</tr>
<tr>
<td>A*B A/B A%B</td>
<td>Multiplication, division, remainder</td>
</tr>
<tr>
<td>A+B A-B</td>
<td>Addition, subtraction</td>
</tr>
<tr>
<td>A B</td>
<td>String concatenation</td>
</tr>
<tr>
<td>A&lt;B A&gt;B A&lt;=B&gt;=B A!=B A==B</td>
<td>Comparisons</td>
</tr>
<tr>
<td>A in V</td>
<td>Regular expression matching</td>
</tr>
<tr>
<td>A &amp;&amp; B</td>
<td>Array membership</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A ? B : C</td>
<td>Logical OR</td>
</tr>
<tr>
<td>V=B V+=B V-=B V*=B V/=B V%=B V^=B</td>
<td>Conditional expression</td>
</tr>
<tr>
<td></td>
<td>Assignment</td>
</tr>
</tbody>
</table>
Note:
1. A, B, C are any expression.
2. i is any expression yielding an integer.
3. V is any variable.

Command-line arguments

`awk` sets the built-in variable ARGC to the number of command-line arguments. The built-in array ARGV has elements subscripted with digits from zero to ARGC - 1, giving command-line arguments in the order they appeared on the command line.

The ARGC count and the ARGV vector do not include command-line options (beginning with `-`) or the program file (following `-F`). They do include the name of the command itself, initialization statements of the form `var = value`, and the names of input data files.

`awk` creates ARGC and ARGV before doing anything else. It then walks through ARGV, processing the arguments. If an element of ARGV is an empty string, `awk` skips it. If it contains an equals sign (`=`), `awk` interprets it as a variable assignment. If it is a minus sign (`-`), `awk` immediately reads input from stdin until it encounters the end of the file. Otherwise, `awk` treats the argument as a file name and reads input from that file until it reaches the end of the file. `awk` runs the program by walking through ARGV in this way. Thus, if the program changes ARGV, `awk` can read different files and make different assignments.

Input

`awk` divides input into records. A record separator character separates each record from the next. The value of the built-in variable RS gives the current record separator character; by default, it begins as the newline (`\n`). If you assign a different character to RS, `awk` uses that as the record separator character from that point on.

`awk` divides records into fields. A field separator string, given by the value of the built-in variable FS, separates each field from the next. You can set a specific separator string by assigning a value to FS, or by specifying the `-F` *ere* option on the command line. You can assign a regular expression to FS. For example:

```
FS = "[,:$]"
```

says that commas, colons, or dollar signs can separate fields. As a special case, assigning FS a string that contains only a blank character sets the field separator to white space. In this case, `awk` considers any sequence of contiguous space or tab characters a single field separator. This is the default for FS. However, if you assign FS a string containing any other character, that character designates the start of a new field. For example, if we set FS=`\t` (the tab character),

```
texta  \t textb  \t  \t  \t textc
```

contains five fields, two of which contain only blanks. With the default setting, this record only contains three fields, since `awk` considers the sequence of multiple blanks and tabs a single separator.

The following list of built-in variables provides various pieces of information about input:

**NF**
Number of fields in the current record

**NR**
Number of records read so far

**FILENAME**
Name of file that contains the current record

**FNR**
Number of records that was read from current file
Field specifiers have the form $n$, where $n$ runs from 1 through NF. Such a field specifier refers to the $n$th field of the current input record. $0$ (zero) refers to the entire current input record.

The getline function can read a value for a variable or $0$ from the current input, from a file, or from a pipe. The result of getline is an integer indicating whether the read operation was successful. A value of 1 indicates success; 0 indicates that the end of the file was encountered; and -1 indicates that an error occurred. Possible forms for getline are:

- **getline**
  - Reads next input record into $0$ and splits the record into fields. NF, NR, and FNR are set appropriately.
- **getline var**
  - Reads the next input record into the variable var. `awk` does not split the record into fields (which means that the current $n$ values do not change), but sets NR and FNR appropriately.
- **getline expr**
  - Interprets the string value of `expr` to be a file name. `awk` reads the next record from that file into $0$, splits it into fields, and sets NF appropriately. If the file is not open, `awk` opens it. The file remains open until you close it with a close function.
- **getline var <expr**
  - Interprets the string value of `expr` to be a file name, and reads the next record from that file into the variable var, but does not split it into fields.
- **expr | getline**
  - Interprets the string value of `expr` as a command line to be run. `awk` pipes output from this command into getline, and reads it into $0$, splits it into fields, and sets NF appropriately. See “System function” on page 42 for additional details.
- **expr | getline var**
  - Runs the string value of `expr` as a command and pipes the output of the command into getline. The result is similar to `getline var <expr`.

You can have only a limited number of files and pipes open at one time. You can close files and pipes during execution using the `close(expr)` function. The `expr` argument must be one that came before `|` or after `<` in getline, or after `>` or `>>` in `print` or printf.

If the function successfully closes the pipe, it returns zero. By closing files and pipes that you no longer need, you can use any number of files and pipes in the course of running an `awk` program.

**Built-in arithmetic functions**

- **atan2(expr1, expr2)**
  - Returns the arctangent of `expr1/expr2` in the range of $-\pi$ through $\pi$.
- **exp(expr), log(expr), sqrt(expr)**
  - Returns the exponential, natural logarithm, and square root of the numeric value of `expr`. If you omit (expr), these functions use $0$ instead.
- **int(expr)**
  - Returns the integer part of the numeric value of `expr`. If you omit (expr), the function returns the integer part of $0$.
- **rand()**
  - Returns a random floating-point number in the range 0 through 1.
- **sin(expr), cos(expr)**
  - Returns the sine and cosine of the numeric value of `expr` (interpreted as an angle in radians).
- **srand(expr)**
  - Sets the seed of the `rand` function to the integer value of `expr`. If you omit (expr), `awk` uses the time of day as a default seed.
**Built-in string functions**

\[ \text{len} = \text{length} (\text{expr}) \]

Returns the number of characters in the string value of \text{expr}. If you omit \text{(expr)}, the function uses $0$ instead. The parentheses around \text{expr} are optional.

\[ \text{n} = \text{split} (\text{string}, \text{array}, \text{regexp}) \]

Splits the \text{string} into fields. \text{regexp} is a regular expression giving the field separator string for the purposes of this operation. This function assigns the separate fields, in order, to the elements of \text{array}; subscripts for \text{array} begin at 1. \text{awk} discards all other elements of \text{array}. \text{split} returns the number of fields into which it divided \text{string} (which is also the maximum subscript for \text{array}).

\text{regexp} divides the record in the same way that the FS field separator string does. If you omit \text{regexp} in the call to \text{split}, it uses the current value of FS.

\[ \text{str} = \text{substr} (\text{string}, \text{offset}, \text{len}) \]

Returns the substring of \text{string} that begins in position \text{offset} and is at most \text{len} characters long. The first character of the string has an \text{offset} of 1. If you omit \text{len}, or if \text{len} specifies more characters than are left in the string, \text{substr} returns the rest of \text{string}.

\[ \text{pos} = \text{index} (\text{string}, \text{str}) \]

Returns the position of the first occurrence of \text{str} in \text{string}. The count is in characters. If \text{index} does not find \text{str} in \text{string}, it returns 0.

\[ \text{pos} = \text{match} (\text{string}, \text{regexp}) \]

Searches \text{string} for the first substring matching the regular expression \text{regexp}, and returns an integer giving the position of this substring counting from 1. If it finds no such substring, \text{match} returns zero. This function also sets the built-in variable RSTART to \text{pos} and the built-in variable RLENGTH to the length of the matched string. If it does not find a match, \text{match} sets RESTART to 0, and RLENGTH to -1. You can enclose \text{regexp} in slashes or specify it as a string.

\[ \text{n} = \text{sub} (\text{regexp}, \text{repl}, \text{string}) \]

Searches \text{string} for the first substring matching the regular expression \text{regexp}, and replaces the substring with the string \text{repl}. \text{awk} replaces any ampersand (&) in \text{repl} with the substring of \text{string} which matches \text{regexp}. An ampersand preceded with a backslash (\text{"\textbackslash"}) is interpreted as the literal ampersand character. An occurrence of two consecutive backslashes is interpreted as just a single literal backslash character. Any other occurrence of a backslash (for example, preceding any other character) is treated as a literal backslash character. If \text{repl} is a string literal, the handling of the ampersand character occurs after any lexical processing, including any lexical backslash escape sequence. If you omit \text{string}, \text{sub} uses the current record instead. \text{sub} returns the number of substrings replaced (which is 1 if it found a match, and 0 otherwise).

\[ \text{n} = \text{gsub} (\text{regexp}, \text{repl}, \text{string}) \]

Works the same way as \text{sub}, except that \text{gsub} replaces all matching substrings (global substitution). The return value is the number of substitutions performed.

\[ \text{str} = \text{sprintf} (\text{fmt}, \text{expr}, \text{expr}...) \]

Formats the expression list \text{expr}, \text{expr}, ... using specifications from the string \text{fmt}, and then returns the formatted string. The \text{fmt} string consists of conversion specifications that convert and add the next \text{expr} to the string, and ordinary characters that \text{sprintf} simply adds to the string. These conversion specifications are similar to those used by the American National Standard C standard.

Conversion specifications have the form

\[
\%
\{[-]\}[0][x][.y]c
\]

where

- Left-justifies the field; default is right justification.
0
(Leading zero) prints numbers with leading zero.

x
Is the minimum field width.

y
Is the precision.

c
Is the conversion character.

In a string, the precision is the maximum number of characters to be printed from the string; in a number, the precision is the number of digits to be printed to the right of the decimal point in a floating-point value. If \( x \) or \( y \) is \(*\) (asterisk), the minimum field width or precision is the value of the next \( expr \) in the call to \texttt{sprintf}.

The conversion character \( c \) is one of following:

\texttt{d}
Decimal integer.

\texttt{i}
Decimal integer.

\texttt{o}
Unsigned octal integer.

\texttt{x,X}
Unsigned hexadecimal integer.

\texttt{u}
Unsigned decimal integer.

\texttt{f,F}
Floating point.

\texttt{e,E}
Floating point (scientific notation).

\texttt{g,G}
The shorter of \texttt{e} and \texttt{f} (suppresses nonsignificant zeros).

\texttt{c}
Single character of an integer value; first character of string.

\texttt{s}
String.

The lowercase \texttt{x} specifies alphabetic hexadecimal digits in lowercase, whereas the uppercase \texttt{X} specifies alphabetic hexadecimal digits in uppercase. The other uppercase-lowercase pairs work similarly.

\( n = \texttt{ord(expr)} \)
Returns the integer value of first character in the string value of \texttt{expr}. This is useful in conjunction with \%c in \texttt{sprintf}.

\( str = \texttt{tolower(expr)} \)
Converts all letters in the string value of \texttt{expr} into lowercase, and returns the result. If you omit \texttt{expr}, \texttt{tolower} uses $0$ instead. This function uses the value of the locale or the LC_CTYPE environment variable.

\( str = \texttt{toupper(expr)} \)
Converts all letters in the string value of \texttt{expr} into uppercase, and returns the result. If you omit \texttt{expr}, \texttt{toupper} uses $0$ instead. This function uses the value of the locale or the LC_CTYPE environment variable.
status = system(expr)

Runs the string value of expr as a command. For example, system("tail " $1) calls the tail command, using the string value of $1 as the file that tail examines. The standard command interpreter runs the command, as discussed in the “Portability” on page 6 section, and the exit status that is returned depends on that command interpreter.

User-defined functions

You can define your own functions using the form:

```
function name(parameter-list) {
    statements
}
```

A function definition can appear in the place of a pattern {action} rule. The parameter-list argument contains any number of normal (scalar) and array variables that are separated by commas. When you call a function, awk passes scalar arguments by value, and array arguments by reference. The names specified in parameter-list are local to the function; all other names that are used in the function are global. You can define local variables by adding them to the end of the parameter list as long as no call to the function uses these extra parameters.

A function returns to its caller either when it runs the final statement in the function, or when it reaches an explicit return statement. The return value, if any, is specified in the return statement (see “Actions” on page 43).

Patterns

A pattern is a regular expression, a special pattern, a pattern range, or any arithmetic expression.

BEGIN is a special pattern used to label actions that awk performs before reading any input records. END is a special pattern that is used to label actions that awk performs after reading all input records.

You can give a pattern range as:

```
pattern1,pattern2
```

This matches all lines from one that matches pattern1 to one that matches pattern2, inclusive.

If you omit a pattern, or if the numeric value of the pattern is nonzero (true), awk runs the resulting action for the line.

Actions

An action is a series of statements ended by semicolons, newlines, or closing braces. A condition is any expression; awk considers a nonzero value true, and a zero value false. A statement is one of the following or any series of statements enclosed in braces:

```
# expression statement, e.g. assignment
expression

# if statement
if (condition)
    statement
[else
    statement]
```
while loop
while (condition)
  statement

# do-while loop
do
  statement
while (condition)

# for loop
for (expression1; condition; expression2)
  statement

The for statement is equivalent to:

evaluation
while (condition) {
  statement
  expression2
}

The for statement can also have the form:

for (i in array)
  statement

awk runs the statement (specified with the statement argument) once for each element in array; on each repetition, the variable i contains the name of a subscript of array, running through all the subscripts in an arbitrary order. If array is multidimensional (has multiple subscripts), i is expressed as a single string with the SUBSEP character separating the subscripts.

• The statement break exits a for or a while loop immediately. continue stops the current iteration of a for or while loop and begins the next iteration (if there is one).
• next ends any processing for the current input record and immediately starts processing the next input record. Processing for the next record begins with the first appropriate rule. If a next statement appears or is invoked in a BEGIN or END action, awk will cause all further BEGIN or END action processing to be abandoned.
• exit[ (expr) ] immediately goes to the END action if it exists; if there is no END action, or if awk is already running the END action, the awk program ends. awk sets the exit status of the program to the numeric value of expr. If you omit (expr), the exit status is 0. return [expr] returns from the execution of a function.

If you specify an expr, the function returns the value of the expression as its result; otherwise, the function result is undefined. delete array[i] deletes element i from the given array. printf fmt, expr, expr, ... is also described in “Output” on page 44.

Output
The printf statement prints its arguments with only simple formatting. If it has no arguments, it prints the entire current input record. awk adds the output record separator ORS to the end of the output that each printf statement produces; when commas separate arguments in the printf statement, the output field separator OFS separates the corresponding output values. ORS and OFS are built-in variables, whose values you can change by assigning them strings. The default output record separator is a newline, and the default output field separator is a space.

The variable OFMT gives the format of floating-point numbers output by printf. By default, the value is %.6g; you can change this by assigning OFMT a different string value. OFMT applies only to floating-point numbers (ones with fractional parts).
The `printf` statement formats its arguments using the `fmt` argument. Formatting is the same as for the built-in function `sprintf`. Unlike `printf`, `printf` does not add output separators automatically. This gives the program more precise control of the output.

The `printf` and `printf` statements write to stdout. You can redirect output to a file or pipe.

If you add `>expr` to a `printf` or `printf` statement, awk treats the string value of `expr` as a file name, and writes output to that file. Similarly, if you add `>>expr`, awk sends output to the current contents of the file. The distinction between `>` and `>>` is important only for the first print to the file `expr`. Subsequent outputs to an already open file append to what is there already.

You cannot use such ambiguous statements as:

```bash
print a > b c
```

Use parentheses to resolve the ambiguity.

If you add `|expr` to a `printf` or `printf` statement, awk treats the string value of `expr` as an executable command and runs it with the output from the statement piped as input into the command.

As a reminder, you can have only a limited number of files and pipes open at any time. To avoid going over the limit, use the close function to close files and pipes when you no longer need them.

`printf` and `printf` are also available as functions with the same calling sequence, but no redirection.

### Examples

1. The following example:

```bash
awk '{print NR ":" $0}' input1
```

outputs the contents of the file `input1` with line numbers that are prepended to each line.

2. The following is an example using `var=value` on the command line:

```bash
awk '{print NR SEP $0}' SEP=":" input1
```

`awk` can also read the program script from a file as in the command line:

```bash
awk -f addline.awk input1
```

which produces the same output when the file `addline.awk` contains:

```bash
{print NR ":" $0}
```

3. The following program appends all input lines that start with January to the file `jan` (which might or might not exist already), and all lines that start with February or March to the file `febmar`:

```bash
/^January/ {print >> "jan"}
/^February|^March/ {print >> "febmar"}
```

4. This program prints the total and average for the last column of each input line:

```bash
{ s += $NF }
END {print "sum is", s, "average is", s/NR}
```

5. The next program interchanges the first and second fields of input lines:

```bash
{ tmp = $1
  $1 = $2
  $2 = tmp
  print
}
```
6. The following inserts line numbers so that output lines are left-aligned:

```c
{printf "%-6d: %s\n", NR, $0}
```

7. The following prints input records in reverse order (assuming sufficient memory):

```c
{
    a[ NR ] = $0 # index using record number
}  END { 
    for (i = NR; i>0; --i)
        print a[i]
}
```

8. The following program determines the number of lines that start with the same first field:

```c
{
    ++a[ $1 ] # array indexed using the first field
}  END {  # note output will be in undefined order
    for (i in a)
        print a[i], "lines start with", i
}
```

You can use the following program to determine the number of lines in each input file:

```c
{
    ++a[ FILENAME ]
}  END { 
    for (file in a)
        if (a[file] == 1)
            print file, "has 1 line"
        else
            print file, "has", a[file], "lines"
}
```

9. The following program illustrates how you can use a two-dimensional array in awk. Assume that the first field of each input record contains a product number, the second field contains a month number, and the third field contains a quantity (bought, sold, or whatever). The program generates a table of products versus month.

```c
BEGIN   {NUMPROD = 5}
{
    array[ $1, $2 ] += $3
}  END   {  
    print "	 Jan	 Feb	 March	 April	 May	 June	 July	 Aug	 Sept	 Oct	 Nov	 Dec"
    for (prod = 1; prod <= NUMPROD; prod++) {
        printf "%7s", "prod#" prod
        for (month = 1; month <= 12; month++) {
            printf "%5d", array[ prod, month ]
        }
        printf "\n"
    }
}
```

10. As the following program reads in each line of input, it reports whether the line matches a predetermined value:

```c
function randint() {
    return (int((rand()+1)*10))
}
BEGIN   {
    prize[ randint(), randint() ] = "$100";
    prize[ randint(), randint() ] = "$10";
    prize[ 1, 1 ] = "the booby prize"
}{
    if ( ($1, $2) in prize)
        printf "You have won %s!\n", prize[ $1, $2 ]
```
11. The following example prints lines, the first and last fields of which are the same, reversing the order of the fields:

```
$1==$NF {
  for (i = NF; i > 0; --i)
    printf "\%s", $i (i>1 ? OFS : ORS)
}
```

12. The following program prints the input files from the command line. The infiles function first empties the passed array, and then fills the array. The extra parameter $i of infiles is a local variable.

```
function infiles(f,i) {
  for (i in f)
    delete f[i]
  for (i = 1; i < ARGC; i++)
    if (index(ARGV[i],"=") == 0)
      f[i] = ARGV[i]
}
BEGIN {
  infiles(a)
  for (i in a)
    print a[i]
  exit
}
```

13. Here is the standard recursive factorial function:

```
function fact(num) {
  if (num <= 1)
    return 1
  else
    return num * fact(num - 1)
}
{ print $0 " factorial is " fact($0) }
```

14. The following program illustrates the use of getline with a pipe. Here, getline sets the current record from the output of the wc command. The program prints the number of words in each input file.

```
function words(file, string) {
  string = "wc " fn
  string | getline
  close(string)
  return ($2)
}
BEGIN {
  for (i=1; i<ARGC; i++) {
    fn = ARGV[i]
    printf "There are %d words in %s.\n",
           words(fn), fn
  }
}
```

**Environment variables**

*awk* uses the following environment variables:

**PATH**

Contains a list of directories that *awk* searches when looking for commands run by `system(expr)`, or input and output pipes.

**_UNIX03**

For more information about the effect of _UNIX03 on the *awk* command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Any other environment variable can be accessed by the *awk* program itself.

**Localization**

*awk* uses the following localization environment variables:
• LANG
• LC_ALL
• LC_COLLATE
• LC_CTYPE
• LC_MESSAGES
• LC_NUMERIC
• LC_SYNTAX
• NLSPATH


Exit values

0
Successful completion
• If the awk program contains no actions and no patterns, but is otherwise a valid awk program, standard input and any file operands are not read and awk exits with an exit status of zero.

1
Any of the following errors:
• Parser internal stack overflow
• Syntax error
• Function redefined
• Internal execution tree error
• Insufficient memory for string storage
• Unbalanced parenthesis or brace
• Missing script file
• Missing field separator
• Missing variable assignment
• Unknown option
• Incorrect character in input
• Newline in regular expression
• Newline in string
• EOF in regular expression
• EOF in string
• Cannot open script file
• Inadmissible use of reserved keyword
• Attempt to redefine built-in function
• Cannot open input file
• Error on printf
• Insufficient arguments to printf or sprintf()
• Array cannot be used as a scalar
• Variable cannot be used as a function
• Too many fields
• Record too long
• Division (/ or %) by zero
• Syntax error
• Cannot assign to a function
• Value required in assignment
• Return outside of a function
• Can delete only array element or array
• Scalar cannot be used as array
• SYMTAB must have exactly one index
• Impossible function call
• Function call nesting level exceeded
• Wrong number of arguments to function
• Regular expression error
• Second parameter to “split” must be an array
• sprintf string longer than allowed number of characters
• No open file name
• Function requires an array
• Is not a function
• Failed to match
• Incorrect collation element
• Trailing \ in pattern
• Newline found before end of pattern
• More than 9 ( \( \) ) pairs
• Number in [0-9] incorrect
• [ ] imbalance or syntax error
• ( ) or \( \) imbalance
• { } or \{ \} imbalance
• Incorrect endpoint in range
• Out of memory
• Incorrect repetition
• Incorrect character class type
• Internal error
• Unknown regex error

When an awk program ends because of a call to exit(), the exit status is the value that is passed to exit().

Limits
Most constructions in this implementation of awk are dynamic, limited only by memory restrictions of the system.

The maximum record size is guaranteed to be at least LINE_MAX as returned by getconf. The maximum field size is guaranteed to be LINE_MAX, also.

The parser stack depth is limited to 150 levels. Attempting to process extremely complicated programs might result in an overflow of this stack, causing an error.

Input must be text files.
Portability
POSIX.2, X/Open Portability Guide UNIX systems.
The ord function is an extension to traditional implementations of awk. The toupper and tolower functions and the ENVIRON array are in POSIX and the UNIX System V Release 4 version of awk. This version is a superset of New awk, as described in The AWK Programming Language by Aho, Weinberger, and Kernighan.

The standard command interpreter that the system function uses and that awk uses to run pipelines for getline, print, and printf is system-dependent. On z/OS UNIX, this interpreter is always /bin/sh.

Related information
ed, egrep, sed, vi

For more information about regexp, see Appendix C, “Regular expressions (regexp),” on page 945.

basename - Return the nondirectory components of a path name

Format
basename name [suffix]

Description
basename strips off the leading part of a path name, leaving only the final component of the name, which is assumed to be the file name. To accomplish this, basename first checks to see if name consists of nothing but slash (/) characters. If so, basename replaces name with a single slash and the process is complete. If not, basename removes trailing slashes. If slashes still remain, basename strips off all leading characters up to and including the final slash. Finally, if you specify suffix and the remaining portion of name contains a suffix that matches suffix, basename removes that suffix.

Examples
The command:

basename src/dos/printf.c

produces:

printf.c

Localization
basename uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values
0
Successful completion.
Failure due to any of the following reasons:

- Unknown command-line option.
- Incorrect number of arguments.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

**Related information**

dirname

---

### batch - Run commands when the system is not busy

**Format**

batch

**Description**

batch lets you run commands in batch mode. It reads the commands from the standard input (stdin). The system records the commands and runs them at a time when the system load is relatively low (that is, when the system is not busy).

The batch command is equivalent to

```
    at -q b -m now
```

For more details, see at.

at, batch, and crontab submit jobs to cron; the data in those jobs may contain double-byte characters. When the jobs are run, the data in the jobs are interpreted in the locale that cron is using. Since it is strongly recommended that cron be started in the POSIX locale, double-byte characters in the job may not be interpreted correctly. You may be able to get around this by calling `setlocale()` in the job itself.

**Environment variables**

batch uses the following environment variable:

**SHELL**

Contains the name of the shell command interpreter used to invoke the batch job.

**Localization**

batch uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

Exit values

0
  Successful completion.

>0
  Returned if the command fails for any reason.

If an error occurs, batch does not schedule the job.

Portability

POSIX.2 User Portability Extension

Related information

at, bg, crontab

---

**bc - Use the arbitrary-precision arithmetic calculation language**

Format

```
bc [-i] [-l] [file]
```

Description

**bc** is a programming language that can perform arithmetic calculations to arbitrary precision. You can use it interactively, by entering instructions from the terminal. It can also run programs that are taken from files.

The *file* arguments that you specify on the command line should be text files containing **bc** instructions. **bc** runs the instructions from those files, in the order that they appear on the command line, and then runs instructions from the standard input. **bc** ends when it runs a `quit` instruction or reaches the end of the file on standard input.

**bc** is a simple but complete programming language with a syntax reminiscent of the C programming language. This version of **bc** is a superset of the standard language available on most systems. It has a number of additional features intended to make the language more flexible and useful. Features unique to this implementation are noted.

Input consists of a series of instructions that assign values to variables or make calculations. It is also possible to define subprograms called *functions*, which perform a sequence of instructions to calculate a single value.

**bc** displays the result of any line that calculates a value, but does not assign it to a variable. For example, the instruction:

```
2+2
```

displays:

```
4
```

By default, **bc** displays the result of any evaluated instruction followed by a newline. **bc** also saves the last value displayed in a special variable `. (dot)`, so that you can use it in subsequent calculations.

For a summary of the UNIX03 changes to this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Options

**bc** supports the following options.
-i

Puts `bc` into interactive mode with a displayed prompt. In this mode, `bc` displays a prompt, which is: (waiting for input). In addition, it handles errors differently. Typically, when `bc` encounters an error while processing a file, the interpreter displays the error message and exits. In interactive mode, the interpreter displays the message and returns to the prompt mode to allow debugging.

-l

Loads a library of standard mathematical functions before processing any other input. This library also sets the `scale` to 20. For a description of the functions in the -l library, see Built-in functions.

Numbers

Numbers consist of an optional minus (-) sign or an optional plus (+) sign followed by a sequence of zero or more digits, followed by an optional decimal point (.), followed by a sequence of zero or more digits. Valid digits are 0 through 9, and the hexadecimal digits A through F. The uppercase letters represent the values from 10 through 15. There must be at least one digit, either before or after the decimal point. If not, `bc` interprets the decimal point as the special variable .

A number can be arbitrarily long and can contain spaces. Here are some valid numbers with an input base of 10:

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.</td>
<td>.0</td>
</tr>
<tr>
<td>-3.14159</td>
<td>+09.</td>
</tr>
<tr>
<td>-12</td>
<td>1 000 000</td>
</tr>
</tbody>
</table>

Here are some valid numbers with an input base of 16 (`ibase`=16):

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FF</td>
</tr>
<tr>
<td>FF.3</td>
<td>-10.444</td>
</tr>
<tr>
<td>A1</td>
<td></td>
</tr>
</tbody>
</table>

See Bases for more information.

You cannot break up numbers with commas; you can write `1000000` or `1 000 000`, but `1,000,000` results in an error message.

Identifiers

Identifiers can include sequences containing any number of letters, digits, or the underscore (_) character but must start with a lowercase letter. Spaces are not allowed in identifiers.

In the POSIX locale, valid identifiers can include sequences containing any number of letters, digits, or the underscore (_) character but must start with a lowercase letter, as defined by the current locale.

For other locales, the character map for that locale determines which characters are valid in an identifier. If you want identifiers to be portable between locales, use characters from the POSIX character set. The use of identifiers longer than one character is an extension of this implementation. Identifiers are used as names for variables, functions, or arrays:

- A **variable** holds a single numeric value. You can declare variables that are local to a function using the `auto` statement. (See Functions.) All other variables are global and you can use them inside any function or outside all functions. You do not need to declare global variables. `bc` creates variables as it requires them, with an initial value of zero. (Remember that there is also the special variable . [dot], which contains the result of the last calculation.)

- A **function** is a sequence of instructions that calculates a single value. A list of zero or more values that are enclosed in parentheses always follow a function name, as in my_func (3.14159). (See Functions.)

- An **array** is a list of values. Values in the list are called elements of the array. These elements are numbered, beginning at zero. Such a number is called a subscript, or index, of the array. Subscripts always appear in square brackets after the array. For example, `a[0]` refers to element zero in the array `a`. The first element of the array always has the subscript 0. If a subscript value is a floating-point number, the fractional part is discarded to make the subscript into an integer. For example, the following expressions all refer to the same element:

```
```
The maximum number of elements in a **bc** array is in the range from 0 to {BC_DIM_MAX}-1 inclusive. Unlike with many languages, you do not need to declare the size of an array. Elements are created dynamically as required, with an initial value of zero.

Since parentheses always follow function names and square brackets always follow array names, **bc** can distinguish between all three types of names: variable names, function names, and array names. Therefore, you can have variables, functions, and arrays with the same name. For example, `foo` might be a variable whereas `foo()` is a function and `foo[ ]` is an array.

### Built-in variables

**bc** has a number of built-in variables that are used to control various aspects of the interpreter. These are described in the following topics.

#### Scale

The scale value is the number of digits to be retained after the decimal point in arithmetic operations. For example, if the scale is 3, each calculation retains at least three digits after the decimal point. This means that:

```
5 / 3
```

has the value:

```
1.666
```

If `-l` is specified, the scale is set to 20; otherwise, the default scale is zero.

The variable `scale` holds the current scale value. To change scales, assign a new value to `scale`, as in:

```
scale = 5
```

Since `scale` is just a regular **bc** variable, it can be used in the full range of **bc** expressions.

The number of decimal places in the result of a calculation is affected not only by the scale, but also by the number of decimal places in the operands of the calculation. **Arithmetic operations** discusses this.

There is also a function `scale`, which can determine the scale of any expression. For example, `scale(1.1234)` returns the result 4, which is the scale of the number 1.1234. The result of the `scale` function is always an integer (that is, it has the scale of 0).

The maximum value for `scale` is given by the configuration variable `{BC_SCALE_MAX}` and the minimum value is 0.

#### Bases

With **bc**, you can specify numbers in different bases. For example, octal (base 8) or hexadecimal (base 16). You can input numbers in one base and output them in a different base, simplifying the job of converting from one base to another. **bc** does this using the built-in variables `ibase` and `obase`.

`ibase` is the base for input numbers. It has an initial value of 10 (normal decimal numbers). To use a different base for inputting numbers, assign an integer to `ibase`, as in:

```
ibase = 8
```

This means that all future input numbers are to be in base 8 (octal). The largest valid input base is 16, and the smallest valid input base is 2. There is no mechanism provided to represent digits larger than 15, so bases larger than 16 are essentially useless. When the base is greater than 10, use the uppercase letters as digits. For example, base 16 uses the digits 0 through 9, and A through F. The digits are allowed in any number, regardless of the setting of `ibase` but are largely meaningless if the base is smaller than the digit.
The one case where this is useful is in resetting the input base to 10. The constant A always has the value 10 no matter what ibase is set to, so to reset the input base to 10, type:

```
ibase = A
```

`obase` is the base in which numbers are output. It has an initial value of 10 (normal decimal numbers). To change output bases, assign an appropriate integer to `obase`.

If the output base is 16 or less, `bc` displays numbers with normal digits and hexadecimal digits (if needed). The output base can also be greater than 16, in which case each digit is printed as a decimal value and digits are separated by a single space. For example, if `obase` is 1000, the decimal number 123 456 789 is printed as:

```
123 456 789
```

Here, the digits are decimal values from 0 through 999. As a result, all output values are broken up into one or more chunks with three digits per chunk. Using output bases that are large powers of 10, you can arrange your output in columns; for example, many users find that 100 000 makes a good output base, because numbers are grouped into chunks of five digits each.

Long numbers are output with a maximum of 70 characters per line. If a number is longer than this, `bc` puts a backslash (\) at the end of the line indicating that the number is continued on the next line. The backslash (\) and newline characters are counted as part of the 70 character length.

Internal calculations are performed in decimal, regardless of the input and output bases. Therefore, the number of places after the decimal point are dictated by the scale when numbers are expressed in decimal form.

The maximum value for `obase` is given by the configuration variable `{BC_BASE_MAX}`.

### Arithmetic operations

`bc` provides many arithmetic operations. Following standard arithmetic conventions, some operations are calculated before others. For example, multiplications take place before additions unless you use parentheses to group operations. Operations that take place first are said to have a higher precedence than operations that take place later.

Operations also have an associativity. The associativity dictates the order of evaluation when you have a sequence of operations with equal precedence. Some operations are evaluated left to right, whereas others are evaluated right to left. The following list shows the operators of `bc` from highest precedence to lowest.

<table>
<thead>
<tr>
<th><code>bc</code> operator</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
<td>Left to right.</td>
</tr>
<tr>
<td>Unary ++ --</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Unary - !</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>^</td>
<td>Right to left.</td>
</tr>
<tr>
<td>* / %</td>
<td>Left to right.</td>
</tr>
<tr>
<td>+ -</td>
<td>Left to right.</td>
</tr>
<tr>
<td>= ^= *=/ %= +=</td>
<td>Right to left.</td>
</tr>
<tr>
<td>== &lt;= &gt;= != &lt; &gt;</td>
<td>None.</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Left to right.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The order of precedence in bc is not the same as the one in C. In C, the assignment operators have the lowest precedence.

The following list describes what each operation does. In the descriptions, A and B can be numbers, variables, array elements, or other expressions. V must be either a variable or an array element.

(A) Indicates that this expression -A should be evaluated before any other operations are performed on it.

-A Is the negation of the expression.

!A Is the logical complement of the expression. If A evaluates to zero, !A evaluates to 1. If A is not zero, !A evaluates to zero. This operator is unique to this version of bc.

++V Adds 1 to the value of V. The result of the expression is the new value of V.

- -V Subtracts 1 from the value of V. The result of the expression is the new value of V.

V++ Adds 1 to the value of V, but the result of the expression is the old value of V.

V- - Subtracts 1 from the value of V, but the result of the expression is the old value of V.

A ^ B Calculates A to the power B. B must be an integer. The scale of the result of A^B is:

\[
\min(\text{scale}(A) \times \text{abs}(B), \max(\text{scale}, \text{scale}(A)))
\]

where \(\min\) calculates the minimum of a set of numbers and \(\max\) calculates the maximum.

A * B Calculates A multiplied by B. The scale of the result is:

\[
\min(\text{scale}(A) + \text{scale}(B), \max(\text{scale}, \text{scale}(A), \text{scale}(B)))
\]

A / B Calculates A divided by B. The scale of the result is the value of scale.

A % B Calculates the remainder from the division of A by B. This is calculated in two steps. First, bc calculates A/B to the current scale. It then obtains the remainder through the formula:

\[
A - (A / B) \times B
\]

calculated to the scale:

\[
\max(\text{scale} + \text{scale}(B), \text{scale}(A))
\]

A + B Adds A plus B. The scale of the result is the maximum of the two scales of the operands.

A-B Calculates A minus B. The scale of the result is the maximum of the two scales of the operands.

Therefore, you can write such operations as \(a=1+(b=2)\). In this operation, the value of the assignment in parentheses is 2 because that is the value assigned to b. Therefore, the value 3 is assigned to a. The possible assignment operators are:

V = B Assigns the value of B to V.

V ^= B Is equivalent to \(V=V^B\).
\$v \ast= b\ \\
\text{Is equivalent to } v=v\ast b.\\n\v/\ast b\ \\
\text{Is equivalent to } v=v/b.\\n\v \%= b\ \\
\text{Is equivalent to } v=v\% b.\\n\v += b\ \\
\text{Is equivalent to } v=v+b.\\n\v -= b\ \\
\text{Is equivalent to } v=v-b.\\n
The next group of operators are all assignment operators. They assign values to objects. An assignment operation has a value, which is the value that is being assigned.

The following expressions are called relations, and their values can be either true (1) or false (0). This version of bc lets you use the relational operators in any expression, not just in the conditional parts of \textit{if}, \textit{while}, or \textit{for} statements. These operators work exactly like their equivalents in the C language. The result of a relation is 0 if the relation is false and 1 if the relation is true.

\$a == b\ \\
\text{Is true if and only if } a \text{ equals } b.\\n\a <= b\ \\
\text{Is true if and only if } a \text{ is less than or equal to } b.\\n\a >= b\ \\
\text{Is true if and only if } a \text{ is greater than or equal to } b.\\n\a != b\ \\
\text{Is true if and only if } a \text{ is not equal to } b.\\n\a < b\ \\
\text{Is true if and only if } a \text{ is less than } b.\\n\a > b\ \\
\text{Is true if and only if } a \text{ is greater than } b.\\n\a && b\ \\
\text{Is true if and only if } a \text{ is true (nonzero) and } b \text{ is true. If } a \text{ is not true, the expression } b \text{ is never evaluated.}\\n\a || b\ \\
\text{Is true if } a \text{ is true or } b \text{ is true. If } a \text{ is true, the expression } b \text{ is never evaluated.}

\textbf{Comments and white space}

A \text{ comment} has the form:

\begin{verbatim}
/* Any string */
\end{verbatim}

Comments can extend over more than one line of text. When bc sees /* at the start of a comment, it discards everything up to the next */. The only effect a comment has is to indicate the end of a token. As an extension, this version of bc also provides an additional comment convention using the # character. All text from the # to the end of the line is treated as a single blank, as in:

\begin{verbatim}
2+2 # this is a comment
\end{verbatim}

bc is free format. You can freely insert blanks or horizontal tab characters to improve the readability of the code. Instructions are assumed to end at the end of the line. If you have an instruction that is so long you need to continue it on a new line, put a backslash (\) as the very last character of the first line and continue on the second, as in:

\begin{verbatim}
a = 2\
++3
\end{verbatim}
The \ indicates that the instruction continues on the next line, so this is equivalent to:

```
a = 2 + 3
```

**Instructions**

A `bc` instruction can be an expression that performs a calculation, an assignment, a function definition, or a statement. If an instruction is not an assignment, `bc` displays the result of the instruction when it has completed the calculation. For example, if you enter:

```
3.14 * 23
```

`bc` displays the result of the calculation. However, with:

```
a = 3.14 * 23
```

`bc` does not display anything, because the expression is an assignment. If you do want to display the value of an assignment expression, simply place the expression in parentheses.

The following list shows the instruction forms recognized by `bc`:

**expression**

Calculates the value of the *expression*.

**"string"**

Is a string constant. When `bc` sees a statement of this form, it displays the contents of the string. For example:

```
"Hello world!"
```

tells `bc` to display `Hello world!`. A newline character is not output after the string. This makes it possible to do things like:

```
foo = 15
"The value of foo is "; foo
```

With these instructions, `bc` displays

```
The value of foo is 15
```

**statement ; statement ...**

Is a sequence of statements on the same line. In `bc`, a semicolon (;) and a newline are equivalent. They both indicate the end of a statement. `bc` runs these statements in order from left to right.

**{statement}**

Is a brace-bracketed statement. Brace brackets are used to group sequences of statements together, as in:

```
{
  statement
  statement
  ...
}
```

Brace brackets can group a series of statements that are split over several lines. Braces are typically used with control statements like if and while.

**break**

Can be used only inside a while or for loop. `break` ends the loop.

**for (initexp ; relation ; endexp) statement**

Is equivalent to:

```
initexp
while (relation) {
  statement
}```
endexp
}

where initexp and endexp are expressions and relation is a relation. For example:

```plaintext
a = 0
for (i = 1; i <= 10; ++i) a += i
```

is equivalent to the while example given earlier. All three items inside the parentheses must be specified. Unlike C, bc does not let you omit any of these expressions.

**if (relation) statement**
Tests whether the given relation is true. If so, bc runs the statement; otherwise, bc skips over the statement and goes to the next instruction. For example:

```plaintext
if ((a%2) == 0) "a is even"
```

displays a is even if a has an even value.

**if (relation) statement1 else statement2**
Is similar to the simple if statement. It runs statement1 if relation is true and otherwise runs statement2. It may be used as follows:

```plaintext
if ((a%2) == 0) "a is even" else "a is odd"
```

There is no statement separator between "a is even" and the else keyword. This differs from the C language.

Here is another example:

```plaintext
if (a<10) {
  "a is "; "less than 10 "
} else {
  "a is "
  "greater than 10 "
}
```

The braces must be on the same line as the if and the else keywords. This is because a new line or a semicolon right after (relation) indicates that the body of the statement is null. One common source of errors in bc programs is typing the statement body portion of an if statement on a separate line. If -i is used, the interpreter displays a warning when if statements with null bodies are encountered.

**while (relation) statement**
Repeatedly runs the given statement while relation is true. For example:

```plaintext
i = 1
a = 0
while (i <= 10) {    a += i
  ++i
}
```

adds the integers from 1 through 10 and stores the result in a.

If relation is not true when bc encounters the while loop, bc does not run statement at all.

**print expression , expression ...**
Displays the results of the argument expressions. Normally, bc displays the value of each expression or string it encounters. This makes it difficult to format your output in programs. For this reason, the z/OS shell version of bc has a print statement to give you more control over how things are displayed. print lets you display several numbers on the same line with strings. This statement displays all its arguments on a single line. A single space is displayed between adjacent numbers (but not between numbers and strings). A print statement with no arguments displays a newline. If the
last argument is null, subsequent output continues on the same line. Here are some examples of how
to use print:

```c
/* basic print statement */
print "The square of ", 2, "is ", 2*2
The square of 2 is 4

/* inserts a space between adjacent numbers */
print 1,2,3
1 2 3

/* note - no spaces */
print 1,"",2,"",3
123

/* just print a blank line */
print

/* two statements with output on same line */
print 1,2,3, ; print 4, 5, 6
1 2 3 4 5 6
```

**quit**

Ends bc. In other implementations of bc, the interpreter exits as soon as it reads this token. This
version of bc treats quit as a real statement, so you can use it in loops, functions, and so on.

**sh ...**

Lets you send a line to the system command interpreter for execution, as in:

```
sh more <foo
```

This command passes everything from the first nonblank character until the end of the line to the
command interpreter for execution.

**void expression**

Throws away, or “voids,” the result of the evaluation of expression instead of displaying it. This
instruction is useful when using ++ and -- operators, or when you want to use a function but don't
want to use the return value for anything. For example:

```
void foo++
```

increments foo but does not display the result. The void statement is unique to this version of bc.

Several other types of statements are relevant only in function definitions. These are described in the next
topic.

**Functions**

A function is a subprogram to calculate a result based on argument values. For example, the following
function converts a temperature given in Fahrenheit into the equivalent temperature in Celsius:

```c
define f_to_c(f) {
    return ((f-32) * 5 / 9)
}
```

This defines a function named `f_to_c` that takes a single argument called `f`. The body of the function is
enclosed in brace brackets. The opening brace must be on the same line as the define keyword. The
function body consists of a sequence of statements to calculate the result of the function. An expression
of the form:

```c
return (expression)
```
returns the value of expression as the result of the function. The parentheses around the expression are optional.

To activate the subprogram you use a function call. This has the form:

```
name(expression, expression,...)
```

where name is the name of the function, and the expressions are argument values for the function. You can use function call anywhere you might use any other expression. The value of the function call is the value that the function returns. For example, with the function `f_to_c()`, `f_to_c(41)` has the value 5 (since 41 Fahrenheit is equivalent to 5 Celsius).

The general form of a function definition is:

```
define name(parameter, parameter,...) {
    auto local, local, ...
    statement
    statement
    ...
}
```

Each parameter on the first line can be a variable name or an array name. Array names are indicated by putting square brackets after them. For example, if `cmpvec` is a function that compares two vectors, the function definition might start with:

```
define cmpvec(a[], b[]) {
```

Parameters do not conflict with arrays or variables of the same name. For example, you can have a parameter named `a` inside a function, and a variable named `a` outside, and the two are considered entirely separate entities. Assigning a value to the variable does not change the parameter and vice versa. All parameters are passed by value. This means that a copy is made of the argument value and is assigned to the formal parameter. This also applies to arrays. If you pass an array to a function, a copy is made of the whole array, so any changes made to the array parameter do not affect the original array.

A function may not need any arguments. In this case, the `define` line does not have any parameters inside the parentheses, as in:

```
define f() {
```

The auto statement declares a sequence of local variables. When a variable or array name appears in an auto statement, the current values of those items are saved and the items are initialized to zero. For the duration of the function, the items have their new values. When the function ends, the old values of the items are restored.

However, `bc` uses dynamic scoping rules, unlike C which uses lexical scoping rules. “Usage notes” on page 64 for more information.

For example:

```
define addarr(a[], l) {
    auto i, s
    for (i=0; i < l; ++i) s += a[i]
    return (s)
}
```

is a function that adds the elements in an array. The argument `l` stands for the number of elements in the array. The function uses two local names: a variable named `i` and a variable named `s`. These variables are “local” to the function `addarr` and are unrelated to objects of the same name outside the function (or in other functions). Objects that are named in an auto statement are called autos. Autos are initialized to 0 each time the function is called. Thus, the sum `s` is set to zero each time this function is called. You can also have local arrays, which are specified by placing square brackets after the array name in the auto statement.

```
define func_with_local_array() {
    auto local_array[];
```
This example defines a local array called `local_array`. Local arrays start out with no elements in them.

If a function refers to an object that is not a parameter and not declared `auto`, the object is assumed to be `external`. External objects may be referred to by other functions or by statements that are outside of functions. For example:

```c
define sum_c(a[], b[], l) {
    auto i
    for (i=0; i < l; ++i) c[i] = a[i] + b[i]
}
```

refers to an external array named `c`, which is the element-by-element sum of two other arrays. If `c` did not exist prior to calling `sum_c`, it is created dynamically. After the program has called `sum_c`, statements in the program or in functions can refer to array `c`.

Functions typically require a `return` statement. This has the form:

```c
return (expression)
```

The argument `expression` is evaluated and used as the result of the function. The expression must have a single numeric value; it cannot be an array.

A `return` statement ends a function, even if there are more statements left in the function. For example:

```c
define abs(i) {
    if (i < 0) return (-i)
    return (i)
}
```

is a function that returns the absolute value of its argument. If `i` is less than zero, the function takes the first `return`; otherwise, it takes the second.

A function can also end by running the last statement in the function. If so, the result of the function is zero. The function `sum_c` is an example of a function that does not have a `return` statement. The function does not need a `return` statement, because its work is to calculate the external array `c`, not to calculate a single value. Finally, if you want to return from a function, but not return a value you can use `return()` or simply `return`. If there are no parameters to the `return` statement, a default value of zero is returned.

### Built-in functions

`bc` has a number of built-in functions that perform various operations. These functions are similar to user-defined functions. You do not have to define them yourself, however; they are already set up for you.

These functions are:

- `length(expression)`
  Calculates the total number of decimal digits in `expression`. This includes digits both before and after the decimal point. The result of `length()` is an integer. For example, `length(123.456)` returns 6.

- `scale(expression)`
  Returns the scale of `expression`. For example, `scale(123.456)` returns 3. The result of `scale()` is always an integer. Subtracting the scale of a number from the length of a number lets you determine the number of digits before the decimal point.

- `sqrt(expression)`
  Calculates the square root of the value of `expression`. The result is truncated in the least significant decimal place (not rounded). The scale of the result is the scale of `expression`, or the value of `scale()`, whichever is larger.

You can use the following functions if `-l` is specified on the command line. If it is not, the function names are not recognized. There are two names for each function: a full name, and a single character name for
compatibility with POSIX.2POSIX.2. The full names are the same as the equivalent functions in the standard C math library.

**arctan(expression) or a(expression)**
Calculates the arctangent of expression, returning an angle in radians. This function can also be called as atan(expression).

**bessel(integer,expression) or j(integer,expression)**
Calculates the Bessel function of expression, with order integer. This function can also be called as jn(integer,expression).

**cos(expression) or c(expression)**
Calculates the cosine of expression, where expression is an angle in radians.

**exp(expression) or e(expression)**
Calculates the exponential of expression (that is, the value e to the power of expression).

**ln(expression) or l(expression)**
Calculates the natural logarithm of expression. This function can also be called as log(expression).

**sin(expression) or s(expression)**
Calculates the sine of expression, where expression is an angle in radians.

The scale value of the result returned by these functions is the value of the scale variable at the time the function is invoked. The value of the scale variable after these functions have completed their execution will be the same value it had upon invocation.

**Examples**

1. Here is a simple function to calculate the sales tax on a purchase. The amount of the purchase is given by purchase, and the amount of the sales tax (in per cent) is given by tax.

   ```c
   define sales_tax(purchase,tax) {
   auto old_scale
   scale = 2
   tax = purchase*(tax/100)
   scale = old_scale
   return (tax)
   }
   ```

   For example:

   ```c
   sales_tax(23.99,6)
   ```

   calculates 6% tax on a purchase of $23.99. The function temporarily sets the scale value to 2 so that the monetary figures have two figures after the decimal point. Remember that bc truncates calculations instead of rounding, so some accuracy may be lost. It is better to use one more digit than needed and perform the rounding at the end. The round2 function, shown later in this topic, rounds a number to two decimal places.

2. Division resets the scale of a number to the value of scale. You can use this to extract the integer portion of a number, as follows:

   ```c
   define integer_part(x) {
   # a local to save the value of scale
   auto old_scale
   # save the old scale, and set scale to 0
   old_scale = scale; scale=0
   # divide by 1 to truncate the number
   x /= 1
   # restore the old scale
   scale=old_scale
   return (x)
   }
   ```

3. Here is a function you can define to return the fractional part of a number:

   ```c
   define fractional_part(x) {return (x - integer_part(x))}
   ```
4. The following function lets you set the scale of number to a given number of decimal places:

```bc
define set_scale(x, s) {
    auto os
    os = scale
    scale = s
    x /= 1
    scale = os
    return (x)
}
```

You can now use `set_scale()` in a function that rounds a number to two decimal places:

```bc
define round2(num) {
    auto temp;
    if(scale(num) < 2) return (set_scale(num, 2))
    temp = (num - set_scale(num, 2)) * 1000
    if(temp > 5) num += 0.01
    return (set_scale(num, 2))
}
```

This function is very useful if you want to work with monetary values. For example, you can now rewrite `sales_tax()` to use `round2()`:

```bc
define sales_tax(purchase, tax) {
    auto old_scale
    scale = 2
    tax = round2(purchase*(tax/100))
    scale = old_scale
    return (tax)
}
```

5. This function recursively calculates the factorial of its argument:

```bc
define fact(x) {
    if(x < 1) return 1
    return (x*fact(x-1))
}
```

You can also write the factorial function iteratively:

```bc
define fact(x) {
    auto result
    result = 1
    while(x>1) result *= x--
    return (result)
}
```

With either version, `fact(6)` returns 720.

6. Here is another recursive function, that calculates the \( n \)th element of the Fibonacci sequence:

```bc
define fib(n) {
    if(n < 3) { return (1)
    } else {
        return (fib(n-1)+fib(n-2))
    }
}
```

**Usage notes**

1. Unlike the C language, which uses lexical scoping rules, **bc** uses dynamic scoping, which is most easily explained with an example:

```bc
a=10
define f1() {}
    auto a;
    a = 13;
    return (f2())
}
define f2() {
    return (a)
}
```
If \( f1() \) is called, \( \text{bc} \) prints the number 13, instead of the number 10. This is because \( f1() \) hides away the old (global) value of \( a \) and then sets it to 13. When \( f2() \) refers to \( a \), it sees the variable dynamically created by \( f1() \) and so prints 13. When \( f1() \) returns, it restores the old value of \( a \). When \( f2() \) is called directly, instead of through \( f1() \), it sees the global value for \( a \) and prints 10. The corresponding C code prints 10 in both cases.

2. Numbers are stored as strings in the program and converted into numbers each time they are used. This is important because the value of a “constant” number may change depending on the setting of the \( ibase \) variable. For example, suppose that the following instructions are given to \( \text{bc} \):

```plaintext
define ten() {
    return (10)
}
ten()
10
ibase=16
ten()
16
```

In this example, when the base is set to 10, \( \text{ten()} \) returns the decimal value 10. However, when the input base is changed to 16, the function returns the decimal value 16. This can be a source of confusing errors in \( \text{bc} \) programs.

3. The library of functions loaded using the \(-l\) option is stored in the file \(/usr/lib/lib.b\) under your root directory. This is a simple text file that you can examine and change to add new functions as desired.

4. In a noninteractive invocation, \( \text{bc} \) exits on any invalid input and the rest of the input are skipped.

**Files**

\( \text{bc} \) uses the following file:

\(/usr/lib/lib.b\)

File containing the library of functions loaded with \(-l\)

**Localization**

\( \text{bc} \) uses the following localization environment variables:

- \( \text{LANG} \)
- \( \text{LC_ALL} \)
- \( \text{LC_CTYPE} \)
- \( \text{LC_MESSAGES} \)
- \( \text{LC_SYNTAX} \)
- \( \text{NLSPATH} \)


**Exit values**

0

Successful completion.

1

Failure due to any of the following errors:

- Break statement found outside loop.
- Parser stack overflow.
• Syntax error.
• End of file in comment.
• End of file in string.
• Numerical constant is too long.
• String is too long.
• Empty evaluation stack.
• Cannot pass scalar to array.
• Cannot pass array to scalar.
• Incorrect array index.
• Built-in variable cannot be used as a parameter or auto variable.
• name is not a function.
• Incorrect value for built-in variable.
• Shell command failed to run.
• Division by 0.
• Incorrect value for exponentiation operator.
• Attempt to take square root of negative number.
• Out of memory.

2
Unknown command-line option.

Limits
The parser stack depth is limited to 150 levels. Attempting to process extremely complicated programs may result in an overflow of this stack, causing an error.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The following are extensions to the POSIX standard:
• The -i option.
• The && and || operators.
• The if ... else ... statement.
• Identifiers of more than one character or containing characters outside the POSIX character set.
• The print statement.
• The sh statement.
• The optional parentheses in the return statement.

In a double-byte environment, remember that only numbers and operators from the POSIX character set can be used. Identifiers can use characters from the current locale; if you want scripts to be portable, use only characters from the POSIX character set.

bg - Move a job to the background

Format
bg [job...]
tcsh shell: bg [%job ...]
**Description**

`bg` runs one or more jobs in the background. The job IDs given on the command line identify these jobs, which should all be ones that are currently stopped. If you do not specify any job IDs, `bg` uses the most recently stopped job. It works only if job control is enabled; see the `–m` option of `set` for more information. Job control is enabled by default in the z/OS shell.

In the tcsh shell:

- `bg` puts the specified jobs (or, without arguments, the current job) into the background, continuing each if it is stopped. `job` can be a number, a string, "", %, +, or -.
- `%job &` is a synonym of the `bg` command.

**Localization**

`bg` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Usage notes**

`bg` is a built-in shell command.

**Exit values**

0  
Successful completion.

>0  
Failure because a `job` argument is incorrect or there is no current job.

If an error occurs, `bg` exits and does not place the job in the background.

**Portability**

POSIX.2 User Portability Extension, UNIX systems.

**Related information**

at, batch, fg, jobs, set, tcsh

---

**bpixmaptext - Display reason code text**

**Format**

`bpixmaptext [-d] reason_code | error_number`

**Description**
bpuxmtext displays the description and action text for a reason code returned from z/OS UNIX System Services, TCP/IP, zFS, TFS, and the C/C++ Runtime Library. It also returns information about error numbers. It can be run from the z/OS UNIX shell and the TSO environment. You can also enter that command through System REXX as an MVS system command.

This command is intended as an aid for problem determination. Reason codes such as those returned by HFS or NFS are not supported by this command. When an error_number is specified, the description for that error number is displayed.

reason_code is specified as 7-8 hexadecimal characters.
error_number is specified as 1-4 hexadecimal characters.

Options
-d
  reason_code or error_number is provided as a decimal number.

Usage notes
1. If no text is available for the reason code, a blank line is displayed.
2. An argument that is not 1-8 hexadecimal digits in length results in a usage message. The message is not translated.

Examples
1. The command:
   bpxmtext 058800B0
   produces data displayed in the following format:
   BPXFSUMT 08/18/98
   JRUserNotPrivileged: The requester of the service is not privileged
   Action: The service requested required a privileged user.
   Check the documentation for the service to understand what privilege is required.

2. To find information for reason code 058800B0, issue:
   F AXR,BPXTEXT 058800B0

Exit values
0
  Successful completion.
2
  Failure due to an argument that is not 1-8 hexadecimal digits.

bpuxtrace - Activate or deactivate traces for processes

68 z/OS: UNIX System Services Command Reference
Format

- `bpxtrace -a [-B debug] [-p pid] [-u userid]
- `bpxtrace -c
  [-f format|full|short|counts]
  [-o outputpath] [-T tempdir] [-v volser]
  [-B debug] [-h columns] [-x] [command]
- `bpxtrace -e [-p pid] [-u userid]
  [-f format|full|short|counts]
  [-o outputpath] [-T tempdir] [-v volser]
  [-B debug] [-h columns] [-x]
- `bpxtrace -i [-p pid] [-u userid]
  [-f format|full|short|counts]
  [-o outputpath] [-T tempdir] [-v volser]
  [-B debug] [-h columns] [-x]
- `bpxtrace -s [-B debug] [-p pid] [-u userid]
- `bpxtrace -S
  [-o outputpath] [-T tempdir]
  [-B debug] [-v volser]
- `bpxtrace -t <seconds> [-p pid] [-u userid]
  [-f format|full|short|counts]
  [-o outputpath] [-T tempdir] [-v volser]
  [-B debug] [-h columns] [-x]

Description

`bpxtrace` activates and deactivates tracing for one or more processes. It can be run from the z/OS UNIX shell or the TSO environment. The captured syscall trace output is dependent on the system-wide setting for SYSOMVS CTRACE options and the amount of z/OS UNIX activity on the system when the `bpxtrace` command is issued.

If you are using SYSOMVS CTRACE, always specify the minimal setting. Otherwise, when you run `bpxtrace` to capture syscall data, the amount of the output might be reduced.

A single user can run `bpxtrace` only once at any given time. Running multiple occurrences might result in allocation errors, dump failures, inconsistent process tracing, or other unexpected behaviors.

The `bpxtrace` command might allocate the following temporary data sets:

- `userid.BPXGDCORE.DUMPDS`
- `userid.BPXGDCORE.LOG`
- `userid.BPXGDCORE.BPXEXEC`
- `userid.BPXGDCORE/DDIR`

The `userid` prefix is the MVS user name of the user running the command and is not affected by the current TSO prefix setting. The caller must have authority to create these data sets or the `bpxtrace` command will fail.

The data set names must be eligible for allocation on the specified or default volumes, or the `bpxtrace` command will fail. For example, any ACS routines that determine the SMS classes and storage groups for these data sets and objects might need to be modified to allow allocation on these volumes.
Options

While all options and parameters are accepted for each function, some might be ignored for a particular function. See the function descriptions in “Format” on page 69 for options that are applicable to that particular function.

-a
   Stops tracing without producing trace output.

-B debug
   Used when diagnosing problems running **bpxtrace**.

   This option is intended for use by service personnel only.

   The debug level is specified as one of the following numbers:

1
   Verbose mode

2
   Verbose mode and traps REXX signals

3
   Verbose mode, traps REXX signals, and retains the temporary files and data sets

4
   Verbose mode, traps REXX signals, retains the temporary files and data sets, and does not disable tracing for the **bpxtrace** command itself

-c command
   Traces the shell command or script that is specified by the **command** string invoked by the `/bin/sh -c command`.

-e
   Ends the tracing of user processes and produces trace output.

-f format|full|short|counts
   Specifies how the trace records (syscall entry and syscall exit records) should be displayed. The values `full`, `short`, and `counts` show the trace records as formatted by the IPCS CTRACE command with the corresponding IPCS format request for COMP(SYSOMVS).

   - **format** processes the trace records and displays one line per trace record that is formatted with relevant information from the trace record. The format value is also filtered based on the user ID or process ID value that was specified. The following screen shows an example of output from bpxtrace -f format:

<table>
<thead>
<tr>
<th>PID</th>
<th>ASID</th>
<th>TCB</th>
<th>Local time</th>
<th>System call</th>
<th>Additional trace data</th>
<th>Additional trace data</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0025</td>
<td>8FF1D8</td>
<td>09:43:04.070651 Call open</td>
<td>pgm=/bin/bpxtrace parms: 0000000D /bin/bpxwrtso 00000002</td>
<td>00000000 00000000 00000000 00000000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0025</td>
<td>8FF1D8</td>
<td>09:43:04.070668 Exit open</td>
<td>rv=00000007 pgm=/bin/bpxtrace parms: 00000007 25000000</td>
<td>00000000 00000000 00000000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0025</td>
<td>8FF1D8</td>
<td>09:43:04.070675 Call read</td>
<td>pgm=/bin/bpxtrace parms: 00000007 25000000</td>
<td>00000000 00000000 00000000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0025</td>
<td>8FF1D8</td>
<td>09:43:04.091468 Exit read</td>
<td>rv=00001000 pgm=/bin/bpxtrace parms: 00000007 25000000</td>
<td>00000000 00000000 00000000</td>
<td></td>
</tr>
</tbody>
</table>

   The first word of the "System Call" column identifies whether the trace entry corresponds to a syscall entry (CALL) or syscall exit (EXIT). The second word identifies the syscall. For "CALL" type entries, parameters that are shown in the "Additional Trace Data" column must be matched with the input and output parameters for the specified syscall. Note that the output parameters might contain residual data upon entry to a syscall. The pgm=field displays information about the process being traced only if it remains active after **bpxtrace** has ended. For "EXIT" type entries, the return value (RV) returned from the syscall is displayed in the "Additional Trace Data" column. Both the input parameter mapping and the return value meaning can be found in **z/OS UNIX System Services Programming: Assembler Callable Services Reference**.
• full corresponds to the full format option of the IPCS CTRACE command when specified with the KERNINFO option.

Restriction: This option is intended for use by service personnel only.

CTRACE COMP(SYSOMVS) FULL OPTIONS((KERNINFO))

The following display shows an example of output from bpxtrace -f full:

```
FCN...open       SYSCALL...BPX1OPN  PID...00000007  MODULE...BPXJCPC
SY1       SYSCALL  0F080001  09:43:04.070651  STANDARD SYSCALL ENTRY TRACE
ASID...0025     USERID... MEGA     STACK@...26D9D0A8
TCB...088F1D8   EUID.... MEGA     PID.... 00000007
+0000  00000026  00000000  D1C3E2C5  8C008004  | .........JCSE.... |
+0010  0000000C  00000000  8288B768  FFFFFFFF  | .........bh....- |
+0020  00000002  26D9EB84  00000000  7F6DCBB8  | .........Ry...... |
+0030  C0000007  00000000  61828989  618297A7  | {........./bin/bpx |
+0040 A699A3A2  96000000  00000000  00000000  | wtrso/......../ |
+0050  00000000  00000000  00000000  00000000  | ........................ |
+0060  00000000  00000000  00000000  00000000  | ........................ |
+0070  00000000  00000000  00000000  00000000  | ........................ |
+0080  00000000  00000000  00000000  00000000  | ........................ |
FCN...open       SYSCALL...BPX1OPN  PID...00000007  MODULE...BPXJCPC
SY1       SYSCALL  0F080002  09:43:04.070668  STANDARD SYSCALL EXIT TRACE
ASID...0025     USERID... MEGA     STACK@...26D9D09A
TCB...088F1D8   EUID.... MEGA     PID.... 00000007
+0000  00000026  00000000  D1C3E2C5  8C008000  | .........JCSE.... |
+0010  0000000C  00000000  82885438  FFFFFFFF  | .........bh...... |
+0020  00000000  618297A7  00000000  7F6DCBBC  | ........./bin/bpx |
+0030  00000000  25D00000  00000000  00000000  | ........................ |
+0040  00001000  00000007  00000000  00000000  | ........................ |
FCN...read       SYSCALL...BPX1RED  PID...00000007  MODULE...BPXJCPC
SY1       SYSCALL  0F080001  09:43:04.070675  STANDARD SYSCALL ENTRY TRACE
ASID...0025     USERID... MEGA     STACK@...26D9D09A
TCB...088F1D8   EUID.... MEGA     PID.... 00000007
+0000  0000002B  00000000  D1C3E2C5  8C008000  | .........JCSE.... |
+0010  00000000  00000000  00000000  00000000  | ........................ |
+0020  00000000  618297A7  00000000  7F6DCBBC  | ....../bin/bpx... |
+0030  40000007  00000007  25D00000  00000000  | ........................ |
+0040  00000000  00000000  00000000  00000000  | ........................ |
FCN...read       SYSCALL...BPX1RED  PID...00000007  MODULE...BPXJCPC
SY1       SYSCALL  0F080002  09:43:04.091468  STANDARD SYSCALL EXIT TRACE
ASID...0025     USERID... MEGA     STACK@...26D9D09A
TCB...088F1D8   EUID.... MEGA     PID.... 00000007
+0000  0000002B  00000000  D1C3E2C5  8C008000  | .........JCSE.... |
+0010  80000000  00000000  00000000  00000000  | ........................ |
+0020  00000000  00000000  00000000  00000000  | ........................ |
```

• short corresponds to the IPCS CTRACE command short format option:

CTRACE COMP(SYSOMVS) SHORT

For an example of the SYSOMVS trace record when the short format option is specified, see z/OS MVS Diagnosis: Tools and Service Aids.

• counts corresponds to the IPCS CTRACE command SYSOMVS sccounts option

CTRACE COMP(SYSOMVS) OPTIONS((SCCOUNTS))

For an example of the SYSOMVS trace record when the sccounts format option is specified, see z/OS MVS Diagnosis: Tools and Service Aids.

The default is -f format.

-h columns
Selects the columns for trace output when -f format is used. columns is specified as a string of characters where each character represents a column.

p
PID column
The characters can be specified in any order but the column order in the output is not affected. An incorrect specification for the columns causes the default to be used. The default is 

- **i**
  Produces trace output, leaving trace enabled.

- **o outputpath**
  Specifies the path name of the z/OS UNIX file to contain the trace output. The default is to write the trace records to the standard output stream, which might cause a significant amount of trace output. When running in a shell environment, stdout can be redirected to a file; otherwise you can use the -o option to avoid having the output written to your session.

  If the specified file exists, the contents are replaced. If redirection is used, the results depend on the redirection operator.

- **p pid,...**
  Specifies the process ID of the process to have trace started, stopped, or formatted. When formatting a trace, the default formatting shows only the records for the processes that are identified by the list of specified PIDs. To have a list of multiple processes traced, specify multiple -p pids instances on the command. By default, tracing is started for all processes for the user.

- **s**
  Starts tracing user processes.

- **S**
  Produces trace output for unusual or unexpected conditions reported by the System Authorization Facility (SAF). To avoid flooding its trace with SAF results, z/OS UNIX traces only those conditions where the callable service documentation and return values do not provide enough information to determine the cause of the problem.

  Users with an effective UID of 0 will see the trace records for all UIDs while non-effective UID 0 users will see only those trace records with a matching UID. The first 40 (decimal) bytes of these trace records are mapped by the ThliSecErrCT section of the BPXYTHLI macro, which is documented in z/OS UNIX System Services Programming: Assembler Callable Services Reference. The portion of the records beyond the first 40 bytes are for use by IBM service.

  The trace facilities utilized by the -S option do not require the prior activation of tracing for the system or for a given process. The trace facilities that are used for the -S option are always active and do not need to be activated or deactivated.

  The following z/OS UNIX services create trace records for unusual or unexpected conditions reported by SAF:

  - BPX1PWD (__passwd)
  - BPX1SUI (setuid)
  - BPX1SEU (seteuid)
  - BPX1SRU (setreuid)
  - BPX1TLS (pthread_security)
  - BPX1GGN (getgrnam)
- BPX1GGR (getgroups)
- BPX1SEC (__login)

-t seconds
Traces for a specified period and then produces output. At the end of the time period, the traced processes will no longer have tracing on.

-T tempdir
Specifies the path name of the directory to be used for temporary files. By default, if bpxtrace is running in a shell environment, it uses the TMPDIR environment variable if available. Otherwise, /tmp is used.

-u userid
Specifies the user ID that is used in processing the trace. If the user ID is not the same as that of the invoking user, bpxtrace attempts to switch to UID(0) before determining which processes are to be traced. If the invoking user does not have permission to switch to UID(0), the bpxtrace command will fail. If both -p and -u are specified, -u is ignored.

-v volser
Specifies the volume to use for allocating temporary data sets. If -v is not specified, unit(sysallda) is used.

-x
Specifies that the output contains only exit trace lines. -f format must also be specified.

Examples
1. To trace the calls made by the df utility and capture the output in a file:

   bpxtrace -c -o trace.output df

Usage notes
If the effective UID of the user is changed after BPXTRACE tracing started, CTRACE records might be missing from the BPXTRACE output. Due to limitations in DUMP processing, only CTRACE records matching the user's EUID at the time of the dump are captured and made available for display.

Exit values
0
Successful completion.
1
The trace was successful but a listing was not generated.
3
Command-line syntax error.
4
Data set or file allocation error.
5
The seteuid failed.
8
z/OS UNIX callable service BPXGMCMDE (MVS IPCS dump open/close service) open failure. This error indicates that the dump data set could not be opened. Try the command again. If the error persists, contact IBM Service.
9
z/OS UNIX callable service BPXGMMPTR (Ptrace IPCS dump access service) command failure. This error indicates that a resource constraint was encountered while attempting to process the dump. Try the command again. If the error persists, contact IBM Service.
Review the error log to determine the problem.

**bpxwmigf - Migrate HFS file systems to zFS**

**Format**

```
bpxwmigf -query [sourcefs][-filter active|success|failed]
bpxwmigf -cancel sourcefs
bpxwmigf -source sourcefs [target targetfs][-mode rw|ro|asis]
[-parm fsparms|-noparm][-swap|-noswap][-srename newname|-nosrename]
[-trename newtname|-notrename][-wait minutes][-status minutes]
[-priority priority]
```

**Description**

Use the `bpxwmigf` command to migrate HFS file systems to zFS. Superuser authority is required.

**Options**

**-filter**

Filters query results. The `-filter` option is ignored for a query for a specific migration.

- `active`
  Filters for currently active migrations. The results include migrations that are queued, mirroring, mirror complete, swapping, and swap defer.

- `failed`
  Filters for failed and canceled migrations.

- `success`
  Filters for successful migrations.

**-mode rw|ro|asis**

The final mount mode for the target zFS file system.

- `rw`
  The zFS file system is mounted read/write.

- `ro`
  The zFS file system is mounted read-only.

- `asis`
  The file system is mounted as is. It is the default.

**-parm|-noparm**

Specifies the parameter options.

- `parm`
  The mount parameter for the zFS file system after the migration is completed.

- `noparm`
  Clears a previous mount request with `-parm`. Following are examples of when you might want to use this option:

  - You have previously set `-parm` but then realize that you no longer want a zFS parameter for this particular mount.
  - If you prefer to specify all command parameters without allowing defaults, you can specify `-noparm` if you do not want a mount parameter.
-priority priority
  Specifies the migration priority 1 - 9 where 1 is the highest priority and 9 is the lowest. To slow down
  the migration, specify a higher number. Using a higher number reduces the impact to applications that
  access the migrating file systems or to extend the migration resource over a longer period.
  Default: 2

-sourcesfs
  The name of the HFS file system to be migrated.

-srename newname|nosrename
  Specifies whether to rename the source HFS file system to newname.
  -srename newname
    Renames the source HFS file system to newname.
  -nosrename
    Does not rename the source HFS file system to newname. It is the default.

-status minutes
  Specifies the status update interval during the wait.

-swap|noswap
  Specifies the swap options.
  -swap
    Performs the swap as soon as the file systems are synchronized.
  -noswap
    Maintains the file systems in synchronization but the file systems are not swapped until a request
    with -swap is made. It is the default.

-targetfs targetfs
  The name of the target zFS file system.

-trename|notrename
  Specifies whether to rename the target zFS file system.
  -rename
    The target zFS file system is renamed after the swap to the source HFS name. -srename must be
    specified or be already in effect.
  -notrename
    The target zFS file system is not renamed after the swap.

-trename newtname
  The target zFS file system is renamed to newtname after the swap.

-wait minutes
  The migration command does not end until the specified number of minutes has elapsed or when the
  migration completes.

Usage notes
1. The target zFS file system must be allocated and formatted as a version 5 zFS file system, and not
   mounted.
2. The target zFS must be empty before migration can begin. If the target zFS is being reused, it must be
   reformatted before migration can begin.
3. To make sure that the proper file system is mounted after a system restart, ensure that the mount
   statements or mount policies are current after the swap.
4. Ensure that the mount statements or mount policies will not mount the new file system after a system
   restart if the migration is canceled.
5. If the command is canceled or aborted and the migration processing has begun, the migration
   continues.
6. The -query option provides information that indicates percent complete, based on the ratio of byte sizes of the target/source.

7. The migration must be started from the z/OS UNIX-owning system.

8. All systems in the shared file system environment must be at least the z/OS V2R3 level.

9. Only one migration at a time can be performed.

Examples

1. To start a migration, issue the following command:

   ```
   bpxwmigf -source omvs.a1234.hfs -target omvs.xyz.zfs -wait 10
   ```

2. After a migration is started, you can override the swap parameters or trigger the swap after a migration is complete but is still unswapped by issuing the following command:

   ```
   bpxwmigf -source omvs.a1234.hfs -mode ro -swap
   ```

3. To cancel a current migration, issue the following command:

   ```
   bpxwmigf -cancel OMVS.1234.HFS
   ```

4. To query the progress of all current migrations, issue the following command:

   ```
   bpxwmigf -query
   ```

5. To wait for status on a migration in progress, issue the following command:

   ```
   bpxwmigf -source omvs.a1234.hfs -wait 10
   ```

Localization

`bpxwmigf` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

For more information, see Appendix F, “Localization,” on page 965.

Exit values

0
  No errors.

1
  Wait completed with noswap.

2
  Wait time expired.

3
  Wait exit due to canceled or failed migration.

7
  Usage error; see the error message.

8
  Usage error; see message.
3
Query failed; see message.

12
Cannot access z/OS UNIX.

Messages
Most diagnostics deal with argument syntax and are self-explanatory. For example:

**Badly constructed output list at list**
Indicates that the list for a -o option did not have the proper syntax.

Portability
An approved POSIX standard does not exist for \texttt{bpxwmigf}.

Related information
None.

---

**break - Exit from a loop in a shell script**

Format
\texttt{break [number]}
\texttt{tcsh shell: break}

Description
\texttt{break} exits from a for, select, while, or until loop in a shell script. If \texttt{number} is given, \texttt{break} exits from the given number of enclosing loops. The default value of \texttt{number} is 1.

\texttt{break} is a special built-in shell command.

In the tcsh shell, \texttt{break} causes execution to resume after the end of the nearest enclosing foreach or while. The remaining commands on the current line are executed. Multilevel breaks are thus possible by writing them all on one line.

Localization
\texttt{break} uses the following localization environment variables:

- \texttt{LANG}
- \texttt{LC_ALL}
- \texttt{LC_CTYPE}
- \texttt{LC_MESSAGES}
- \texttt{NLSPATH}


Exit values
\texttt{break} always exits with an exit status of 0.

Portability
POSIX.2, X/Open Portability Guide.
c++ - Compile C and C++ source code, link-edit and create an executable file

See c89/xlc or man xlc.

Note: When working in the shell, to view man page information about c++, type: man c89 or man xlc.

c89 - Compiler invocation using host environment variables

Format

```c
| c89 | cc | c++ [+CcEFgOqrsVv0123] |
[-D name[=value]]... [-U name]...
[-e function] [-u function]...
[-W phase[=option]]...
[-o outfile]
[-I directory]... [-L directory]...
```

Notes:

1. The `c99` command is only supported by the `xlc` utility.
2. In this information, `-l` signifies `-l` (a lowercase L) and not an uppercase `I`.

Description

The `c89` and `cc` commands compile, assemble, and link-edit C programs; the `cxx` or `c++` command does the same for C++ programs.

- The `c89` command should be used when compiling C programs that are written according to Standard C.
- The `cc` command should be used when compiling C programs that are written according to Common Usage C.
- The `cxx` or `c++` command must be used when compiling C++ programs. Prior to z/OS V1R2, the C++ compiler supported the Draft Proposal International Standard for Information Systems — Programming Language C++ (X3J16). As of z/OS V1R7, the C++ compiler supports the Programming languages - C++ (ISO/IEC 14882:2003(E)) standard, as well as the Programming languages - C++ (ISO/IEC 14882:1998) standard. The `c++` command can compile both C++ and C programs, and can also be invoked by the name `cxx` (all references to the `c++` command throughout this document apply to both names).

The `c89`, `cc`, and `c++` commands call other programs for each step of the compilation, assemble, and link-editing phases. Table 4 on page 79 contains the step name and the name of the document that describes the program you use for that step and the document that describes any messages issued by that program, and prefixes to those messages.
Table 4: Reference documentation for programs invoked by c89, cc, and c++ commands

<table>
<thead>
<tr>
<th>Step Name</th>
<th>Document Describing Options and How to Call Program</th>
<th>Document Containing Messages Issued by Program</th>
<th>Prefix of Messages Issued by Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPILE, IPACOMP, TEMPLINC, IPATEMP, IPALINK</td>
<td>z/OS C/C++ User’s Guide for releases prior to z/OS V1R7 and z/OS XL C/C++ User’s Guide for z/OS V1R7 and later releases</td>
<td>z/OS C/C++ Messages for z/OS V1R5 and z/OS V1R6 releases and z/OS XL C/C++ Messages for z/OS V1R7 and later releases</td>
<td>CCN for z/OS V1R2 and later releases</td>
</tr>
<tr>
<td>LINKEDIT (Program Management Binder)</td>
<td>z/OS MVS Program Management: User's Guide and Reference</td>
<td>z/OS MVS System Messages, Vol 8 (IEF-IGD)</td>
<td>IEW</td>
</tr>
</tbody>
</table>

Execution of any Language Environment® program can result in runtime messages. These messages are described in z/OS Language Environment Runtime Messages and have an EDC prefix.

In order for the c89, cc, and c++ commands to perform C and C++ compiles, the z/OS C/C++ Optional Feature must be installed on the system. The z/OS C/C++ Optional Feature provides a C compiler, a C++ compiler, C++ Class Libraries, and some utilities. See z/OS Introduction and Release Guide for further details. Also see prefix_CLIB_PREFIX and prefix_PLIB_PREFIX in “Environment variables” on page 92 for information about the names of the z/OS XL C/C++ Optional Feature data sets that must be made available to the c89/cc/c++ command.

**Note:** The term prefix is defined in “Environment variables” on page 92.

First, the c89, cc, and c++ commands perform the compilation phase (including preprocessing) by compiling all source file operands (file.C, file.i, and file.c, as appropriate). For the c++ command, if automatic template generation is being used (which is the default), then z/OS XL C++ source files may be created or updated in the tempinc subdirectory of the working directory during the compilation phase (the tempinc subdirectory will be created if it does not already exist). Then, the c89, cc, and c++ commands perform the assemble phase by assembling all operands of the file.s form. The result of each compile step and each assemble step is a file.o file. If all compilations and assemblies are successful, or if only file.o and/or file.a files are specified, the c89, cc, and c++ commands proceed to the link-editing phase. For the c++ command, the link-editing phase begins with an automatic template generation step when applicable. For IPA (Interprocedural Analysis) optimization an additional IPA Link step comes next. The link-edit step is last. See the environment variable prefix_STEPS under “Environment variables” on page 92 for more information about the link-editing phase steps.

In the link-editing phase, the c89, cc, and c++ commands combine all file.o files from the compilation phase along with any file.C files that were specified on the command line. For the c++ command, this is preceded by compiling all C++ source files in the tempinc subdirectory of the working directory (possibly creating and updating additional C++ source files during the automatic template generation step). After compiling all the C++ source files, the resulting object files are combined along with the file.o files from the compilation phase and the command line. Any file.a files, file.x files and -l libname operands that were specified are also used.
The usual output of the link-editing phase is an executable file. For the c89, cc, and c++ commands to produce an executable file, you must specify at least one operand which is of other than -l libname form. If -r is used, the output file is not executable.

For more information about automatic template generation, see z/OS XL C/C++ User's Guide and the information on Using the TEMPINC compiler option in z/OS XL C/C++ Programming Guide. Note that the c++ command only supports using the tempinc subdirectory of the working directory for automatic template generation.

For more information on IPA, see “Options” on page 80.

Options

++
Specifies that all source files are to be recognized as C++ source files. All .c, .o, and .a files will continue to be recognized as assembler source, object, and archive files respectively. However, any C .c or .i files will be processed as corresponding C++ .C or .i files, and any other file suffix which would otherwise be unrecognized will be processed as a .C file.

This option effectively overrides the environment variable prefix_EXTRA_ARGS. This option is only supported by the c++ command.

-C
Specifies that C and C++ source comments should be retained by the preprocessor. By default, all comments are removed by the preprocessor. This option is ignored except when used with the -E option.

-c
Specifies that only compilations and assemblies be done. Link-edit is not done.

-D name[=value]
Defines a C or C++ macro for use in compilation. If only name is provided, a value of 1 is used for the macro it specifies. For information about macros that the c89/cc/c++ command automatically defines, see Usage Note 5 and Usage Note 13.

Notes:

• The xlc utility has slightly different semantics for processing -D options.

• As of z/OS V1R12, to define a macro name that contains an escape character (that is, the backslash) using an option such as -D or -Wc,DEFINE, you must specify the option in a way that can preserve the backslash character when the macro reaches the compiler parser. Because an option passes through the UNIX shell and the compiler options processor, both of which are sensitive to backslash characters, the rules for such characters must be followed to ensure that the compiler parser receives a macro with the backslash character. The UNIX shell and the compiler options parser both interpret and consume backslash characters that are unquoted or quoted by double quotation marks. On the other hand, the UNIX shell does not interpret backslash characters that are quoted by single quotation marks, while the compiler options parser is not sensitive to single quotation marks. For example, for the compiler parser to receive \u0024 as the macro symbol, the compiler options processor must receive \u0024, so you must specify -D'\\u0024' or -D"\\u0024" on the command line. This also applies to the -Wc,DEFINE option, which is an alternative method of defining macros (for example, -Wc,'DEFINE(\\u0024)' or -Wc,"DEFINE(\\u0024)"). The same is true for any compiler option that requires the use of a backslash to suppress special meaning of a particular character.

-E
Specifies that output of the compiler preprocessor phase be copied to stdout. Object files are not created, and no link-editing is performed.

-e function
Specifies the name of the function to be used as the entry point of the program. This can be useful when creating a fetchable program, or a non-C or non-C++ main, such as a COBOL program. Non-C++ linkage symbols of up to 1024 characters in length may be specified. You can specify an S-name by...
preceding the function name with double slash (//). (For more information about S-names, see Usage Note 23.)

Specify a null S-name ("-e //") so that no function name is identified by the c89/cc/c++ command as the entry point of the program. In that case, the Program Management Binder (link editor) default rules will determine the entry point of the program. For more information about the Program Management Binder and the ENTRY control statement, see z/OS MVS Program Management: User's Guide and Reference.

The function //ceestart is the default. When the default function entry point is used, a binder ORDER control statement is generated by the c89/cc/c++ command to cause the CEESTART code section to be ordered to the beginning of the program. Specify the name with a trailing blank to disable this behavior, as in "//ceestart ". For more information about the Program Management Binder and the ORDER control statement, see z/OS MVS Program Management: User's Guide and Reference.

This option might be required when building products which are intended to be installed using the IBM SMP/E product. When installing ++MOD elements with SMP/E, binder control statements should be provided in the JCLIN created to install the product instead of being embedded in the elements themselves.

-F
Ignored by the cc command. Provided for compatibility with historical implementations of the cc command. Flagged as an error by the c89 and c++ commands.

-f
Ignored by the cc command. Provided for compatibility with historical implementations of the cc command. Flagged as an error by the c89 and c++ commands.

Historical implementations of C/C++ used this option to enable floating-point support. Floating-point is automatically included in z/OS XL C/C++. However, in z/OS XL C/C++, two types of floating-point support are available:

HEXADECIMAL
Base 16 IBM System z9 hexadecimal format. The IBM System z9 hexadecimal format is referred to as the hexadecimal floating-point format, and is unique to IBM System z9 hardware. This is the default.

IEEE754
Base 2 IEEE-754 binary format. The IEEE-754 binary format is referred to as binary floating-point format. The IEEE-754 binary format is the more common floating point format used on other platforms.

If you are porting an application from another platform, transmitting floating-point numbers between other platforms or workstations, or your application requires the larger exponent range provided by IEEE-754 binary format, then you should consider using IEEE floating-point. The contains more information on the FLOAT compiler option.

An example of compiling with IEEE-754 binary floating point format:

c89 -o outfile -Wc,'float(ieee)' file.c

-g
Specifies that a side file that contains symbolic information is emitted and the executable is to be loaded into read/write storage, which is required for source-level debugging with dbx, and other debuggers.

For 32-bit compiles, if the _DEBUG_FORMAT=ISD environment variable is exported, then -g specifies that the output file (executable) is to contain symbolic information and is to be loaded into read/write storage, which is required for source-level debugging with dbx, and other debuggers.

When specified for the compilation phase, the compiler produces symbolic information for source-level debugging.
When specified for the link-editing phase, the executable file is marked as being serially reusable and will always be loaded into read/write storage.

dbx requires that all the executables comprising the process be loaded into read/write storage so that it can set break points in these executables. When dbx is attached to a running process, this cannot be guaranteed because the process was already running and some executables were already loaded.

There are two techniques that will cause all the executables comprising the process to be loaded into read-write storage:

1. Specify the -g option for the link-editing phase of each executable. After this is done, the executable is always loaded into read/write storage.

   Because the executable is marked as being serially reusable, this technique works except in cases where the executable must be marked as being reentrant. For example:

   • If the executable is to be used by multiple processes in the same user space.
   • If the executable is a DLL that is used on more than one thread in a multithreaded program.

   In these cases, use the following technique instead:

2. Do not specify the -g option during the link-editing phase so that the executable will be marked as being reentrant. Before invoking the program, export the environment variable _BPX_PTRACE_ATTACH with a value of YES. After you do this, then executables will be loaded into read/write storage regardless of their reusability attribute.

If you compile an MVS data set source using the -g option, you can use dbx to perform source-level debugging for the executable file. You must first issue the dbx use subcommand to specify a path of double slash (//), causing dbx to recognize that the symbolic name of the primary source file is an MVS data set.

For more information about using dbx, see z/OS UNIX System Services Programming Tools.

For more information on the _BPX_PTRACE_ATTACH environment variable, see z/OS UNIX System Services Programming: Assembler Callable Services Reference.

The GONUMBER option is automatically turned on by the -g option, but can also be turned on independently. There is no execution path overhead incurred for turning on this option, only some additional space for the saved line number tables.

For 31-bit compiles and In Storage Debug (ISD) information, the GONUMBER option generates tables that correspond to the input source file line numbers. These tables make it possible for Debug Tools and for error trace back information in CEE dumps to display the source line numbers. Having source line numbers in CEE dumps improves serviceability costs of applications in production.

An example of compiling with the GONUMBER compiler option:

```
c89 -o outfile -Wc,'GONUM' file.c
```

**Note:** -g forces the NOOPTIMIZE compiler option regardless of its position in the command line.

**-I directory**

**Note:** The I option signifies an uppercase i, not a lowercase L.

-1 specifies the directories to be used during compilation in searching for include files (also called header files).

Absolute path names specified on #include directives are searched exactly as specified. The directories specified using the -I option or from the usual places are not searched.

If absolute path names are not specified on #include directives, then the search order is as follows:

1. Include files that are enclosed in double quotation marks (") are first searched for in the directory of the file containing the #include directive. Include files that are enclosed in angle-brackets (< >) skip this initial search.
2. The include files are then searched for in all directories specified by the -I option, in the order specified.

3. Finally, the include files are searched for in the usual places. (See Usage Note 4 for a description of the usual places.)

You can specify an MVS data set name as an include file search directory. Also, MVS data set names can explicitly be specified on #include directives. You can indicate both by specifying a leading double slash (//).

To include the include file DEF that is a member of the MVS PDS ABC.HDRS, code your C or C++ source as follows:

```c
#include <//'abc.hdrs(def)'>
```

MVS data set include files are handled according to z/OS XL C/C++ compiler conversion rules (See Usage Note 4). When specifying an #include directive with a leading double slash (in a format other than #include<//'dsname'> and #include<//dd:ddname>), the specified name is paired only with MVS data set names specified on the -I option. That is, when you explicitly specify an MVS data set name, any z/OS UNIX file system directory names specified on the -I option are ignored.

**Note:** As of z/OS V1R12, a directory name that contains a comma must be quoted by double quotation marks, and the comma must be escaped by the backslash character. For example, -Imy, directory can result in two directories "my" and "directory". If the intended name is a single directory name that contains a comma, the option must be specified as -I"my\,directory" to suppress the special meaning of the comma as suboption separator.

**-L directory**

Specifies the directories to be used to search for archive libraries specified by the -l operand. The directories are searched in the order specified, followed by the usual places. You cannot specify an MVS data set as an archive library directory.

For information on specifying C370LIB libraries, see the description of the -l libname operand. Also see Usage Note 7 for a description of the usual places.

**-0, -O (-1), -2, -3**

Specifies the level of compiler optimization (including inlining) to be used. The level -1 (number one) is equivalent to -O (capital letter O). The level -3 gives the highest level of optimization. The default is -0 (level zero), no optimization and no inlining, when not using IPA (Interprocedural Analysis).

When optimization is specified, the default is ANSI/ALIAS. The ANSI/ALIAS default specifies whether type-based aliasing is to be used during optimization. That is, the optimizer assumes that pointers can only be used to access objects of the same type. Type-based aliasing improves optimization. Applications that use pointers that point to objects of a different type will need to specify NOANSIALIAS when the optimization compiler option is specified. If your application works when compiled with no optimization and fails when compiled with optimization, then try compiling your application with both optimization and NOANSIALIAS compiler options. The z/OS XL C/C++ User's Guide contains more information on ANSI/ALIAS.

**Notes:**

1. Options can also be specified as -O1 (using capital letter O), -O2, and -O3. For further information, see Usage Note 12.

2. These options cannot be overridden by specifying optimization options using the -Wc syntax. This behavior differs from the behavior of the xlc utility, which allows the use of -q and -Wc syntax to override the flag optimization options.

An example of a compile with the highest level of optimization and no type-based aliasing:

```bash
c89 -o outfile -O3 -Wc,NOANSIALIAS file.c
```

When optimization is specified, you may want to obtain a report on the amount of inlining performed and increase or decrease the level of inlining. More inlining will improve application performance and...
increase application memory usage. The z/OS XL C/C++ User’s Guide contains more information on the INLINE compiler option.

An example of a compilation with optimization with no report generated, a threshold of 500 abstract code units, and a limit of 2500 abstract code units:

```
c89 -o outfile -O2 -Wc,'inline(auto,noreport,500,2500)' file.c
```

When using IPA, the default is -O (level 1) optimization and inlining. IPA optimization is independent from and can be specified in addition to this optimization level. IPA is further described in this topic.

If compiling with PDF, the same optimization level must be used in the PDF1 and PDF2 steps.

If you compile your program to take advantage of dbx source-level debugging and specify -g (see the -g option description in this topic), you will always get -0 (level zero) optimization regardless of which of these compiler optimization levels you specify.

In addition to using optimization techniques, you may want to control writable strings by using the #pragma strings,readonly) directive or the ROSTRING compiler option. As of z/OS Version 1 Release 2, ROSTRING is the default.

For more information on this topic, refer to reentrancy in z/OS XL C/C++ in z/OS XL C/C++ Programming Guide or the description of the ROSTRING option in the z/OS XL C/C++ User’s Guide.

-o outfile

Specifies the out put file name of the c89/cc/c++ command.

If the -o option is specified in addition to the -c option, and only one source file is specified, then this option specifies the name of the output file associated with the one source file. See file.o under “Operands” on page 90 for information on the default name of the output file.

Otherwise the -o option specifies the name of the executable file produced during the link-editing phase. The default output file is a.out.

-p

Ignored by the cc command. Provided for compatibility with historical implementations of the cc command. Flagged as an error by the c89 and c++ commands.

-q

Ignored by the cc command. Provided for compatibility with historical implementations of the cc command. Flagged as an error by the c89 and c++ commands.

-r

Specifies that the c89/cc/c++ command is to save relocation information about the object files which are processed. When the output file (as specified on -o) is created, it is not made an executable file. Instead, this output file can later be used as input to the c89/cc/c++ command. This can be used as an alternative to an archive library.

IPA usage note: When using -r and link-editing IPA compiled object files, you must link-edit with IPA (see the description of IPA under the -W option). However, the -r option is typically not useful when you create an IPA optimized program. This is because link-editing with IPA requires that all of the program information is available to the link editor. It is not acceptable to have unresolved symbols, especially the program entry point symbol, which is usually main. The -r option is normally used when you want to combine object files incrementally. Specify object files during the initial link-edit that uses -r. Later, specify the output of the initial link-edit, along with the remaining object files in a final link-edit that is done without using -r. When you want to combine IPA compiled object files, use the alternative that does not involve the link editor, that is, concatenating the object files into one larger file by using the cp or cat utilities. You can use this larger file later in a final link-edit when the remainder of the object files are also made available.

-S

Specifies that the output file produced by the compiler is in assembler source code format. The absence of the -S flag indicates that the output file produced is in object code format. The -S flag is
supported only with the METAL compiler option. The compiler does not produce an object file when the -S flag is used.

By default, the assembler source file name is based on the C source file name specified on the command line. The suffix is determined based on the appropriate environment variable. However, the assembler source file name can be affected by the use of the -o option.

When you specify the -o option, the assembler source file name is based on the name specified with the option. For example, when you specify c89 -S -Wc,metal -c -o foo.x hello.c, the output assembler source file name is foo.x.

The following specifications have the same result:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>c89 -S -Wc,metal hello.c</td>
</tr>
<tr>
<td>c89 -S -Wc,metal -o hello.s hello.c</td>
</tr>
<tr>
<td>c89 -S -Wc,metal -c hello.c</td>
</tr>
<tr>
<td>c89 -S -Wc,metal -c -o hello.s hello.c</td>
</tr>
</tbody>
</table>

-s Specifies that the compilation phase is to produce a file.o file that does not include symbolic information, and that the link-editing phase produce an executable that is marked reentrant. This is the default behavior for the c89/cc/c++ command.

-U name Undefines a C or C++ macro specified with name. This option affects only macros defined by the -D option, including those automatically specified by the c89/cc/c++ command. For information about macros that the c89/cc/c++ command automatically define, see Usage Note 5 and Usage note 13.

Note: The xlc utility uses different semantics for handling the -U option.

-u function Specifies the name of the function to be added to the list of symbols which are not yet defined. This can be useful if the only input to the c89/cc/c++ command is archive libraries. Non–C++ linkage symbols of up to 255 characters in length may be specified. You can specify an S-name by preceding the function name with double slash (//). (For more information about S-names, see Usage Note 23.) The function //ceemain is the default for non-IPA Link-editing, and the function main is the default for IPA Link-editing. However, if this -u option is used, or the DLL link editor option is used, then the default function is not added to the list.

-V This verbose option produces and directs output to stdout as compiler, assembler, IPA linker, prelinker, and link editor listings. If the -O, -2, or -3 options are specified and cause the c89/cc/c++ command to use the compiler INLINE option, then the inline report is also produced with the compiler listing. Error output continues to be directed to stderr. Because this option causes the c89/cc/c++ command to change the options passed to the steps producing these listings so that they produce more information, it may also result in additional messages being directed to stderr. In the case of the compile step, it may also result in the return code of the compiler changing from 0 to 4.

Note: This option has a different meaning when using the xlc utility.

-v This verbose option causes pseudo-JCL to be written to stdout before the compiler, assembler, IPA linker, prelinker, and link editor programs are run.

It also causes phaseid information to be emitted in stderr:

FSUM0000I Utility(c89) Level(U999999)

It provides information about exactly which compiler, prelinker, and link editor options are being passed, and also which data sets are being used. If you want to obtain this information without actually invoking the underlying programs, specify the -v option more than once on the c89/cc/c++ command string. For more information about the programs which are executed, see Usage Note 14.
Specifies options to be passed to the steps associated with the compile, assemble, or link-editing phases of the `c89/cc/c++` command. The valid phase codes are:

- **0** Specifies the compile phase (used for both non-IPA and IPA compilation).
- **a** Specifies the assemble phase.
- **c** Same as phase code 0.
- **I** Enables IPA (Interprocedural Analysis) optimization.

Unlike other phase codes, the IPA phase code I does not require that any additional options be specified, but it does allow them. In order to pass IPA suboptions, specify those suboptions using the IPA phase code.

To specify that an IPA Compile should save source line number information, without writing a listing file, specify:

```
c89 -c -W I,list file.c
```

To specify that an IPA Link-edit should write the map file to stdout, specify:

```
c89 -W I,map file.o
```

- **l** Specifies the link-editing phase.
  - To pass options to the prelinker, the first link-editing phase option must be **p** or **P**. All the following options are then prelink options.

    To write the prelink map to stdout, specify:
    
    ```
c89 -W l,p,map file.c
```

    **Note:** The prelinker is no longer used in the link-editing phase in most circumstances. If it is not used, any options that are passed are accepted but ignored. See the environment variable `prefix_STEPS` under `Environment variables` for more information about the link-editing phase prelink step.
  - To pass options to the IPA linker, the first link-editing phase option must be **i** or **I**. All the following options are then IPA Link options.

    To specify the size of the SPILL area to be used during an IPA Link-edit, you could specify:

    ```
c89 -W l,I,"spill(256)" file.o
```

- To link-edit a DLL (Dynamic Link Library) and produce a sidedeck, the link-editing phase option DLL must be specified.

    To accomplish this task, you could specify:

    ```
c89 -o outdll -W l,dll file.o
```

Most z/OS XL C/C++ extensions can be enabled by using this option. Those which do not directly pass options through to the underlying steps, or involve files which are extensions to the compile and link-edit model, are described here:

**DLL (Dynamic Link Library)**

A DLL is a part of a program that is not statically bound to the program. Instead, linkage to symbols (variables and functions) is completed dynamically at execution time. DLLs can improve storage utilization, because the program can be broken into smaller parts, and some parts may
not always need to be loaded. DLLs can improve maintainability, because the individual parts can be managed and serviced separately.

In order to create a DLL, some symbols must be identified as being exported for use by other parts of the program. This can be done with the z/OS XL C/C++ #pragma export compiler directive, or by using the z/OS XL C/C++ EXPORTALL compiler option. If during the link-editing phase some of the parts have exported symbols, the executable which is created is a DLL. In addition to the DLL, a definition sidedeck is created, containing link-editing phase IMPORT control statements which name those symbols which were exported by the DLL. In order for the definition sidedeck to be created, the DLL link editor option must be specified. This definition sidedeck is subsequently used during the link-editing phase of a program which is to use the DLL. See the file.x operand description in the “Operands” on page 90 topic for information on where the definition sidedeck is written. In order for the program to refer to symbols exported by the DLL, it must be compiled with the DLL compiler option.

To compile and link a program into a DLL, you could specify:

```
c89 -o outdll -W c,exportall -W l,dll file.c
```

To subsequently use file.x definition side-decks, specify them along with any other file.o object files specified for the c89/cc/c++ link-editing phase.

To accomplish this task, you could specify:

```
c89 -o myappl -W c,dll myappl.c outdll.x
```

In order to run an application which is link-edited with a definition sidedeck, the DLL must be made available (the definition sidedeck created along with the DLL is not needed at execution time). When the DLL resides in the z/OS UNIX file system, it must be in either the working directory or in a directory named on the LIBPATH environment variable. Otherwise it must be a member of a data set in the search order used for MVS programs.

**Note:** For non-DLL C++ compiles, a dummy definition side file will be allocated to prevent the binder from issuing a warning message. If you do want the binder to issue a warning message when an exported symbol is encountered, specify the DLL=NO option for the link-editing phase; for example:

```
c++ -o outfile -W l,dll=no file.C
```

**IPA (interprocedural analysis)**

IPA optimization is independent from and can be used in addition to the c89/cc/c++ optimization level options (such as -O). IPA optimization can also improve the execution time of your application. IPA is a mechanism for performing optimizations across function boundaries, even across compilation units. It also performs optimizations not otherwise available with the z/OS XL C/C++ compiler.

When phase code I (capital letter I) is specified for the compilation phase, then IPA compilation steps are performed. When phase code I is specified for the link-editing phase, or when the first link-editing phase (code I) option is I or i, then an additional IPA Link step is performed prior to the prelink and link-edit steps.

With conventional compilation and link-editing, the object code generation takes place during the compilation phase. With IPA compilation and link-editing, the object code generation takes place during the link-editing phase. Therefore, you might need to request listing information about the program (such as with the -V option) during the link-editing phase.

Unlike the other phase codes, phase code I does not require that any additional options be specified. If they are, they should be specified for both the compilation and link-editing phases.

No additional preparation needs to be done in order to use IPA.
To create the executable `myIPApgm` using `c89` with some existing source program `mypgm.c`, you could specify:

```
c89 -W I -o myIPApgm mypgm.c
```

When IPA is used with the `c++` command, and automatic template generation is being used, phase code I will control whether the automatic template generation compiles are done using IPA. If you do not specify phase code I, then regular compiles will be done. Specifying I as the first option of the link-editing phase option (that is, `W l, I`), causes the IPA linker to be used, but will not cause the IPA compiler to be used for automatic template generation unless phase code I (that is, `W I`) is also specified.

The IPA Profile-Directed Feedback (PDF) option tunes optimizations, where results from sample program execution are used to improve optimization near conditional branches and in frequently executed code sections. The profiling information is placed in the file specified by the PDFNAME(filename) suboption. If PDFNAME(filename) is not specified, the default name of the file containing profile information is PDF.

**Note:** When using the `c89` command to invoke the compiler for IPA Compile and IPA Link processing using a single command line, some compiler options are not passed to both the IPA Compile and IPA Link steps; for example, the LIST compiler option is not passed. If you want to pass it to both steps, you must use `-Wl, I, list` syntax so that it is also passed to the IPA Link step. The `xlc` utility passes all compiler options to both the IPA Compile and IPA Link step.

**LP64**

The LP64 option instructs the compiler to generate AMODE 64 code utilizing the z/Architecture® 64-bit instructions.

To compile 64-bit code, specify the z/OS XL C/C++ LP64 compiler option.

The following example shows how to compile and bind using the LP64 option:

```
c89 -o -W c, LP64 -Wl, LP64 file.c
```

**XPLINK (Extra Performance Linkage)**

z/OS XPLINK provides improved performance for many C/C++ programs. The XPLINK compiler option instructs the z/OS XL C/C++ compiler to generate high performance linkage for subroutine calls. It does so primarily by making subroutine calls as fast and efficient as possible, by reducing linkage overhead, and by passing function call parameters in registers. Furthermore, it reduces the data size by eliminating unused information from function control blocks.

An XPLINK-compiled program is implicitly a DLL-compiled program (the C/C++ DLL compiler option need not be specified along with the XPLINK option). XPLINK improves performance when crossing function boundaries, even across compilation units, since XPLINK uses a more efficient linkage mechanism.

For more information about the z/OS XL C/C++ XPLINK compiler option, refer to *z/OS XL C/C++ User’s Guide*. For more information about Extra Performance Linkage, refer to *z/OS Language Environment Programming Guide*.

To use XPLINK, you must both compile and link-edit the program for XPLINK. All C and C++ source files must be compiled XPLINK, as you cannot statically link together XPLINK and non-XPLINK C and C++ object files (with the exception of non-XPLINK "OS" linkage). You can however mix XPLINK and non-XPLINK executables across DLL and fetch() boundaries.

To compile a program as XPLINK, specify the z/OS XL C/C++ XPLINK compiler option. If there are any exported symbols in the executable and you want to produce a definition sidedeck, specify the DLL link editor option. When XPLINK is specified in the link-editing step, different link-edit libraries will be used.

Following is an example of compiling and link-editing an XPLINK application in one command:

```
c89 -o outxpl -W c, XPLINK -W l, XPLINK, dll file.c
```
In order to execute an XPLINK program, the SCEERUN2 as well as the SCEERUN data set must be in the MVS program search order (see the prefix _PLIB_PREFIX environment variable).

You cannot use -W to override the compiler options that correspond to c89 flag options, with the following exceptions:

- Listing options (corresponding to -V)
- Inlining options (corresponding to -O, -2, and -3)
- Symbolic options (corresponding to -s and -g); symbolic options can be overridden only when neither -s nor -g is specified.

Notes:

1. Most compiler, prelinker, and IPA linker options have a positive and negative form. The negative form is the positive with a prepended NO (as in XREF and NOXREF).
2. The compiler #pragma options directives as well as any other #pragma directives which are overridden by compiler options, will have no effect in source code compiled by the c89/cc/c++ command.
3. Link editor options must be specified in the name=value format. Both the option name and value must be spelled out in full. If you do not specify a value, a default value of YES is used, except for the following options, which if specified without a value, have the default values shown here:

   **ALIASES**
   
   ALIASES=ALL

   **DYNAM**
   
   DYNAM=DLL

   **LET**
   
   LET=8

   **LIST**
   
   LIST=NOIMPORT

Notes:

a. The binder default is COMPAT=MIN. For downward compatibility (when -Wc,'target(release)' is used), COMPAT should also be used (for example, -Wl,compat=min, or the specific program object format level supported by the target deployment system, if it is known). For more information, see the Downward Compatibility section of z/OS XL C/C++ User's Guide.

b. As of z/OS V1R8, the default for the COMPAT option is no longer emitted. In prior releases, the default was COMPAT=CURRENT.

c. References throughout this document to the link editor are generic references. The c89/cc/c++ command specifically uses the Program Management binder for this function.

4. The z/OS XL C/C++ compiler is described in z/OS XL C/C++ User's Guide. Related information about the z/OS XL C/C++ runtime library, including information about DLL and IPA support, is described in z/OS XL C/C++ Programming Guide. Related information about the C and C++ languages, including information about compiler directives, is described in z/OS XL C/C++ Language Reference.

5. Since some compiler options are only used by z/OS XL C and some compiler options are only used by z/OS XL C++, you may get warning messages and a compiler return code of 4, if you use this option and compile both C and C++ source programs in the same c++ command invocation.

6. The prelinker is described in z/OS XL C/C++ User's Guide.

7. The z/OS XL C/C++ User's Guide also describes z/OS XL C/C++ compiler options. Any messages produced by it (CCN messages) are documented in z/OS XL C/C++ Messages.

8. You may see runtime messages (CEE or EDC) in executing your applications. These messages are described in z/OS Language Environment Debugging Guide.
9. The link editor (the Program Management binder) is described in z/OS MVS Program Management: User’s Guide and Reference. The Program Management binder messages are described in z/OS MVS System Messages, Vol 8 (IEF-IGD).

Operands

The c89/cc/c++ command generally recognizes their file operand types by file suffixes. The suffixes shown here represent the default values used by the c89/cc/c++ command. See Environment variables for information about changing the suffixes to be used.

Unlike the c89 and c++ commands, which report an error if given an operand with an unrecognized suffix, the cc command determines that it is either an object file or a library based on the file itself. This behavior is in accordance with the environment variable prefix_EXTRA_ARGS.

file.a

Specifies the name of an archive file, as produced by the ar command, to be used during the link-editing phase. You can specify an MVS data set name, by preceding the file name with double slash (/ /), in which case the last qualifier of the data set name must be LIB. The data set specified must be a C370LIB object library or a load library. See the description of the -l name operand for more information about using data sets as libraries.

file.C

Specifies the name of a C++ source file to be compiled. You can specify an MVS data set name by preceding the file name with double slash (/ /), in which case the last qualifier of the data set name must be CXX. This operand is only supported by the c++ command.

file.c

Specifies the name of a C source file to be compiled. You can specify an MVS data set name by preceding the file name with double slash (/ /), in which case the last qualifier of the data set name must be C. (The conventions formerly used by c89 for specifying data set names are still supported. See the environment variables prefix_OSUFFIX_HOSTRULE and prefix_OSUFFIX_HOSTQUAL for more information.)

file.I

Specifies the name of a IPA linker output file that is produced during the c89/cc/c++ link-editing phase, when the -W option is specified with phase code I. IPA is further described in the Options topic. By default the IPA linker output file is written to a temporary file. To have the IPA linker output file written to a permanent file, see the environment variable prefix_TMPS under Environment variables.

When an IPA linker output file is produced by the c89/cc/c++ command, the default name is based upon the output file name. See the -o option description in the Options topic, for information on the name of the output file.

If the output file is named a.out, then the IPA linker output file is named a.I, and is always in the working directory. If the output file is named //a.load, then the IPA linker output file is named //a.IPA. If the output file specified already has a suffix, that suffix is replaced. Otherwise the suffix is appended. This file may also be specified on the command line, in which case it is used as a file to be link-edited.

file.i

Specifies the name of a preprocessed C or C++ source file to be compiled. You can specify an MVS data set name, by preceding the file name with double slash (/ /), in which case the last qualifier of the data set name must be CEX.

When using the c++ command, this source file is recognized as a C++ source file, otherwise it is recognized as a C source file. The c++ command can be made to distinguish between the two. For more information, see the environment variables prefix_IXXSUFFIX and prefix_IXXSUFFIX_HOST.

file.o

Specifies the name of a C, C++, or assembler object file, produced by the c89/cc/c++ command, to be link-edited.
When an object file is produced by the `c89/cc/c++` command, the default name is based upon the source file. If the source file is named `file.c`, then the object file is named `file.o`, and is always in the working directory. If the source file were a data set named `//file.C`, then the object file is named `//file.OBJ`.

If the data set specified as an object file has undefined (U) record format, then it is assumed to be a load module. Load modules are not processed by the prelinker.

You can specify an MVS data set name to be link-edited, by preceding the file name with double slash (`//`), in which case the last qualifier of the data set name must be `OBJ`.

If a partitioned data set is specified, more than one member name may be specified by separating each with a comma (`,`):

```
c89 //file.OBJ(mem1,mem2,mem3)
```

`file.p`

Specifies the name of a prelinker composite object file produced during the `c89/cc/c++` link-editing phase. By default, the composite object file is written to a temporary file. To have the composite object file written to a permanent file, see the environment variable `prefix_TMPS` under Environment variables.

When a composite object file is produced by the `c89/cc/c++` command, the default name is based upon the output file name. See the `-o` option description in the Options topic, for information on the name of the output file.

If the output file is named `a.out`, then the composite object file is named `a.p`, and is always in the working directory. If the output file is named `//a.load`, then the composite object file is named `//a.CPOBJ`. If the output file specified already has a suffix, that suffix is replaced. Otherwise the suffix is appended. This file may also be specified on the command line, in which case it is used as a file to be link-edited.

`file.s`

Specifies the name of an assembler source file to be assembled. You can specify an MVS data set name, by preceding the file name with double slash (`//`), in which case the last qualifier of the data set name must be `ASM`.

`file.x`

Specifies the name of a definition sidedeck produced during the `c89/cc/c++` link-editing phase when creating a DLL (Dynamic Link Library), and used during the link-editing phase of an application using the DLL. DLLs are further described under the `-W` option.

When a definition sidedeck is produced by the `c89/cc/c++` command, the default name is based upon the output file name. See the `-o` option description in the Options topic, for information on the name of the output file.

If the output file is named `a.dll`, then the definition sidedeck is named `a.x`, and is always in the working directory. If the output file is named `//a.DLL`, then the definition sidedeck is named `//a.EXP`. If the output file specified already has a suffix, that suffix is replaced. Otherwise the suffix is appended.

You can specify an MVS data set name to be link-edited, by preceding the file name with double slash (`//`), in which case the last qualifier of the data set name must be `EXP`.

If a partitioned data set is specified, more than one member name may be specified by separating each with a comma (`,`):

```
c89 //file.EXP(mem1,mem2,mem3)
```

`-l` `name`

Specifies the name of an archive library. The `c89/cc/c++` command searches for the file `lib libname.a` in the directories specified on the `-L` option and then in the usual places. The first occurrence of the archive library is used. For a description of the usual places, see Usage Note 7.
You can also specify an MVS data set; you must specify the full data set name, because there are no rules for searching library directories.

The data set specified must be a C370LIB object library or a load library. If a data set specified as a library has undefined (U) record format, then it is assumed to be a load library. For more information about how load libraries are searched, see Usage Note 7.

**Environment variables**

You can use environment variables to specify necessary system and operational information to the c89/cc/c++ command. When a particular environment variable is not set, the c89/cc/c++ command uses the default shown. For information about the JCL parameters used in these environment variables, see z/OS MVS JCL User’s Guide.

Each environment variable has a prefix (shown in italics) that should be replaced by one of the following strings, depending on the command name used:

- _CC
- _CXX
- _C89

This means that to specify the cc environment variables, the name that is shown must be prefixed with _CC (for example, _CC_ACCEPTABLE_RC). To specify c89 environment variables, the name that is shown must be prefixed with _C89 (for example, _C89_ACCEPTABLE_RC). To specify c++/cxx environment variables, the name that is shown must be prefixed with _CXX (for example, _CXX_ACCEPTABLE_RC). The following examples show how to code one or more MVS data sets:

- export _CXX_LSYSLIB=CEE.SCEELKED
- export _CXX_LSYSLIB=CEE.SCEELKED:CEE.SCEELKEX

**Notes:**

1. For most environment variables, you can use all three prefixes (_CC, _CXX, _C89). In the list of environment variables that follows, you should assume that all three prefixes can be used unless otherwise indicated.

2. The c89/cc/c++ command can accept parameters only in the syntax indicated here. A null value indicates that the c89/cc/c++ command should omit the corresponding parameters during dynamic allocation. Numbers in parentheses following the environment variable name correspond to usage notes, which indicate specific usage information for the environment variable.

3. The _CCN_IPA_WORK_SPACE environment variable does not include a prefix.

**_CCN_32_RUNOPTS**

- Specifies Language Environment runtime options that apply to the environment in which the 32-bit compiler components are running.

**_CCN_64_RUNOPTS**

- Specifies Language Environment runtime options that apply to the environment in which the 64-bit compiler components are running.

**_CCN_IPA_WORK_SPACE**

- The SPACE parameter used by the z/OS XL C/C++ compiler for the unnamed temporary work data set related to IPALINK.

When _CCN_IPA_WORK_SPACE is not specified, the default is to use the settings from prefix_WORK_SPACE. In this case, prefix_WORK_SPACE must be set large enough for the potentially large work files that can be generated by IPALINK. If _CCN_IPA_WORK_SPACE is used, prefix_WORK_SPACE can be tuned for the typically smaller work files generated by the rest of the compiler.
The maximum allowed return code (result) of any step (compile, assemble, IPA Link, prelink, or link-edit). If the result is between zero and this value (inclusive), then it is treated internally by the `c89/cc/c++` command exactly as if it were a zero result, except that message FSUM3065 is also issued. The default value is 4.

When used under the `c89/cc/c++` command, the prelinker by default returns at least a 4 when there are duplicate symbols or unresolved writable static symbols (but not for other unresolved references). The link editor returns at least a 4 when there are duplicate symbols, and at least an 8 when there are unresolved references and automatic library call was used.

The suffix by which the `c89/cc/c++` command recognizes an archive file. This environment variable does not affect the treatment of archive libraries specified as `-l` operands, which are always prefixed with `lib` and suffixed with `.a`. The default value is `a`.

The suffix by which the `c89/cc/c++` command recognizes a library data set. This environment variable does not affect the treatment of data set libraries specified as `-l` operands, which are always used exactly as specified. The default value is `LIB`.

Controls how the `c89/cc/c++` command does parsing. The default behavior of the `c89/cc/c++` command is to expect all options to precede all operands. Setting this variable allows compatibility with historical implementations (other `cc` commands). When set to 1, the `c89/cc/c++` command operates as follows:

- Options and operands can be interspersed.
- The double dash (`--`) is ignored.

Setting this variable to 0 results in the default behavior. The default value is 0.

The prefix for the following named data sets used during the compilation phase and execution of your C++ application.

To be used, the following data sets must be cataloged:

- The data sets `${prefix_CLASSLIB_PREFIX}.SCLBH.+` contain the z/OS XL C++ Class Library include (header) files.
- The data set `${prefix_CLASSLIB_PREFIX}.SCLBSID` contains the z/OS XL C++ Class Library definition side-decks.

The following data sets are also used:

The data sets `${prefix_CLASSLIB_PREFIX}.SCLBDLL` and `${prefix_CLASSLIB_PREFIX}.SCLBDLL2` contain the z/OS XL C++ Class Library DLLs and messages.

The preceding data sets contain MVS programs that are invoked during the execution of a C++ application built by the `c++` command. To be executed correctly, these data sets must be made part of the MVS search order. Regardless of the setting of this or any other `c++` environment variable, the `c++` command does not affect the MVS search order. These data sets are listed here for information only, to assist in identifying the correct data sets to be added to the MVS program search order.

The default value is the value of the environment variable: `_CXX_CLIB_PREFIX`

The prefix `CLASSLIB_PREFIX` environment variable applies only to the `c++` and `cxx` command names. `_CXX` is the only valid prefix.

The version of the C++ Class Library to be invoked by the `c++` command. The setting of this variable allows `c++` to control which C++ Class Library named data sets are used during the `c++` processing phases. It also sets default values for other environment variables.

The format of this variable is the same as the result of the Language Environment C/C++ runtime library function `_librel()`. See `z/OS XL C/C++ Runtime Library Reference` for a description of the
_librel() function. The default value is the same as the value for the _CVERSION environment variable. If _CVERSION is not set, then the default value will be the result of the C/C++ runtime library _librel() function.

The prefix_CLASSVERSION environment variable applies only to the c++ and cxx command names. _CXX is the only valid prefix.

(prefix_CLIB_PREFIX (Usage Note 17))
The prefix for the following named data set used during the compilation phase.

The data set $(prefix_CLIB_PREFIX).SCCNCMP contains the compiler programs called by the c89/cc/c++ command.

The preceding data set contains MVS programs that are invoked during the execution of the c89/cc/c++ command and during the execution of a C/C++ application built by the c89/cc/c++ command. To be executed correctly, the data set must be made part of the MVS search order.

Regardless of the setting of this or any other c89/cc/c++ environment variable, c89/cc/c++ does not affect the MVS search order. The data set is listed here for information only, to assist in identifying the correct data set to be added to the MVS program search order.

The following data set is also used:

The data set $(prefix_CLIB_PREFIX).SCCNOBJ contains object files required to instrument the code for profile-driven feedback optimization.

The default value is CBC.

(prefix_CMEMORY)
A suggestion as to the use of compiler C/C+ Runtime Library memory files. When set to 0, the c89/cc/c++ command will prefer to use the compiler NOMEMORY option. When set to 1, c89/cc/c++ will prefer to use the compiler MEMORY option. When set to 1, and if the compiler MEMORY option can be used, c89/cc/c++ needs not allocate data sets for the corresponding work files. In this case it is the responsibility of the user to not override the compiler options (using the -W option) with the NOMEMORY option or any other compiler option which implies the NOMEMORY option.

The default value is 1.

(prefix_CMGS (Usage Note 14))
The Language Environment national language name used by the compiler program. A null value will cause the default Language Environment NATLANG runtime name to be used, and a non-null value must be a valid Language Environment NATLANG runtime option name (Language Environment runtime options are described in z/OS Language Environment Programming Guide). The default value is " " (null).

(prefix_CNAME (Usage Note 14))
The name of the compiler program called by the c89/cc/c++ command. It must be a member of a data set in the search order used for MVS programs. The default value is CCNDVR. If the c89/cc/c++ command is being used with prefix_CVERSION set to a release prior to z/OS V1R2, the default value will be CBCDRVR.

(prefix_CSUFFIX (Usage Note 15))
The suffix by which the c89/cc/c++ command recognizes a C source file. The default value is c.

(prefix_CSUFFIX_HOST (Usage Note 15))
The suffix by which the c89/cc/c++ command recognizes a C source data set. The default value is C.

(prefix_CSYSLIB (Usage Note 4, Usage Note 16))
The system library data set concatenation to be used to resolve #include directives during compilation.

Normally #include directives are resolved using all the information specified including the directory name. When the c89/cc/c++ command can determine that the directory information can be used, such as when the Language Environment include (header) files are installed in the default location (in accordance with prefix_INCDIRS), then the default concatenation is " " (null).
When the c89/cc/c++ command cannot determine that the directory information can be used, then the default concatenation is:

```
"${prefix}_PLIB_PREFIX}.SCEE.H"
"${prefix}_PLIB_PREFIX}.SCEEH.SYS.H"
"${prefix}_PLIB_PREFIX}.SCEEH.ARPA.H"
"${prefix}_PLIB_PREFIX}.SCEEH.NET.H"
"${prefix}_PLIB_PREFIX}.SCEEH.NETINET.H"
```

When this variable is a null value, then no allocation is done for compiler system library data sets. In this case, the use of //DD:SYSLIB on the -I option and the #include directive will be unsuccessful. Unless there is a dependency on the use of //DD:SYSLIB, it is recommended that for improved performance this variable be allowed to default to a null value.

**prefix_CVERSION**

The version of the z/OS XL C/C++ compiler to be invoked by the c89/cc/c++ command. The setting of this variable allows the c89/cc/c++ command to control which z/OS XL C/C++ compiler program is invoked. It also sets default values for other environment variables.

The format of this variable is the same as the result of the Language Environment C/C++ runtime library function _librel(). See z/OS XL C/C++ Runtime Library Reference for a description of the _librel() function. The default value is the result of the C/C++ runtime library _librel() function.

**prefix_CXXSUFFIX (Usage Note 15)**

The suffix by which the c++ command recognizes a C++ source file. The default value is C. This environment variable is only supported by the c++ and cxx command names. _CXX is the only valid prefix.

**prefix_CXXSUFFIX_HOST (Usage Note 15)**

The suffix by which the c++ command recognizes a C++ source data set. The default value is CXX. This environment variable is only supported by the c++ and cxx command names. _CXX is the only valid prefix.

**prefix_DAMPLEVEL**

The minimum severity level of dynamic allocation messages returned by dynamic allocation message processing. Messages with severity greater than or equal to this number are written to stderr. However, if the number is out of the range shown here (that is, less than 0 or greater than 8), then the c89/cc/c++ dynamic allocation message processing is disabled. The default value is 4. Valid values are as follows:

- 0: Informational
- 1–4: Warning
- 5–8: Severe

**prefix_DAMPNAME (Usage Note 14)**

The name of the dynamic allocation message processing program called by the c89/cc/c++ command. It must be a member of a data set in the search order used for MVS programs. The default dynamic allocation message processing program is described in z/OS MVS Programming: Authorized Assembler Services Guide. The default value is IEFDB476.

**prefix_DCBF2008 (Usage Note 21)**

The DCB parameters used by the c89/cc/c++ command for data sets with the attributes of record format fixed unblocked and minimum block size of 2008. The block size must be in multiples of 8, and the maximum depends on the phase in which it is used but can be at least 5100. The default value is (RECFM=F,LRECL=4088,BLKSIZE=4088).

**prefix_DCBU (Usage Note 21)**

The DCB parameters used by the c89/cc/c++ command for data sets with the attributes of record format undefined and data set organization partitioned. This DCB is used by the c89/cc/c++ command for the output file when it is to be written to a data set. The default value is (RECFM=U,LRECL=0,BLKSIZE=6144,DSORG=PO).
The DCB parameters used by the `c89/cc/c++` command for data sets with the attributes of record format fixed blocked and logical record length 121, for data sets whose records may contain machine control characters. The default value is (RECFM=FBM,LRECL=121,BLKSIZE=3630).

The DCB parameters used by the `c89/cc/c++` command for data sets with the attributes of record format fixed blocked and logical record length 133, for data sets whose records may contain machine control characters. The default value is (RECFM=FBM,LRECL=133,BLKSIZE=3990).

The DCB parameters used by the `c89/cc/c++` command for data sets with the attributes of record format variable blocked and logical record length 137. The default value is (RECFM=VB,LRECL=137,BLKSIZE=882).

The DCB parameters used by the `c89/cc/c++` command for data sets with the attributes of record format variable blocked and logical record length 137, for data sets whose records may contain ISO/ANSI control characters. The default value is (RECFM=VB,LRECL=137,BLKSIZE=882).

The DCB parameters used by the `c89/cc/c++` command for data sets with the attributes of record format fixed blocked and logical record length 3200. The default value is (RECFM=FB,LRECL=3200,BLKSIZE=12800).

The DCB parameters used by the `c89/cc/c++` command for data sets with the attributes of record format fixed blocked and logical record length 80. This value is also used when the `c89/cc/c++` command allocates a new data set for an object file. The default value is (RECFM=FB,LRECL=80,BLKSIZE=3200).

This variable is used to determine to which debug format (DWARF or ISD) the `-g` flag is translated. If `_DEBUG_FORMAT` is set to DWARF, then `-g` is translated to DEBUG(FORMAT(DWARF)). If `_DEBUG_FORMAT` is set to ISD, then `-g` is translated to TEST. The default value is DWARF.

Note: This environment variable only applies to 31-bit compiles.

This variable controls whether the output of the `-E` option will include `#line` directives. `#line` directives provide information about the source file names and line numbers from which the preprocessed source came. The preprocessor only inserts `#line` directives where it is necessary. When set to 1, the output of the `c89/cc/c++` `-E` option will include `#line` directives where necessary. When set to 0, the output will not include any `#line` directives. The default value is 0.

The setting of this variable controls whether the `c89/cc/c++` command treats a file operand with an unrecognized suffix as an error, or attempts to process it. When the `c++` command `-+` option is specified, all suffixes which otherwise would be unrecognized are instead recognized as C++ source, effectively disabling this environment variable. See “Options” on page 80 for information on the `-+` option.

When set to 0, the `c89/cc/c++` command treats such a file as an error and the command will be unsuccessful, because the suffix will not be recognized.

When set to 1, the `c89/cc/c++` command treats such a file as either an object file or a library, depending on the file itself. If it is neither an object file nor a library then the command will be unsuccessful, because the link-editing phase will be unable to process it. The default value for the `c89` and `c++` commands is 0. The default value for the `cc` command is 1.

The system definition side deck list that is used to resolve symbols during the IPA Link step of the link-editing phase when using LP64 (see the description of LP64 in “Options” on page 80). The default value is whatever `prefix_IL6SYSIX` is set to or defaults to.
prefix_IL6SYSLIB (Usage Note 7, Usage Note 16)
The system library data set list that is used to resolve symbols during the IPA Link step of the link-
editing phase when using LP64 (see the description of LP64 in “Options” on page 80). The default value is whatever `prefix_IL6SYSLIB` is set to or defaults to.

prefix_ILCTL (Usage Note 14)
The name of the control file used by the IPA linker program. By default the control file is not used, so the `-W` option must be specified to enable its use, as in:

```c89  -WI,control ...
```

The default value is `ipa.ctl`.

prefix_ILMSGS (Usage Note 14)
The name of the message data set member, or the Language Environment national language name, used by the IPA linker program. The default value is whatever `prefix_ILMSGS` is. So if `prefix_ILMSGS` is set or defaults to "" (null), the default value is "" (null).

prefix_ILNAME (Usage Note 14)
The name of the IPA linker program called by the `c89/cc` command. It must be a member of a data set in the search order used for MVS programs. The default value is whatever `prefix_ILNAME` is. So if `prefix_ILNAME` is set or defaults to `CCNDRVR` the default value is `CCNDRVR`.

prefix_ILSUFFIX (Usage Note 15)
The suffix that the `c89/cc` command uses when creating an IPA linker output file. The default value is I.

prefix_ILSUFFIX_HOST (Usage Note 15)
The suffix that the `c89/cc` command uses when creating an IPA linker output data set. The default value is IPA.

prefix_ILSYSLIB (Usage Note 7, Usage Note 16)
The system library data set list to be used to resolve symbols during the IPA Link step of the link-
editing phase of non-XPLINK programs. The default value is whatever `prefix_ILSYSLIB` is set or defaults to, followed by whatever `prefix_ILSYSIX` is set or defaults to.

prefix_ILSYSIX (Usage Note 7, Usage Note 16)
The system definition sidedeck list to be used to resolve symbols during the IPA Link step of the link-
editing phase in non-XPLINK programs. The default value is whatever `prefix_ILSYSIX` is set or defaults to.

prefix_ILXSYSLIB (Usage Note 7, Usage Note 16)
The system library data set list to be used to resolve symbols during the IPA Link step of the link-
editing phase when using XPLINK (see XPLINK (Extra Performance Linkage) in “Options” on page 80). The default value is whatever `prefix_ILXSYSLIB` is set or defaults to.

prefix_ILXSYSIX (Usage Note 7, Usage Note 16)
The system definition sidedeck list to be used to resolve symbols during the IPA Link step of the link-
editing phase when using XPLINK (see XPLINK (Extra Performance Linkage) in “Options” on page 80). The default value is whatever `prefix_ILXSYSIX` is set or defaults to.

prefix_INCDIRS (Usage Note 22)
The directories used by the `c89/cc/c++` command as a default place to search for include files during compilation (before searching `prefix_INCLIBS` and `prefix_CSYSLIB`). If the `c++` command is not being used the default value is `/usr/include`. If the `c++` command is being used the default value is `/usr/include /usr/lpp/cbcclib/include`.

prefix_INCLIBS (Usage Note 22)
The directories used by the `c89/cc/c++` command as a default place to search for include files during compilation (after searching `prefix_INCDIRS` and before searching `prefix_CSYSLIB`). The default value depends on whether or not the `c++` command is being used. If the `c++` command is not being used the default value is `/#{$(prefix_PLIB_PREFIX).SCEEH.+}'

If the `c++` command is being used, the default value is `/#{$(prefix_PLIB_PREFIX).SCEEH.+}' `/#{$ (prefix_CLIB_PREFIX).SCLBH.+}'
**prefix_ISUFFIX (Usage Note 15)**
The suffix by which the `c89/cc/c++` command recognizes a preprocessed C source file. The default value is `i`.

**prefix_ISUFFIX_HOST (Usage Note 15)**
The suffix by which the `c89/cc/c++` command recognizes a preprocessed (expanded) C source data set. The default value is `CEX`.

**prefix_IXXSUFFIX (Usage Note 15)**
The suffix by which the `c++` command recognizes a preprocessed C++ source file. The default value is `i`. This environment variable is only supported by the `c++` and `cxx` command names. `_CXX` is the only valid prefix.

**prefix_IXXSUFFIX_HOST (Usage Note 15)**
The suffix by which the `c++` command recognizes a preprocessed (expanded) C++ source data set. The default value is `CEX`. This environment variable is only supported by the `c++` and `cxx` command names. `_CXX` is the valid prefix.

**prefix_L6SYSIX (Usage Note 7, Usage Note 16)**
The system definition sidedeck list that resolves symbols during the link-editing phase when using LP64 (see the description of LP64 in “Options” on page 80). A definition sidedeck contains link-editing phase IMPORT control statements, which name symbols that are exported by a DLL. The default value depends on whether or not the `c++` command is used. If `c++` is not used, the default value is: `${prefix_PLIB_PREFIX}.SCEELIB(CELQS003)`. If `c++` is used, the default value is the list concatenation:

```
"${prefix_PLIB_PREFIX}.SCEELIB(CELQS003,CELQSCPP,C64)"
"${prefix_CLASSLIB_PREFIX}.SCLBSID(I0S64)"
```

**prefix_L6SYSLIB (Usage Note 7, Usage Note 16)**
The system library data set concatenation that is used to resolve symbols during the link-editing step when using LP64 (see the description of LP64 in “Options” on page 80). The default value is the concatenation:

```
"${prefix_PLIB_PREFIX}.SCEEBND2"
"${prefix_SLIB_PREFIX}.CSSLIB"
```

**prefix_LIBDIRS (Usage Note 22)**
The directories used by the `c89/cc/c++` command as the default place to search for archive libraries which are specified using the `-l` operand. The default value is `/lib` `/usr/lib`.

**prefix_LSYSLIB (Usage Note 7, Usage Note 16)**
The system library data set concatenation to be used when resolving symbols during the IPA Link step and the link-edit phase of the non-XPLINK link-editing phase. The `prefix_PSYSLIB` libraries always precede the `prefix_LSYSLIB` libraries when resolving symbols in the link-editing phase. The default value is the concatenation:

```
"${prefix_PLIB_PREFIX}.SCEELKEX"
"${prefix_PLIB_PREFIX}.SCEELKD"
"${prefix_SLIB_PREFIX}.CSSLIB"
```

**prefix_LXSYSLIB (Usage Note 7, Usage Note 16)**
The system library data set concatenation to be used when resolving symbols during the IPA Link step and the link-edit phase when using XPLINK (see XPLINK (Extra Performance Linkage) in “Options” on page 80). The default value is the concatenation:

```
"${prefix_PLIB_PREFIX}.SCEEBND2"
"${prefix_SLIB_PREFIX}.CSSLIB"
```

**prefix_LXSYSIX (Usage Note 7, Usage Note 16)**
The system definition sidedeck list to be used when resolving symbols during the link-editing phase when using XPLINK (see XPLINK (Extra Performance Linkage) in “Options” on page 80). A definition sidedeck contains link-editing phase IMPORT control statements naming symbols which are exported by a DLL. The default value depends on whether or not the `c++` command is being used. For 32-bit
objects, if c++ is not being used, the default value is the list "${prefix_PLIB_PREFIX}.SCEELIB(CELHS003,CELHS001). For 32-bit objects, if c++ is being used with
prefix_PVERSION and prefix_CLASSVERSION defaulted to the current z/OS release, the default value is the list concatenation:

"${prefix_PLIB_PREFIX}.SCEELIB(CELHS003,CELHS001,CELHSCPP,C128)"
"${prefix_CLASSLIB_PREFIX}.SCLBSID(IOSTREAM,COMPLEX)"

For 32-bit objects, if the c++ command is being used with prefix_PVERSION and prefix_CLASSVERSION set to a release prior to z/OS V1R2 for a 32-bit program, the default value is
the list concatenation:

"${prefix_PLIB_PREFIX}.SCEELIB(CELHS003,CELHS001,CELHSCPP)"
"${prefix_CLASSLIB_PREFIX}.SCLBSID(ASCCOLL,COMPLEX,IOSTREAM)"

Note: For 64–bit objects, see prefix_L6SYSIX.

prefix_MEMORY
A suggestion as to the use of XL C/C++ runtime library memory files by the c89/cc/c++ command. When set to 0, the c89/cc/c++ command uses temporary data sets for all work files. When set to 1, the c89/cc/c++ command uses memory files for all work files that it can. The default value is 1.

prefix_NEW_DATACLAS (Usage Note 18)
The DATACLAS parameter used by the c89/cc/c++ command for any new data sets it creates. The default value is "" (null).

prefix_NEW_DSNTYPE (Usage Note 18, Usage Note 20)
The DSNTYPE parameter used by the c89/cc/c++ command for any new data sets it creates. The default value is "" (null).

prefix_NEW_MGMTCLAS (Usage Note 18)
The MGMTCLAS parameter used by the c89/cc/c++ command for any new data sets it creates. The default value is "" (null).

prefix_NEW_SPACE (Usage Note 18, Usage Note 19)
The SPACE parameters used by the c89/cc/c++ command for any new data sets it creates. A value for the number of directory blocks should always be specified. When allocating a sequential data set, the c89/cc/c++ command automatically ignores the specification. The default value is (10,10,10).

prefix_NEW_STORCLAS (Usage Note 18)
The STORCLAS parameter used by the c89/cc/c++ command for any new data sets it creates. The default value is "" (null).

prefix_NEW_UNIT (Usage Note 18)
The UNIT parameter used by the c89/cc,c++ command for any new data sets it creates. The default value is "" (null).

prefix_NOCMDOPTS (Usage Note 27)
Controls how the compiler processes the default options set by the c89 command. Setting this variable to 1, reverts the compiler to the behavior that was available prior to z/OS V1R5, when the compiler was unable to distinguish between the c89 defaults and the user-specified options. Setting this variable to 0, results in the default behavior where the compiler is now able to recognize c89 defaults. The default value is 0.

prefix_OPERANDS (Usage Note 22)
These operands are parsed as if they were specified after all other operands on the c89/cc/c++ command line. The default value is "" (null).

prefix_OPTIONS (Usage Note 22)
These options are parsed as if they were specified before all other options on the c89/cc/c++ command line. The default value is "" (null).

prefix_OSUFFIX (Usage Note 15)
The suffix by which the c89/cc/c++ command recognizes an object file. The default value is o.
**prefix_OSUFFIX_HOST (Usage Note 15)**

The suffix by which the `c89/cc/c++` command recognizes an object data set. The default value is OBJ.

**prefix_OSUFFIX_HOSTQUAL**

The data set name of an object data set is determined by the setting of this option. If it is set to 0, then the suffix `prefix_OSUFFIX_HOST` is appended to the source data set name to produce the object data set name. If it is set to 1, then the suffix `prefix_OSUFFIX_HOST` replaces the last qualifier of the source data set name to produce the object data set name (unless there is only a single qualifier, in which case the suffix is appended). The default value is 1.

**Note:** Earlier versions of the c89 utility always appended the suffix, which was inconsistent with the treatment of files in the hierarchical file system. It is recommended that any existing data sets be converted to use the new convention.

**prefix_OSUFFIX_HOSTRULE**

The way in which suffixes are used for host data sets is determined by the setting of this option. If it is set to 0, then data set types are determined by the rule described in the note which follows. If it is set to 1, then the data set types are determined by last qualifier of the data set (just as a suffix is used to determine the type of hierarchical file system file). Each host file type has an environment variable by which the default suffix can be modified. The default value is 1.

**Notes:**

1. Earlier versions of the c89 utility scanned the data set name to determine if it was an object data set. It searched for the string OBJ in the data set name, exclusive of the first qualifier and the member name. If it was found, the data set was determined to be an object data set, and otherwise it was determined to be a C source data set. It is recommended that any existing data sets be converted to use the new convention. Also, because the earlier convention only provided for recognition of C source files, assembler source cannot be processed if it is used.

2. The `c++` command does not support this environment variable, as the earlier convention would not provide for recognition of both C++ and C source files. Therefore regardless of its setting, the `c++` command always behaves as if it is set to 1.

**prefix_PLIB_PREFIX (Usage Note 17)**

The prefix for the following named data sets used during the compilation, assemble, and link-editing phases, and during the execution of your application.

To be used, the following data sets must be cataloged:

- The data sets `${prefix_PLIB_PREFIX}.SCEEH.+` contain the include (header) files for use with the runtime library functions (where + can be any of H, SYS.H, ARPA.H, NET.H, and NETINET.H).
- The data set `${prefix_PLIB_PREFIX}.SCEEMAC` contains COPY and MACRO files to be used during assembly.
- The data sets `${prefix_PLIB_PREFIX}.SCEEOBJ` and `${prefix_PLIB_PREFIX}.SCEECPP` contain runtime library bindings which exploit constructed reentrancy, used during the link-editing phase of non-XPLINK programs.
- The data set `${prefix_PLIB_PREFIX}.SCEELKEX` contains C runtime library bindings which exploit L-names used during the link-editing phase of non-XPLINK programs. For more information about L-names, see usage note 23.
- The data set `${prefix_PLIB_PREFIX}.SCEELKED` contains all other Language Environment runtime library bindings, used during the link-editing phase of non-XPLINK programs.
- The data set `${prefix_PLIB_PREFIX}.SCEEBND2` contains all static Language Environment runtime library bindings, used during the link-editing phase of XPLINK programs.
- The data set `${prefix_PLIB_PREFIX}.SCEELIB` contains the definition side-decks for the runtime library bindings, used during the link-editing phase of XPLINK programs.

The following data sets are also used:
• The data sets `{prefix_LIB_PREFIX}.SCEERUN` and `{prefix_LIB_PREFIX}.SCEERUN2` contains the runtime library programs.

These data sets contain MVS programs that are invoked during the execution of the `c89/cc/c++` command and during the execution of a C/C++ application built by the `c89/cc/c++` command. To be executed correctly, these data sets must be made part of the MVS search order. Regardless of the setting of this or any other `c89/cc/c++` environment variable, the `c89/cc/c++` command does not affect the MVS program search order. These data sets are listed here for information only, to assist in identifying the correct data sets to be added to the MVS program search order.

The default value is CEE.

**prefix_MEMORY**
A suggestion as to the use of prelinker C/C++ Runtime Library memory files. When set to 0, the `c89/cc/c++` command uses the prelinker NOMEMORY option. When set to 1, the `c89/cc/c++` command uses the prelinker MEMORY option. The default value is 1.

**prefix_PMSGS (Usage Note 14)**
The name of the message data set used by the prelinker program. It must be a member of the cataloged data set `{prefix_LIB_PREFIX}.SCEEMSGP`. The default value is EDCPMSGE.

**prefix_PNAME (Usage Note 14)**
The name of the prelinker program called by the `c89/cc/c++` command. It must be a member of a data set in the search order used for MVS programs. The prelinker program is shipped as a member of the `{prefix_LIB_PREFIX}.SCEERUN` data set. The default value is EDCPRLK.

**prefix_PSUFFIX (Usage Note 15)**
The suffix that the `c89/cc/c++` command uses when creating a prelinker (composite object) output file. The default value is p.

**prefix_PSUFFIX_HOST (Usage Note 15)**
The suffix that the `c89/cc/c++` command uses when creating a prelinker (composite object) output data set. The default value is CPOBJ.

**prefix_PSYSLIB (Usage Note 16)**
The system library data set list to be used to resolve symbols during the non-XPLINK link-editing phase. The `prefix_PSYSLIB` libraries always precede the `prefix_LSYSLIB` libraries when resolving symbols in the link-editing phase. The default value depends on whether or not the `c++` command is being used. If the `c++` command is not being used, the default value is the list containing the single entry: `{prefix_LIB_PREFIX}.SCEEOBJ`.

If the `c++` command is being used with `prefix_PVERSION` and `prefix_CLASSVERSION` set to a release prior to z/OS V1R2, the default value is the list `{prefix_CLASSLIB_PREFIX}.SCLBSID(ASCCOLL,COMPLEX,IOSTREAM)`.

**prefix_PSYSLIB (Usage Note 16)**
The system library data set list to be used to resolve symbols during the non-XPLINK link-editing phase. The `prefix_PSYSLIB` libraries always precede the `prefix_LSYSLIB` libraries when resolving symbols in the link-editing phase. The default value depends on whether or not the `c++` command is being used. If the `c++` command is not being used, the default value is the list containing the single entry: `{prefix_LIB_PREFIX}.SCEEOBJ`.

If the `c++` command is being used, the default value is the list:

```
"${prefix_LIB_PREFIX}.SCEEOBJ:${prefix_LIB_PREFIX}.SCEECP"
```

**prefix_PVERSION (Usage Note 26)**
The version of the Language Environment runtime library to be used with the `c89/cc/c++` command. The setting of this variable allows `c89/cc/c++` to control which Language Environment named data sets are used during the `c89/cc/c++` processing phases. These named data sets include those
required for use of the C/C++ runtime library as well as the ISO C++ Library. It also sets default values for other environment variables.

The format of this variable is the same as the result of the Language Environment C/C++ runtime library function \_librel(). See z/OS XL C/C++ Runtime Library Reference for a description of the \_librel() function. The default value is the result of the C/C++ runtime library \_librel() function.

**prefix\_SLIB\_PREFIX (Usage Note 17)**

The prefix for the named data sets used by the link editor (CSSLIB) and the assembler system library data sets (MACLIB and MODGEN). The data set \$\{prefix\_SLIB\_PREFIX\}.CSSLIB contains the z/OS UNIX assembler callable services bindings. The data sets \$\{prefix\_SLIB\_PREFIX\}.MACLIB and \$\{prefix\_SLIB\_PREFIX\}.MODGEN contain COPY and MACRO files to be used during assembly. These data sets must be cataloged to be used. The default value is SYS1.

**prefix\_SNAME (Usage Note 14)**

The name of the assembler program called by the c89/cc/c++ command. It must be a member of a data set in the search order used for MVS programs. The default value is ASMA90.

**prefix\_SSUFFIX (Usage Note 15)**

The suffix by which the c89/cc/c++ command recognizes an assembler source file. The default value is s.

**prefix\_SSUFFIX\_HOST (Usage Note 15)**

The suffix by which the c89/cc/c++ command recognizes an assembler source data set. The default value is ASM.

**prefix\_SSYSLIB (Usage Note 16)**

The system library data set concatenation to be used to find COPY and MACRO files during assembly. The default concatenation is:

```
"\$\{prefix\_PLIB\_PREFIX\}.SCEEMAC"
"\$\{prefix\_SLIB\_PREFIX\}.MACLIB"
"\$\{prefix\_SLIB\_PREFIX\}.MODGEN"
```

**prefix\_STEPS**

The steps that are executed for the link-editing phase can be controlled with this variable. For example, the prelinker step can be enabled, so that the inputs normally destined for the link editor instead go into the prelinker, and then the output of the prelinker becomes the input to the link editor.

This variable allows the prelinker to be used in order to produce output which is compatible with previous releases of the c89/cc/c++ command. The prelinker is normally used by the c89/cc/c++ command when the output file is a data set which is not a PDSE (partitioned data set extended).

**Note:** The prelinker and XPLINK are incompatible. When using the link editor XPLINK option, the prelinker cannot be used. Thus, specifying the prelinker on this variable will have no effect.

The format of this variable is a set of binary switches which either enable (when turned on) or disable (when turned off) the corresponding step. Turning a switch on will not cause a step to be enabled if it was not already determined by the c89/cc/c++ command that any other conditions necessary for its use are satisfied. For example, the IPA Link step will not be executed unless the -W option is specified to enable the IPA linker. Enabling the IPA linker is described in "Options" on page 80.

Considering this variable to be a set of 32 switches, numbered left-to-right from 0 to 31, the steps corresponding to each of the switches are as follows:

**0-27**

Reserved

**28**

TEMPINC/IPATEMP

**29**

IPALINK

**30**

PRELINK
To override the default behavior of the c89/cc/c++ command and cause the prelinker step to be run (this is also the default when the output file is a data set which is not a PDSE), set this variable to:
0xffffffff or the equivalent, -1. The default value when the output file is a z/OS UNIX file or a PDSE data set is 0xffffffff or the equivalent, -3.

**Note:** The IPATEMP step is the IPA equivalent of the TEMPINC (automatic template generation) step, just as the IPACOMP step is the IPA equivalent of the COMPILE step. See the description of IPA under the -W option for more information.

**prefix_SUSRLIB (Usage Note 16)**
The user library data set concatenation to be used to find COPY and MACRO files during assembly (before searching prefix_SSYSLIB). The default value is "" (null).

**prefix_TMPS**
The use of temporary files by the c89/cc/c++ command can be controlled with this variable.

- The format of this variable is a set of binary switches which either cause a temporary file to be used (when turned on) or a permanent file to be used (when turned off) in the corresponding step.
- The correspondence of these switches to steps is the same as for the variable prefix_STEPS. Only the prelinker and IPA linker output can be captured using this variable.

To capture the prelinker output, set this variable to: 0xffffffffD or the equivalent, -3. The default value is 0xffffffff or the equivalent, -1.

**prefix_WORK_DATACLASS (Usage Note 18)**
The DATACLASS parameter used by the c89/cc/c++ command for unnamed temporary (work) data sets. The default value is "" (null).

**prefix_WORK_DSNTYPE (Usage Note 18, Usage Note 20)**
The DSNTYPE parameter used by the c89/cc/c++ command for unnamed temporary (work) data sets. The default value is "" (null).

**prefix_WORK_MGMTCLASS (Usage Note 18)**
The MGMTCLASS parameter used by the c89/cc/c++ command for unnamed temporary (work) data sets. The default value is "" (null).

**prefix_WORK_SPACE (Usage Note 18, Usage Note 19)**
The SPACE parameters used by the c89/cc/c++ command for unnamed temporary (work) data sets. You must set prefix_MEMORY to 0 for the prefix_WORK_SPACE settings to take effect. The default value is (32000,(30,30)). See also _CCN_IPA_WORK_SPACE.

**prefix_WORK_STORCLASS (Usage Note 18)**
The STORCLASS parameter used by the c89/cc/c++ command for unnamed temporary (work) data sets. The default value is "" (null).

**prefix_WORK_UNIT (Usage Note 18)**
The UNIT parameter used by the c89/cc/c++ command for unnamed temporary (work) data sets. The default value is SYSDA.

**prefix_XSUFFIX (Usage Note 15)**
The suffix by which the c89/cc/c++ command recognizes a definition sidedeck file of exported symbols. The default value is x.

**prefix_XSUFFIX_HOST (Usage Note 15)**
The suffix by which the c89/cc/c++ command recognizes a definition sidedeck data set of exported symbols. The default value is EXP.

**Files**

- libc.a
  - z/OS XL C/C++ runtime library function library (see Usage Note 7)
libm.a
C/C++ Runtime Library math function library (see Usage Note 7)

libl.a
lex function library

liby.a
yacc function library

/dev/fd0, /dev/fd1, ...
Character special files required by the c89/cc/c++ command. For installation information, see z/OS UNIX System Services Planning.

/usr/include
The usual place to search for include files (see Usage Note 4)

/lib
The usual place to search for runtime library bindings (see Usage Note 7)

/usr/lib
The usual place to search for runtime library bindings (see Usage Note 7)

Usage notes
1. To be able to specify an operand that begins with a dash (-), before specifying any other operands that do not, you must use the double dash (-- end-of-options delimiter. This also applies to the specification of the -l operand. (See the description of environment variable prefix_CCMODE for an alternate style of argument parsing.)

2. When invoking the c89/cc/c++ command from the shell, any option-arguments or operands specified that contain characters with special meaning to the shell must be escaped. For example, some -W option-arguments contain parentheses. Source files specified as PDS member names contain parentheses; if they are specified as fully qualified names, they contain single quotation marks.

   To escape these special characters, either enclose the option-argument or operand in double quotation marks, or precede each character with a backslash.

3. Some c89/cc/c++ behavior applies only to hierarchical files (and not to data sets).
   • If the compile or assemble is not successful, the corresponding object file (file.o) is always removed.
   • If the DLL option is passed to the link-editing phase, and afterwards the file.x file exists but has a size of zero, then that file is removed.

4. MVS data sets may be used as the usual place to resolve C and C++ #include directives during compilation.

   Such data sets are installed with the Language Environment runtime library. When it is allocated, searching for these include files can be specified on the -I option as //DD:SYSLIB. (See the description of environment variable prefix_CSYSLIB for information.

   When include files are MVS PDS members, z/OS XL C/C++ uses conversion rules to transform the include (header) file name on a #include preprocessor directive into a member name. If the "///dataset_prefix.+
   " syntax is not used for the MVS data set which is being searched for the include file, then this transformation strips any directory name on the #include directive, and then takes the first 8 or fewer characters up to the first dot (.).

   If the "///dataset_prefix.+
   " syntax is used for the MVS data set which is being searched for the include file, then this transformation uses any directory name on the #include directive, and the characters following the first dot (.), and substitutes the "+" of the data set being searched with these qualifiers.

   In both cases the data set name and member name are converted to uppercase and underscores (_) are changed to at signs (@).
If the include (header) files provided by the Language Environment runtime library are installed into the hierarchical file system in the default location (in accordance with the prefix_INCDIRS environment variable), then the compiler will use those files to resolve #include directives during compilation. The c89/cc/c++ command by default searches the directory /usr/include as the usual place, just before searching the data sets just described. See the description of environment variables prefix_CSYSLIB, prefix_INCDIRS, and prefix_INCLIBS for information about customizing the default directories to search.

5. Feature test macros control which symbols are made visible in a source file (typically a header file). The c89/cc/c++ command automatically defines the following feature test macros along with the errno macro, according to whether or not the cc command was invoked.

- Other than cc
  -D "errno=(*__errno())"
  -D _OPEN_DEFAULT=1
- cc
  -D "errno=(*__errno())"
  -D _OPEN_DEFAULT=0
  -D _NO_PROTO=1

The c89/cc/c++ command adds these macro definitions only after processing the command string. Therefore, you can override these macros by specifying -D or -U options for them on the command string.

6. The default LANGLVL and related compiler options are set according to whether the cc, c89, or c++ (cxx) command was invoked. These options affect various aspects of the compilation, such as z/OS XL C/C++ predefined macros, which are used like feature test macros to control which symbols are made visible in a source file (typically a header file), but are normally not defined or undefined except by this compiler option. They can also affect the language rules used by the compiler. For more information about the compiler options listed here, see z/OS XL C/C++ User's Guide. For more information about z/OS XL C/C++ predefined macros, see Examples of predefined macros in z/OS XL C/C++ Language Reference. The options are shown here in a syntax that the user can specify on the c89/cc/c++ command line to override them:

- c89 (also c++ (cxx) when using a C++ compiler older than z/OS v1r2)
  -W "c,langlvl(ansi),noupconv"
- c++ (cxx)
  -W "c,langlvl(extended,nolibext,nolonglong)"
- cc
  -W "c,langlvl(commonc),upconv"

7. By default the usual place for the -L option search is the /lib directory followed by the /usr/lib directory. See the description of environment variable prefix_LIBDIRS for information on customizing the default directories to search.

The archive libraries libc.a and libm.a exist as files in the usual place for consistency with other implementations. However, the runtime library bindings are not contained in them. Instead, MVS data sets installed with the Language Environment runtime library are used as the usual place to resolve runtime library bindings. In the final step of the link-editing phase, any MVS load libraries specified on the -l operand are searched in the order specified, followed by searching these data sets. See the prefix_PLIB_PREFIX description, as well as descriptions of the environment variables featured in the following list.

prefix_ILSYSLIB
prefix_ILSYSIX
prefix_LSYSLIB
prefix_PSYSIX
prefix_PSYSLIB
This list of environment variables affects the link-editing phase of the c89 utility, but only for non-XPLINK link-editing. See XPLINK (Extra Performance Linkage) in “Options” on page 80.

The following list of environment variables affects the link-editing phase of the c89 utility, but only for ILP32 XPLINK link-editing. See XPLINK (Extra Performance Linkage) in “Options” on page 80.

```
prefix_ILXSYSLIB
prefix_ILXSYSIX
prefix_LXSYSLIB
prefix_LXSYSIX
```

The following list of environment variables affects the link-editing phase of the c89 utility, but only for LP64 link-editing. See the description of LP64 in “Options” on page 80.

```
prefix_IL6SYSLIB
prefix_IL6SYSIX
prefix_L6SYSLIB
prefix_L6SYSIX
```

8. Because archive library files are searched when their names are encountered, the placement of `-l` operands and `file.o` operands is significant. You may have to specify a library multiple times on the command string, if subsequent specification of `file.o` files requires that additional symbols be resolved from that library.

9. When the prelinker is used during the link-editing phase, you cannot use as input to the `c89/cc/c++` command an executable file produced as output from a previous use of the `c89/cc/c++` command. The output of `c89/cc/c++` when the `-r` option is specified (which is not an executable file) may be used as input.

10. All MVS data sets used by the `c89/cc/c++` command must be cataloged (including the system data sets installed with the z/OS XL C/C++ compiler and the Language Environment runtime library).

11. The `c89/cc/c++` operation depends on the correct setting of their installation and configuration environment variables (see “Environment variables” on page 92). Also, they require that certain character special files are in the `/dev` directory. For additional installation and configuration information, see `z/OS UNIX System Services Planning`.

12. Normally, options and operands are processed in the order read (from left to right). Where there are conflicts, the last specification is used (such as with `-g` and `-s`). However, some c89 utility flag options will override others, regardless of the order in which they are specified. The option priorities, in order of highest to lowest, are as follows:

   `-v specified twice`
   The pseudo-JCL is printed only, but the effect of all the other options and operands as specified is reflected in the pseudo-JCL.

   `-E`
   Overrides `-0`, `-O`, `-1`, `-2`, `-3`, `-V`, `-c`, `-g` and `-s` (also ignores any `file.s` files).

   `-g`
   Overrides `-0`, `-O`, `-1`, `-2`, `-3`, and `-s`.

   `-s`
   Overrides `-g` (the last one specified is honored).

   `-O (zero), -O (capital letter O), -1, -2, -3, -V, -c`
   All are honored if not overridden. `-0`, `-O`, `-1`, `-2`, `-3` override each other (the last one specified is honored).

   **Note:** The preferred way for specifying optimization options, is `-O (capital letter O)` followed by a number; for example, `-O2`.

13. For options that have option-arguments, the meaning of multiple specifications of the options is as follows:
-D
All specifications are used. If the same name is specified on more than one -D option, only the first definition is used.

-e
The entry function used will be the one specified on the last -e option.

-I
All specifications are used. If the same directory is specified on more than one -I option, the directory is searched only the first time.

-L
All specifications are used. If the same directory is specified on more than one -L option, the directory is searched only the first time.

-o
The output file used will be the one specified on the last -o option.

-U
All specifications are used. The name is not defined, regardless of the position of this option relative to any -D option specifying the same name.

-u
All specifications are used. If a definition cannot be found for any of the functions specified, the link-editing phase will be unsuccessful.

-W
All specifications are used. All options specified for a phase are passed to it, as if they were concatenated together in the order specified.

14. The following environment variables can be at most eight characters in length. For those whose values specify the names of MVS programs to be executed, you can dynamically alter the search order used to find those programs by using the STEPLIB environment variable.

c89/cc/cc++ environment variables do not affect the MVS program search order. Also, for the c89/cc/cc++ command to work correctly, the setting of the STEPLIB environment variable should reflect the Language Environment library in use at the time that c89/cc/cc++ is invoked.

For more information on the STEPLIB environment variable, see z/OS UNIX System Services Planning. It is also described under the sh command. Note that the STEPLIB allocation in the pseudo-JCL produced by the -v verbose option is shown as a comment, and has no effect on the MVS program search order. Its appearance in the pseudo-JCL is strictly informational.

prefix_CMSGS
prefix_CNAME
prefix_DAMPNAME
prefix_ILCTL
prefix_ILNAME
prefix_ILMSGS
prefix_PMSGS
prefix_PNAME
prefix_SNAME

15. The following environment variables can be at most 15 characters in length. You should not specify any dots (.) when setting these environment variables since they would then never match their corresponding operands:

prefix_ASUFFIX
prefix_ASUFFIX_HOST
prefix_CSUFFIX
prefix_CSUFFIX_HOST
prefix_CXXSUFFIX
prefix_CXXSUFFIX_HOST
prefix_ISRUFFIX
16. The following environment variables are parsed as colon-delimited data set names, and represent a data set concatenation or a data set list. The maximum length of each specification is 1024 characters:

```
prefix_CSYSLIB
prefix_IL6SYSIX
prefix_IL6SYSLIB
prefix_ILSYSIX
prefix_ILSYSLIB
prefix_ILXSYSIX
prefix_ILXSYSLIB
prefix_L6SYSIX
prefix_L6SYSLIB
prefix_LSYSLIB
prefix_LXSYSIX
prefix_LXSYSLIB
prefix_PSYSIX
prefix_PSYSLIB
prefix_SSYSLIB
prefix_SUSRLIB
```

17. The following environment variables can be at most 44 characters in length:

```
prefix_CLASSLIB_PREFIX
prefix_CLIB_PREFIX
prefix_PLIB_PREFIX
prefix_SLIB_PREFIX
```

18. The following environment variables can be at most 63 characters in length:

```
prefix_NEW_DATACLAS
prefix_NEW_DSNTYPE
prefix_NEW_MGMTCLAS
prefix_NEW_SPACE
prefix_NEW_STORCLAS
prefix_NEW_UNIT
prefix_WORK_DATACLAS
prefix_WORK_DSNTYPE
prefix_WORK_MGMTCLAS
prefix_WORK_SPACE
prefix_WORK_STORCLAS
prefix_WORK_UNIT
```
19. The following environment variables are for specification of the SPACE parameter, and support only the syntax as shown with their default values (including all commas and parentheses). Also as shown with their default values, individual subparameters can be omitted, in which case the system defaults are used.

   _CCN_IPA_WORK_SPACE
   prefix_NEW_SPACE
   prefix_WORK_SPACE

20. The following environment variables are for specification of the DSNTYPE parameter, and support only the subparameters LIBRARY or PDS (or null for no DSNTYPE):

   prefix.NEW_DSNTYPE
   prefix_WORK_DSNTYPE

21. The following environment variables can be at most 127 characters in length:

   prefix_DCBBF2008
   prefix_DCBBU
   prefix_DCBB121M
   prefix_DCBB133M
   prefix_DCBB137
   prefix_DCBB137A
   prefix_DCBB3200
   prefix_DCBB80
   prefix_DEBUG_FORMAT

   These environment variables are for specification of DCB information, and support only the following DCB subparameters, with the noted restrictions:

   RECFM
      Incorrect values are ignored.

   LRECL
      None

   BLKSIZE
      None

   DSORG
      Incorrect values are treated as if no value had been specified.

22. The following environment variables are parsed as blank-delimited words, and therefore no embedded blanks or other white-space is allowed in the value specified. The maximum length of each word is 1024 characters:

   prefix_INCDIRS
   prefix_INCLIBS
   prefix_LIBDIRS
   prefix_OPTIONS
   prefix_OPERANDS

23. An S-name is a short external symbol name, such as produced by the z/OS XL C/C++ compiler when compiling C programs with the NOLONGNAME option. An L-name is a long external symbol name, such as produced by the z/OS XL C/C++ compiler when compiling C programs with the LONGNAME option.

24. The z/OS XL C/C++ runtime library supports a file naming convention of // (the filename can begin with exactly two slashes). The c89/cc/c++ command indicates that the file naming convention of // can be used.

   However, the Shell and Utilities feature does not support this convention. Do not use this convention (//}} unless it is specifically indicated (as here in c89/cc/c++). The z/OS Shell and Utilities feature does support the z/OS UNIX file naming convention where the file name can be selected from the set of character values excluding the slash and the null character.
25. When coding in C and C++, the **c89**, **cc**, and **c++** commands, by default, produce reentrant executables. For more information about reentrancy, see *z/OS XL C/C++ Programming Guide*. When coding in assembly language, the code must not violate reentrancy. If it does, the resulting executable may not be reentrant.

26. The `prefix_CVERSION`, `prefix_PVERSION` and `prefix_CLASSVERSION` environment variables are set to a hex string in the format 0xPVVRRMMM where P is product, VV is version, RR is release and MMM is modification level. For example, the `prefix_CVERSION` and `prefix_CLASSVERSION` for the z/OS V1R2 compiler is 0x41020000.

27. c89 passes some options to the compiler so that expected behavior is achieved; for example, POSIX behavior. These options are passed onto the compiler as defaults that the user can overwrite. When default options passed by c89 are in conflict with options or pragmas that the user specified, the compiler issues diagnostic messages and may terminate processing. Since the user did not specify options that c89 passed as defaults, these messages may confuse the user. Prior to the z/OS V1R5 release, the compiler was unable to differentiate between the options that c89 passed as defaults and the user-specified options so it was unable to correctly resolve conflicting pragma/option combinations. In some cases, the compiler would overwrite pragmas with the options that c89 passed as defaults thus limiting a user's ability to use pragmas. As of z/OS V1R5, the compiler is now able to recognize c89 defaults and avoid confusion from messages for options, which were not explicitly specified by the user, and overriding pragmas, when the user did not explicitly request it. It is believed that most users will benefit from this feature so it is the default behavior. To enable the old behavior, environment variable `prefix_NOCMDOPTS` must have a nonzero value.

   The following sequence will preserve the old behavior:

   ```bash
   export _C89_NOCMDOPTS=1
   c89 -o hello hello.c
   ```

28. The following example shows the concatenation of data sets in environment variables. It shows how to use an environment variable to setup the SYSLIB DD when using the **c89** command name:

   ```bash
   export _C89_LSYSLIB="CEE.SCEELKEX:CEE.SCEELKED:CBC.SCCNOBJ:SYS1.CSSLIB"
   ```

   This environment variable will produce the following SYSLIB concatenation:

   ```
   //SYSLIB DD DSN=CEE.SCEELKEX,DISP=SHR
   //           DD DSN=CEE.SCEELKED,DISP=SHR
   //           DD DSN=CBC.SCCNOBJ,DISP=SHR
   //           DD DSN=SYS1.CSSLIB,DISP=SHR
   ```

**Localization**

The **c89/cc/c++** command uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`

**Exit values**

- **0**
  - Successful completion.
- **1**
  - Failure due to incorrect specification of the arguments.
- **2**
  - Failure processing archive libraries:
• Archive library was not in any of the library directories specified.
• Archive library was incorrectly specified, or was not specified, following the -l operand.

3
Step of compilation, assemble, or link-editing phase was unsuccessful.

4
Dynamic allocation error, when preparing to call the compiler, assembler, IPA linker, prelinker, or link editor, for one of the following reasons:
• The file or data set name specified is incorrect.
• The file or data set name cannot be opened.

5
Dynamic allocation error, when preparing to call the compiler, assembler, prelinker, IPA linker, or link editor, due to an error being detected in the allocation information.

6
Error copying the file between a temporary data set and a hierarchical file system file (applies to the -2 option, when processing assembler source files, and -x option processing).

7
Error creating a temporary control input data set for the link-editing phase.

8
Error creating a temporary system input data set for the compile or link-editing phase.

**Portability**

For the **c89** command, X/Open Portability Guide, POSIX.2 C-Language Development Utilities Option.

For the **cc** command, POSIX.2 C-Language Development Utilities Option, UNIX systems.

Extensions to the POSIX standard are as follows:
• The -v, -V, -0, -1, -2 and -3 options.
• DLL support.
• IPA optimization support.
• The behavior of the -o option in combination with the -c option and a single source file.

**Note:** -0x (where x is 0, 1, 2, or 3) is equivalent to -x because -x overrides -0. This happens to match the standard compliant syntax of optimization level x (-0x), but 0x is not treated as a single entity. It may appear redundant to use -0x but it is recommended because it improves portability. In order to avoid creating non-portable legacy, the **xlc** utility does not support -x extension syntax. For example, the following commands are equivalent but the first syntax is recommended:

```bash
  c89 -O2 hello.c
  c89 -2 hello.c
```

Features have been added to z/OS releases, which have made it easier to port applications from other platforms to z/OS and improve performance. For compatibility reasons, these portability and performance enhancements could not be made the default. If you are porting an application from another platform to z/OS, you might want to start by specifying the following options:

```bash
  c89 -o HelloWorld -2 -Wc,NOANSIALIAS -Wc,XPLINK\ -Wl,XPLINK -Wc,'FLOAT(IEEE)' -Wc,'GONUM' HelloWorld.c
```

**Note:** The string that is shown in this example is one line (it had to be split to fit the page). A space exists between -Wc,XPLINK and -Wl,XPLINK.
c99 - Compile C source code, link-edit and create an executable file

See xlc.

Note: When working in the shell, to view man page information about c99, type: man xlc.

cal - Display a calendar for a month or year

Format

cal [month] [year]

Description

cal displays a calendar on standard output (stdout).

• With no arguments, cal displays a calendar for the current month of the current year.
• If one argument is given and it is numeric, cal interprets it as a year (for example, 1991). If a single argument is not numeric, cal interprets it as the name of a month, possibly abbreviated (for example, apr).
• If two arguments are given, cal assumes that the first argument is the month (either a number in the range 1 - 12 or a month name) and the second is the year.

Localization

cal uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH


Usage notes

Year numbers that are less than 100 refer to the early Christian era, not the current century. This command prints the Gregorian calendar, handling September 1752 correctly. Many cultures observe other calendars.

Exit values

0

Successful completion.

1

Failure due to any of the following reasons:

• An incorrect command-line argument.
• An incorrect date.
• A year outside the range 1 - 9999 A.D.
Portability
X/Open Portability Guide, UNIX systems.

calendar - Display all current appointments

Format

```
calendar [-]
```

The `calendar` utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, do not use this utility for applications that are intended to be portable to other systems branded by UNIX.

Description

If you do not specify any options, `calendar` displays all current appointments on standard output (stdout). It searches the file `calendar` in the current directory, looking for lines that match either today's date or tomorrow's date. On Friday, Saturday, or Sunday, tomorrow extends through to Monday. Each appointment must fit on a single line, with the date formatted as one of:

```
January 27
1/27
jan 27
```

The name of the month can be abbreviated to three letters. Also, the case is not significant and the month can be given numerically.

Options

- `-`

  Searches the RACF database to find user IDs. `calendar` uses the `mailx` command (or, alternatively, the command named in the MAILER environment variable) to send mail to the corresponding user for any appointments that are found to be current. Because `calendar` cannot determine each user's locale, it runs in the POSIX locale when this option is used; otherwise it runs in the user's locale, processing data in single-byte mode.

Examples

If today is Friday April 7 and the following `calendar` file is found in the current directory:

```
tue mar 7 1:00 pm dentist
Sat April 8 Trip to the zoo
mon april 10 3:30 pm job interview
4/11 vacation starts
```

`calendar` prints the following output:

```
Sat April 8 Trip to the zoo
mon april 10 3:30 pm job interview
```

Environment variables

`calendar` uses the following environment variable:

**MAILER**

Contains the name of the command that `calendar` uses to send mail. If this variable is not set, `calendar` uses `/bin/mail` as the default mail command.
cancel

Files
calendar uses the following file:

**calendar**
File used in the current directory, or user's home directory.

Localization
calendar uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


Exit values

0
Successful completion.

1
Failure due to any of the following reasons:
- An incorrect command-line argument.
- An inability to open the calendar file.

Portability
X/Open Portability Guide, UNIX systems

The MAILER environment variable is an extension to traditional implementations of calendar.

Related information

mailx

cancel - Cancel print queue requests (stub command)

Format

```
cancel [print_(D ...] printer ...
cancel print_(D ... [printer ...]
```

The cancel utility is fully supported for compatibility with earlier UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, this utility should be avoided for applications that are intended to be portable to other UNIX-branded systems.

Description

cancel cancels print queue requests. *print_ID* specifies the particular job (or jobs) to be canceled; the *print_ID* number is reported by *lp* when the job is submitted, or by *lpsstat*.

cancel is recognized, but its functions are not supported.

If you are using the z/OS Infoprint Server feature, your system automatically uses that version of the cancel command.
**captoinfo** - Print the terminal entries in the terminfo database

**Format**

```
captoinfo [-1vV] [-w width] [file ...]
```

**Description**

`captoinfo` prints all of the terminal entries in the terminfo database to standard output in terminfo format. You can either look at the output or send it to a file that can be processed by `tic`.

The Curses application uses the terminfo database, which contains a list of terminal descriptions. You can manipulate a terminal's display regardless of the terminal type. To create the terminfo database, use `tic`. See the section on customizing the terminfo database in *z/OS UNIX System Services Planning* for information about defining the terminfo database.

For more information about curses, see *z/OS C Curses*.

**Options**

2. Print the program version.
3. Print debugging information (verbose) to standard error.
4. Specifies the width of the output.

**filename**

Specifies the termcap entries to be processed.

**Examples**

1. This example shows how to print all the terminal entries in the file `/etc/termcap.src` in terminfo format. The entry for a vt52 is shown. Issue:

   ```
captoinfo /etc/termcap.src
   ```

2. To print all the terminal entries in the file `/etc/termcap.src` in terminfo format with each entry on a separate line, issue:

   ```
captoinfo -1 /etc/termcap.src
   ```

3. This example shows how to write all the terminal entries in the file `/etc/termcap.src` to the file `/test/terminfo.ti`. The resulting file can be processed by `tic`. Notice that the error messages are written to standard error.

   ```
captoinfo /etc/termcap.src 1> /test/terminfo.ti
   ```

**Related information**

`infocmp`, `tic`
cat - Concatenate or display text files

Format

cat [-Bsu] [-v[et]] [-W option[,option]...] [file ... file ...]

Description

cat displays and concatenates files. It copies each file argument to the standard output (stdout). If you specify no files or specify a dash (-) as a file name, cat reads the standard input (stdin).

You can use cat with the scrolling facility of the z/OS UNIX TSO/E command to browse data files.

Options

-B
Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-e
Displays a $ character at the end of each line. This option works only if you also specify -v.

-s
Does not produce an error message if cat cannot find or read a specified file.

-t
Displays tabs as ^I. This option works only if you also specify -v.

-u
Does not buffer output.

-v
Displays all characters including those that are unprintable characters. With a double-byte character set, an unprintable wide character is converted back to its double-byte representation. Each byte is then checked as if it were a single-byte character. If the character is unprintable, one of the following three representations is used:

- M-X is used for character X if the significant bit is set.
- ^X is used for the control character X (for example, ^A for CTRL-A).
- \xxx represents a character with the octal value xxx.

The \xxx form is used if neither of the other representations can be used.

-W option[,option]...
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

filecodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.
When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see *z/OS Unicode Services User's Guide and Reference*.

**pgmcodeset=**<code>codeset</code>

Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is <code>codeset</code>. <code>codeset</code> can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See *z/OS UNIX System Services Planning* for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

**Examples**

1. To display the contents of a text file to the standard output (stdout):
   ```
cat myTextFile
   ```

2. To display the concatenation of two text files to the standard output (stdout):
   ```
cat myTextFile01 myTextFile02
   ```

3. To display the contents of a text file containing UTF-8 characters to the standard output (stdout), assuming that:
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file):
     ```
cat -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myutf8File
     ```

4. To display the contents of a text file that contains UTF-8 characters to the standard output (stdout), assuming that automatic conversion was enabled but the text file is incorrectly tagged as ASCII:
   ```
cat -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File
   ```

**Localization**

`cat` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_SYNTAX`
- `NLSPATH`

**Environment variables**

**cat** uses the following environment variable:

_similarity:._TEXT_CONV

Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-w option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

**Exit values**

0

Successful completion

1

- The code set is not valid.
- Could not turn off automatic conversion.
- Could not perform requested text conversion.
- An incorrect command-line argument.
- Inability to open the input file.
- End of the file detected on stdout.
- The input file is the same as the output file.

2

An incorrect command-line argument

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems

The –B, –e, –s, –t, –v, and -W options are extensions of the POSIX standard.

**Related information**

cp, more, mv

---

**cc - Compile C source code, link-edit, and create an executable file**

See c89/xlc or man xlc.

**Note:**

1. The cc command is fully supported for compatibility with older UNIX systems. However, it is recommended that the c89 command be used instead because it may provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

2. When working in the shell, to view man page information about cc, type: man c89 or man xlc.

---

**cd - Change the working directory**

**Format**

- `cd [directory]`
- `cd old new`
- `cd -`
The command `cd directory` changes the working directory of the current shell execution environment (see `sh`) to `directory`. If you specify `directory` as an absolute path name, beginning with `/`, this is the target directory. `cd` assumes the target directory to be the name just as you specified it. If you specify `directory` as a relative path name, `cd` assumes it to be relative to the current working directory.

If the variable CDPATH is defined in the shell, the built-in `cd` command searches for a relative path name in each of the directories that are defined in CDPATH. If `cd` finds the directory outside the working directory, it displays the new working directory.

Use colons to separate directories in CDPATH. In CDPATH, a null string represents the working directory. For example, if the value of CDPATH begins with a separator character, `cd` searches the working directory first; if it ends with a separator character, `cd` searches the working directory last.

In the shell, the command `cd -` is a special case that changes the current working directory to the previous working directory by exchanging the values of the variables PWD and OLDPWD.

**Note:** Repeating this command toggles the current working directory between the current and the previous working directory.

Calling `cd` without arguments sets the working directory to the value of the HOME environment variable, if the variable exists. If there is no HOME variable, `cd` does not change the working directory.

The form `cd old new` is an extension to the POSIX standard and optionally to the Korn shell. The shell keeps the name of the working directory in the variable PWD. The `cd` command scans the current value of PWD and replaces the first occurrence of the string `old` with the string `new`. The shell displays the resulting value of PWD, and it becomes the new working directory.

If either directory is a symbolic link to another directory, the behavior depends on the setting of the shell's `-o logical` option. See the `set` command for more information.

For `cd` in the tcsh shell, if a directory name is given, `cd` changes the tcsh shell's working directory to `name`. If not, it changes the directory to home. If `name` is `-' it is interpreted as the previous working directory. If `name` is not a subdirectory of the current directory (and does not begin with `/`, `./` or `../`), each component of the tcsh variable `cdpath` is checked to see whether it has a subdirectory name. Finally, if all else fails but `name` is a tcsh shell variable whose value begins with `/`, then this is tried to see whether it is a directory (see also the `implicitcd` tcsh shell variable).

Options for the `cd` tcsh built-in command are:

- `-l`  
  Output is expanded explicitly to home or the path name of the home directory for the user.
  
- `-n`  
  Entries are wrapped before they reach the edge of the screen.
  
- `-p`  
  Prints the final directory stack.
  
- `-v`  
  Entries are printed one per line, preceded by their stack positions.
  
  If more than one of `-n` or `-v` is given, `-v` takes precedence. `-p` is accepted but does nothing.

**Environment variables**

`cd` uses the following environment variables:

**CDPATH**

Contains a list of directories for `cd` to search in when `directory` is a relative path name.

**HOME**

Contains the name of your home directory. This is used when you do not specify `directory` on the command line.
OLDPWD
   Contains the path name of the previous working directory. This is used by cd -.

PWD
   Contains the path name of the current working directory. This is set by cd after changing to that directory.

Localization

cd uses the following localization environment variables:
   • LANG
   • LC_ALL
   • LC_CTYPE
   • LC_MESSAGES
   • NLSPATH


Usage notes

cd is a built-in shell command.

Exit values

0
   Successful completion.

1
   Failure due to any of the following reasons:
       • No HOME directory.
       • No previous directory.
       • A search for directory failed.
       • An old-to-new substitution failed.

2
   An incorrect command-line option or too many arguments.

Messages

Possible error messages include:

dir bad directory
   cd could not locate the target directory. This does not change the working directory.

Restricted
   You are using the restricted version of the shell (for example, by specifying the -r option for sh). The restricted shell does not allow the cd command.

No HOME directory
   You did not assign a value to the HOME environment variable. Thus, when you run cd in order to return to your home directory, cd cannot determine what your home directory is.

No previous directory
   You tried the command cd - to return to your previous directory; but there is no record of your previous directory.

Pattern old not found in dir
   You tried a command of the form cd old new. However, the name of the working directory dir does not contain any string that matches the regular expression old.
Portability

POSIX.2, X/Open Portability Guide.

All UNIX systems feature the first form of the command.

The cd old new form of the command is an extension of the POSIX standard.

Related information
dirs, popd, pushd, set, sh, tcsh

ccebltx - Transform message source files into assembler source files

Format

ccebltx

| [-C csect_name]| [-I secondary_file_name] |
| [-P] [-S] [-c class] [-d APOST'] | QUOTE | "|
| [-l BAL | C | COBOL | FORTRAN | PLI] [-s id] |

in_file out_file

The ccebltx utility only works with z/OS UNIX files; MVS data sets are not applicable.

Description

The ccebltx utility creates several files from the message source file. It creates an assembler source file, which can be assembled into an object (text) file and link-edited into a module in an MVS load library. When the name of the module is placed in a message module table, the Language Environment message services can dynamically access the messages. See Creating a Message Module Table in the Language Environment Programming Guide for more information about creating a message module table.

The ccebltx utility optionally creates secondary input files (COPY or INCLUDE), which contain declarations for the condition tokens associated with each message in the message source file. When a program uses the secondary input file, the condition tokens can then be used to reference the messages from the message table. The :msgname. tag indicates the symbolic name of the condition token.

See the topic on Using and handling messages in z/OS Language Environment Programming Guide for a description of creating message source files and other corresponding information.

Operands

in_file

The name of the file containing the message source.

out_file

The name of the resulting assembler source file containing the messages, inserts, and others items, suitable for input into the High Level Assembler. An extension of ".s" is assumed if none is present.
Options

-C `csect_name`
This option is used to explicitly specify the CSECT name. An uppercase version of the CSECT name will be used. By default, the CSECT name is the output file base name.

-I `secondary_file_name`
The -I (uppercase i) option provides the name of the secondary input file generated for the language specified with the -l (lowercase L) option. If no suffix is present in the `secondary_file_name` specified, the extension will be ".h" for C, ".fortran" for Fortran, and ".copy" for all others.

-P
This option is used to save previous prologs, if files being generated exist in the directory and contain prologs. By default, previous prologs are not reused.

-S
This option is used to indicate sequence numbers should be generated in the files produced. By default, no sequence numbers are generated.

-c `class`
This option is used to specify the default value for :msgclass. in cases where the tag is not coded.

-d `APOST | ' | QUOTE | "`
This option is used to specify which COBOL delimiter to use and is used in combination with the -l (lowercase L) COBOL option. By default, APOST is used as the delimiter. Quotation marks should be escaped in order to prevent them from being treated as shell metacharacters, for example:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceebltxt -l COBOL -I secondary_file_name -d ' in_file out_file</td>
</tr>
<tr>
<td>ceebltxt -l COBOL -I secondary_file_name -d &quot; in_file out_file</td>
</tr>
<tr>
<td>ceebltxt -l COBOL -I secondary_file_name -d QUOTE in_file out_file</td>
</tr>
</tbody>
</table>

-l `BAL | C | COBOL | FORTRAN | PLI`
The -l (lowercase L) option is used to specify the language to be used in generating a secondary input file and is used in combination with the I `secondary_file_name` option. The file will contain declarations for the condition tokens associated with each message in the message source file. The language is accepted in lowercase and uppercase. C370 is also supported.

-s `id`
This option is used to specify the default value for :msgsubid. in cases where the tag is not coded.

Examples

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceebltx -l PLI -I exmplcop example exmplasm</td>
</tr>
</tbody>
</table>

Where the `in_file` is example, the `out_file` is exmplasm.s, and the PL/I `secondary_file_name` is exmplcop.copy.

After the `out_file` is generated, the High Level Assembler can be used to assemble the `out_file` into an object file

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>as</code> exmplasm.s</td>
</tr>
</tbody>
</table>

and the binder can be used to link-edit it into an MVS load library:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ld</code> -o &quot;/mylib(exmplasm)&quot; -e/ -u//exmplasm exmplasm.o</td>
</tr>
</tbody>
</table>

A CSECT name greater than 8 characters requires the use of the High Level Assemble GOFF option for assembling the primary output file.
Exit values

-1
Rexx terminated execution due to lack of storage. (See IRX0005I in z/OS TSO/E Messages.)

Attempt one of the following options:
1. Increase the virtual storage space available on the system.
2. Split up the script in_file, into two or more files, and adjust the Message Module Table for the corresponding split.

0
Successful completion.

5
Error reading file sssssss.

6
Error erasing file sssssss.

7
Error writing file sssssss.

8
Bad file name sssssss: forward slash not allowed at the end of a file name.

9
Option x requires an argument.

10
Invalid option = x. Valid options are: CIPScdls.

11
Bad data set name sssssss.

20
CSECT name sssssss is greater than 63 characters.

21
CSECT name sssssss does not begin with a letter, $, #, @ or underscore (_).

28
sssssss SCRIPT not found on any accessed disk.

40
Error on line nnn in message nnnn. Insert number greater than mmmm.

44
Error on line nnn. Duplicate: FACID. tags found with the given script file.

48
No :FACID. tag found within the given script file.

52
Error on line nnn. Message number nnnn found out of range mmmm to mmmm.

56
Number of hexadecimal digits not divisible by 2 on line nnn in message nnnn.

60
Invalid hexadecimal digits on line nnn in message nnnn.

64
Number of DBCS bytes not divisible by 2 on line nnn in message nnnn.

68
PLAS out_file name must be longer than the message facility ID pppp.

72
Message facility ID pppp on line nnn was longer than 4 characters.

76
Message class on line nnn was not a valid message class type: IWESCFA.
Tag not recognized on line $nnn$.

The first tag was not a :FACID. tag on line $nnn$.

Unexpected tag found on line $nnn$.

Duplicate tags $ttt$ found on line $nnn$.

No :MSGNO. tags found within the given SCRIPT file.

No :MSGCLASS. (or :MSGCL.) tag found for message $nnnn$.

Insert number was not provided or was less than 1 on line $nnn$.

Message subid was out of the range $mmmm$ to $mmmm$ on line $nnn$.

Existing secondary file, $ssssss$, found, but not on A-disk.

The current ADDRESS environment not CMS, TSO/E, or z/OS UNIX.

Undefined error number $nnn$ issued. Contact your service representative.

**chaudit - Change audit flags for a file**

**Format**

`chaudit [-Fdai] attr pathname ...`

**Description**

`chaudit` changes the audit attributes of the specified files or directories. Audit attributes determine whether or not accesses to a file are audited by the system authorization facility (SAF) interface.

The `chaudit` command can be used only by the file owner or a superuser for non-auditor-requested audit attributes. Only a user with auditor authority can change the auditor-requested audit attributes.

**Options**

- **-F**
  
  If you specify a directory as a path name on the command, `chaudit` changes the audit characteristics of all files in that directory. Subdirectory audit characteristics are not changed.

- **-d**
  
  If you specify a directory as a path name on the command, `chaudit` changes the audit characteristics of all the subdirectories in that directory. File audit characteristics are not changed.

- **-a**
  
  Auditor-requested audit attributes are to be changed for the specified files or directories. If -a is not specified, user-requested audit attributes are changed.

- **-**
  
  Does not issue error messages concerning file access authority, even if `chaudit` encounters such errors.
The symbolic form of the `attr` argument has the form:

```
[operation]
op auditcondition[op auditcondition ...]
```

The `operation` value is any combination of the following:

- `r` Sets the file to audit read attempts.
- `w` Sets the file to audit write attempts.
- `x` Sets the file to audit execute attempts.

The default is `rwx`.

The `op` part of a symbolic mode is an operator telling whether `chaudit` should turn file auditing on or off. The possible values are:

- `+` Turns on specified audit conditions.
- `-` Turns off specified audit conditions.
- `=` Turns on the specified audit conditions and turns off all others.

The `auditcondition` part of a symbolic mode is any combination of the following:

- `s` Audit on successful access if the audit attribute is on.
- `f` Audit on failed access if the audit attribute is on.

You can specify multiple symbolic `attr` values if you separate them with commas.

**Examples**

1. The command:

   ```
   chaudit -s file
   ```

   changes the file `file` so that successful file accesses are not audited.

2. The command:

   ```
   chaudit rwx=sf file1
   ```

   changes the file `file1` so that all successful and unsuccessful file accesses are audited.

3. The command:

   ```
   chaudit r=f file2
   ```

   changes the file `file2` so that unsuccessful file read accesses are audited.

4. The command:

   ```
   chaudit r-f,w+s file3
   ```

   changes the file `file3` to not audit unsuccessful file read accesses and to audit successful write accesses.

**Localization**

`chaudit` uses the following localization environment variables:
chcp

* LANG
* LC_ALL
* LC_MESSAGES
* NLSPATH


**Exit values**

0  
Successful completion

1  
Failure due to any of the following reasons:
  * Inability to access a specified file.
  * Inability to change the audit attributes for a specified file.
  * Inability to not read the directory containing item to change.
  * Irrecoverable error when using the -F or -d option.

2  
Failure due to any of the following reasons:
  * Missing or incorrect *attr* argument.
  * Too few arguments.

**Messages**

Possible error messages include:

**fatal error during -F or -d option**
  You specified the -F or -d option, but some file or directory in the directory structure was inaccessible. This may happen because of permissions or because you have removed a removable unit.

**read directory name**
  You do not have read permissions on the specified directory.

**Portability**

None. This is a security extension that comes with z/OS UNIX services.

**Related information**

chmod, chown, ls

---

**chcp - Set or query ASCII/EBCDIC code pages for the terminal**

**Format**

```bash
chcp [-r | -q]
chcp [-s] [-a ASCII_cp] [-e EBCDIC_cp]
```

**Description**

*chcp* sets, resets, or queries the current ASCII/EBCDIC code conversion in effect for the controlling terminal. Use it when the terminal requires ASCII data and the shell application uses EBCDIC. Do not use *chcp* if you are logged on through the TSO/E OMVS command. The _BPX_TERMPATH_ environment variable enables shell scripts to tell if the user logged on from TSO, rather from rlogin or telnet.
Options

-a ASCII_cp
The name of the ASCII code page used by the terminal. EBCDIC data from the shell application is converted to this ASCII code page before it is sent out to the terminal. Data from the terminal is converted from this ASCII code page to EBCDIC before the application receives it.

The name of the ASCII code page is case-sensitive.

For a list of code pages supported by the shell, see z/OS XL C/C++ Programming Guide.

-e EBCDIC_cp
The name of the EBCDIC code page used for this session. EBCDIC data from the shell application is converted from this EBCDIC code page to ASCII before it is sent out to the terminal. ASCII data from the terminal is converted to this EBCDIC code page before the application receives it.

The name of the EBCDIC code page is case-sensitive.

For a list of code pages supported by the z/OS shell, see z/OS XL C/C++ Programming Guide.

-q
Queries the current ASCII and EBCDIC code pages for this terminal. The results are written to standard output. You cannot use any other options if you use the -q option.

-r
Resets the ASCII/EBCDIC conversion for the terminal to the default code pages. The default ASCII code page is ISO8859-1, and the default EBCDIC code page is IBM-1047.

You cannot use -q with any other options.

-s
Specifies that the ASCII/EBCDIC conversion for the terminal is to use the code pages specified by the -a and -e options. You cannot use -s with any other options other than -a or -e. Either -a or -e (or both) must also be specified if -s is used.

The chcp query output is written to standard output. For example, if you enter

```
chcp -q
```

You get the following output:

```
Current ASCII code page = ISO8859-1
Current EBCDIC code page = IBM-1047
```

Examples

1. To set the ASCII and EBCDIC code pages to IBM-eucJP and IBM-939, enter:

   ```
   chcp -a IBM-eucJP -e IBM-939
   ```

2. To change just the EBCDIC code page to IBM-277, enter:

   ```
   chcp -se IBM-277
   ```

3. To change just the ASCII code page to IBM-850, enter:

   ```
   chcp -a IBM-850
   ```

4. To reset ASCII/EBCDIC code page conversion to the default code pages for this terminal, enter:

   ```
   chcp -r
   ```

5. To query the current ASCII and EBCDIC code pages for this terminal, enter:

   ```
   chcp -q
   ```
Usage notes

1. Do not use `chcp` when you are logged on from the TSO/E OMVS command because the OMVS command does not do any ASCII/EBCDIC code page conversion.

   Shell scripts can test the `_BPX_TERMPATH` environment variable and bypass `chcp` when the user is logged on through OMVS. (The `_BPX_TERMPATH` environment variable enables shell scripts to tell if the user logged on from TSO/E rather than from rlogin or telnet.)

   Before starting the session, the TSO/E OMVS command sets `_BPX_TERMPATH` to OMVS.

   Sample shell script code:

   ```
   # -----------------------------------------------
   # Issue chcp only if not using TSO/E OMVS command
   # -----------------------------------------------
   if test "$_BPX_TERMPATH" != "OMVS"
   then
     chcp -a IBM-850 -e IBM-047
   fi
   ```

2. After running `chcp -s` to change the EBCDIC code page for the session, you may also need to alter or set the following environment variables to match the new code page:

   - `LANG`
   - `LC_ALL`
   - `LC_COLLATE`
   - `LC_CTYPE`
   - `LC_MESSAGES`
   - `LC_SYNTAX`
   - `NLSPATH`

3. The code page names supplied with the `-a` and `-e` options are passed to `iconv_open()` without any uppercase or lowercase conversion. Code page converters that convert between the specified ASCII and EBCDIC code pages must be available for `iconv()`.

4. If ASCII/EBCDIC conversion is not active for this terminal, both the ASCII and EBCDIC code pages must be specified on the `chcp -s` command. At other times, omit `-a` when just the EBCDIC code page needs to be changed. Omit `-e` when just the ASCII code page needs to be changed.

5. All code pages with names not known to `chcp` are considered to be single-byte (SBCS) user-defined code pages. User-defined multibyte code pages are not supported.

6. `chcp` cannot check user-defined code page names to make sure that `-a` really specifies an ASCII code page and `-e` specifies an EBCDIC code page. In this case, specifying the wrong code pages may cause terminal input and output to be completely unreadable. It may also be impossible to enter any more shell commands.

7. `chcp` operates on the controlling terminal.

8. `chcp` should not be run as a background job.

9. The `-d` option specifies that special debugging information be printed. Specify this option only when requested by IBM.

Localization

`chcp` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_COLLATE`
- `LC_CTYPE`
- `LC_MESSAGES`
Exit values

0  Successful completion.
1  Incorrect command-line arguments or options.
2  Any of the following errors:
   • There is no controlling terminal.
   • The controlling terminal does not support ASCII/EBCDIC code page conversion (the TSO/E OMVS command, for example).
   • iconv() fails when passed the code page names specified on the command line.
   • chcp cannot build SBCS conversion tables using iconv() when required.
   • An I/O error occurred on the controlling terminal.
   • Either the -a or -e was omitted and the chcp –s command was run while the terminal code page conversion is in binary mode.

Portability

None. chgp is not described in any standard.

Related information

lm, rlogin

chgrp - Change the group owner of a file or directory

Format

chgrp [-fhR] group pathname ...

Description

chgrp sets the group ID to group for the files and directories named by the pathname arguments. group can be a group name from a group database, or it can be a numeric group ID (GID).

chgrp can be used only by the file owner or a superuser. The file owner must have the new group as his or her group or one of the supplementary groups.

chgrp also turns off the set-user-ID bit and set-group-ID bit of the named files and directories.

Options

-f  Does not issue an error message if chgrp cannot change the group ID. In this case, chgrp always returns a status of 0.
-h  Does not attempt to follow the symbolic link (or external link), but instead makes changes to the symbolic link (or external link) itself.
-R  If a pathname on the command line is the name of a directory, chgrp changes the group ID of all files and subdirectories in that directory. If chgrp cannot change some file or subdirectory in the directory,
it continues to try to change the other files and subdirectories in the directory, but exits with a nonzero status.

Localization
chgrp uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH

Exit values
0
You specified -f, or chgrp successfully changed the group ownership of all the specified files and directories.

1
Failure due to any of the following reasons:
• Inability to access a specified file.
• Inability to change the group of a specified file.
• An unrecoverable error was encountered when you specified the -R option.

2
Failure due to any of the following reasons:
• The command line contained an unknown option or too few arguments.
• chgrp did not recognize the specified group.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The -f and -h options are an extension of the POSIX standard.

Related information
chmod, chown

chlabel - Set the security label of files and directories

Format
chlabel [-cqR] [-h|-L] seclabel pathname ...

Description
chlabel sets the security label of the files and directories that are specified by pathname. Setting the security label is only allowed if the user has RACF SPECIAL authority, and no security label currently exists on the resource. Once a security label is set, it cannot be changed.

seclabel is a 1-8 character security label that corresponds to a RACF security level with a set of zero or more security categories. See z/OS Planning for Multilevel Security and the Common Criteria for restrictions on security label.
If chlabel could not set the security label for a file or object, it continues to try to change the other files but exits with a nonzero status.

When –R is specified, chLabel will not cross device boundaries from the directory that is specified by pathname unless the -c option is used.

Options

-c
Cross device boundaries.

-h
Does not follow the symbolic link (or external link), but instead makes changes to the symbolic link (or external link) itself. Cannot be used with -L.

-L
Follow symbolic links. Cannot be used with -h.

-q
Quiet mode. chlabel suppresses all warning messages. The condition that caused the warning does not affect the exit value.

-R
chlabel sets the security label on all the file objects and subdirectories under the directory that is specified by pathname.

Usage notes

1. See z/OS Planning for Multilevel Security and the Common Criteria for more information about multilevel security and security labels.
2. chlabel will not set the security label for a symbolic link, or for the file to which it points, unless either the -h or -L option is specified. If neither option is specified, chlabel prints a warning, continues to the next file and exits with a nonzero status.
3. chlabel is typically run to set up security labels on file systems before multilevel security is activated.
4. Only the zFS file system supports the setting of security labels.
5. The SECLABEL class must be active before the chlabel command will set a security label. If the SECLABEL class is not active, security labels will not be set.

Exit values

0
Successful completion.

1
Failure due to any of the following reasons:
• The user does not have RACF SPECIAL authority.
• The user specified a security label with more than 8 characters.
• The file system does not support setting security labels.
• The RACF SECLABEL class is not active.

2
Command syntax error.

3
One or more warnings occurred, due to any of the following reasons:
• The path name already has a security label assigned.
• A symbolic link was encountered, but neither -h nor -L was specified.
• Device boundary not crossed.
### Examples

1. To set the security label TOPSEC for file `secret_file`:

   ```
   chlabel TOPSEC secret_file
   ```

2. To set the security label SYSLOW for a symbolic link `mylink`:

   ```
   chlabel -h SYSLOW mylink
   ```

3. To set the security label `SYSLOW` for the file to which the symbolic link `mylink` points:

   ```
   chlabel -L SYSLOW mylink
   ```

4. To recursively set the security label `SYSHIGH` for all files, symbolic links, and subdirectories under the directory `Team`:

   ```
   chlabel -Rh SYSHIGH Team
   ```

### chmod - Change the mode of a file or directory

#### Format

```
chmod [-fhR] mode pathname
```

#### Description

`chmod` changes the access permissions, or *modes*, of the specified file or directory. (Modes determine who can read, write, or search a directory or file.) Users with read access to `SUPERUSER.FILESYS.CHANGEPERMS` (a UNIXPRIV class profile), can use the `chmod` command to change the permission bits of any file.

`chmod` can be used only by the file owner or a superuser.

#### Options

- **-f**
  
  Does not issue error messages concerning file access permissions, even if `chmod` encounters such errors.

- **-h**
  
  Suppresses a mode change for the file or directory pointed to by the encountered symbolic link (or external link). Symbolic link (or external link) permissions cannot be changed on a z/OS system.

- **-R**
  
  Recursively change file mode bits. For each path name operand that names a directory, `chmod` will change the file mode bits of the directory and all files in the file hierarchy under it.

`chmod` never changes the permissions of symbolic links (or external links), because, on a z/OS system, the permissions on symbolic links (and external links) are never used. When `-h` is not specified and symbolic links (or external links) are specified or encountered during the file hierarchy traversal, the links are followed, and the resolved directory (and files and subdirectories) are changed.

You can specify the *mode* value on the command line in either symbolic form or as an octal value.

The symbolic form of the *mode* argument has the form:

```
[who] op permission[op permission ...]
```

The *who* value is any combination of the following:
Sets owner (user or individual) permissions.

g
Sets group permissions.

Sets other permissions.

Sets all permissions; this is the default. If a who value is not specified, the default is a, modified by umask.

The op part of a symbolic mode is an operator that tells chmod to turn the permissions on or off. The possible values are:

+  
Turns on a permission.

-  
Turns off a permission.

=  
Turns on the specified permissions and turns off all others.

The permission part of a symbolic mode is any combination of the following:

r
Read permission. If this is off, you cannot read the file.

x
Execute permission. If this is off, you cannot run the file.

X
Execute or search permission for a directory; or execute permission for a file only when the current mode has at least one of the execute bits set.

w
Write permission. If this is off, you cannot write to the file.

s
If in owner permissions section, the set-user-ID bit is on; if in group permissions section, the set-group-ID bit is on.

A superuser or the file owner can use a chmod command or chmod() function to change two options for an executable file. The options are set in two file mode bits:

• Set-user-ID (S_ISUID) with the setuid option.
• Set-group-ID (S_ISGID) with the setgid option.

If one or both of these bits are on, the effective UID, effective GID, or both, plus the saved UID, saved GID, or both, for the process running the program are changed to the owning UID, GID, or both, for the file. This change temporarily gives the process running the program access to data the file owner or group can access.

In a new file, both bits are set off. Also, if the owning UID or GID of a file is changed or if the file is written in, the bits are turned off. In shell scripts, these bits are ignored.

If the RACF profile named FILE.GROUPOWNER.SETGID exists in the UNIXPRIV class, then the set-group-ID bit for a directory determines how the group owner is initialized for new objects created within the directory:

• If the set-gid bit is on, then the owning GID is set to that of the directory.
• If the set-gid bit is off, then the owning GID is set to the effective GID of the process.

t
This represents the sticky bit. For a file, the sticky bit causes a search for the program in the user's STEPLIB, the link pack area, or link list concatenation. For a directory, the sticky bit allows files in a
directory or subdirectories to be deleted or renamed only by the owner of the file, by the owner of the directory, or by a superuser.

You can specify multiple symbolic names if you separate them with commas.

Absolute modes are octal numbers specifying the complete list of attributes for the files; you specify attributes by ORing together these bits.

<table>
<thead>
<tr>
<th>Octal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>Set-user-ID bit</td>
</tr>
<tr>
<td>2000</td>
<td>Set-group-ID bit</td>
</tr>
<tr>
<td>1000</td>
<td>Sticky bit</td>
</tr>
<tr>
<td>0400</td>
<td>User read</td>
</tr>
<tr>
<td>0200</td>
<td>User write</td>
</tr>
<tr>
<td>0100</td>
<td>User execute (or list directory)</td>
</tr>
<tr>
<td>0040</td>
<td>Group read</td>
</tr>
<tr>
<td>0020</td>
<td>Group write</td>
</tr>
<tr>
<td>0010</td>
<td>Group execute</td>
</tr>
<tr>
<td>0004</td>
<td>Other read</td>
</tr>
<tr>
<td>0002</td>
<td>Other write</td>
</tr>
<tr>
<td>0001</td>
<td>Other execute</td>
</tr>
</tbody>
</table>

**Examples**

1. To remove write permission from orgcht:

   ```
   chmod -w orgcht
   ```

2. To turn on read, write, and execute permissions, and turn off the set-user-ID bit, set-group-ID bit, and sticky bit attributes. This is equivalent to `chmod 0777 aprsal`:

   ```
   chmod a=rwx aprsal
   ```

3. To set all permission bits on (anyone can read/write/execute):

   ```
   chmod 777 scratch
   ```

4. To set user (owner) executable permission bit on:

   ```
   chmod u+x file
   ```

5. To set group read/write permission bits:

   ```
   chmod g+rw file
   ```

6. To set other write permission off on 2 files:

   ```
   chmod o-w file1 file2
   ```

7. To set group read/write/execute permissions on the directory `/public/teamdir` and all its files and subdirectories:

   ```
   chmod -R g+rwx /public/teamdir
   ```

8. To set group read/execute on, group write off on `/u/ateam/pgm`:

   ```
   chmod g=rx /u/ateam/pgm
   ```

**Localization**

`chmod` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_MESSAGES
• NLSPATH

Exit values

0  Successful completion

1  Failure due to any of the following reasons:
   • Inability to access a specified file.
   • Inability to change the modes on a specified file.
   • Inability to read the directory containing the item to change.
   • An unrecoverable error was encountered when using the -R option.

2  Failure due to any of the following reasons:
   • Missing or incorrect mode argument.
   • Too few arguments.

Messages

Possible error messages include:

function not implemented
This error may occur if the directory is under automount control.

irrecoverable error during -R option
The -R option was specified, but some file or directory in the directory structure was inaccessible. This may happen because of permissions.

read directory name
Read permissions are not on the specified directory.

Portability

POSIX.2, X/Open Portability Guide.
The -f and -h options and the t permission are extensions of the POSIX standard.

Related information
ls, setfacl, umask

chmount - Change the mount attributes of a file system

Format
chmount [-DRrsw] [-dsysname] [-d destsys] [-a yes|no|unmount|include,exclude,sysname1,...,sysnameN] pathname ...

Description

The chmount shell command, which is located in /usr/sbin, changes the mount attributes of a specified file system.
**Restriction:** A **chmount** user must have UID(0) or at least have READ access to the SUPERUSER.FILES.MOUNT resource found in the UNIXPRIV class.

**Options**

- **-a yes | no | unmount|include,sysname1,...,sysnameN|exclude,sysname1,...,sysnameN**
  The -a option specifies the AUTOMOVE attribute of the file system in a sysplex environment where systems are exploiting the shared file system capability.
  - -a yes allows the system to automatically move logical ownership for a specified file system as needed. This is the default.
  - -a no prevents ownership movement in some situations.
  - -a unmount unmounts the file system in some situations.
  - -a include,sysname1,...,sysnameN specifies a list of systems, in priority order, to which the file system's ownership can be moved. include can be abbreviated to i.
  - -a exclude,sysname1,...,sysnameN specifies a list of systems, in priority order, to which the file system's ownership cannot be moved. exclude can be abbreviated to e.

  See the data movement section in *z/OS UNIX System Services Planning* for details about the behavior of the AUTOMOVE options.

- **-D**
  Reassigns logical ownership of a file system to any available file system that is participating in shared file system.

- **-d destsys**
  To designate a specific reassignment, use -d destsys, where destsys becomes the logical owner of a file system in a shared file system environment.

- **-R**
  Changes the attributes of a specified file system and all file systems that are mounted under it in the file system hierarchy.

- **-r**
  Switches the specified file system to read-only mode.

- **-s**
  Remounts the specified file system but does not change the current mode.

- **-w**
  Switches the specified file system to read/write mode.

  **pathname...** specifies the path names to use for locating the file systems that need attributes that are changed.

**Examples**

To move ownership of the file system that contains /u/wjs to SY1:

```
chmount -d SY1 /u/wjs
```

**Usage notes**

Because the path name for chmount and umount is a node, symbolic links cannot be followed unless a trailing slash is added to the symbolic link name. For example, if /etc has been converted into a symbolic link, /etc -> $SYSNAME/etc, issuing chmount -w /etc without the trailing slash will result in trying to chmount -w /etc -> $SYSNAME/etc. Depending on the security access for the symbolic link, RACF errors might occur. However, if you specify chmount -w /etc/ with the trailing slash, the symbolic link will be followed and RACF will determine the access from the file being linked to.
Exit values

0
Successful completion

Related information
mount, unmount

chown - Change the owner or group of a file or directory

Format
chown [-fhR] owner[:group] pathname ...

Description
chown sets the user ID (UID) to owner for the files and directories that are named by pathname arguments. owner can be a user name from the user database, or it can be a numeric user ID. (If a numeric owner exists as a user name in the user database, the user ID number associated with that user name is used.) If there is no change to the UID, then specify - - -1.

If you include a group name (that is, if you specify owner followed immediately by a colon (:)) and then group with no intervening spaces, such as owner:group) chown also sets the group ID (GID) to group for the files and directories named. group can be a group name from the security facility group database, or it can be a numeric group ID. If a numeric group exists as a group name in the group database, the group ID number that is associated with that group is used. If there is no change to the GID, then specify -1 (or do not specify the :group).

Restriction: Only a superuser can change the UID. To change the GID, you must either be a superuser, or the effective user ID of the process must be equal to the user ID of the file owner, and the owner argument is also equal to the user ID of the file owner or -1, and the group argument is the calling process’s effective group ID or one of its supplementary group IDs.

chown also turns off the set-user-ID bit and set-group-ID bit of the named files and directories.

For additional information related to chown usage, see the description of the UNIXPRIV class profiles CHOWN.UNRESTRICTED and SUPERUSER.FILESYS.CHOWN in z/OS UNIX System Services Planning.

Options

- f
  Does not issue an error message if chown cannot change the owner. In this case, chown always returns a status of zero. Other errors may cause a nonzero return status.

- h
  Does not attempt to follow the symbolic link (or external link), but instead makes the changes on the symbolic link (or external link) itself.

- R
  If pathname on the command line is the name of a directory, chown changes all the files and subdirectories in that directory to belong to the specified owner (and group, if :group is specified).

  If a symbolic link is specified or encountered during the traversal of a file hierarchy, chown changes the directory referenced by the symbolic link and all files in the file hierarchy under it.

  If chown cannot change some file or subdirectory in the directory, it continues to try to change the other files and subdirectories in the directory, but exits with a nonzero status.

Localization

chown uses the following localization environment variables:
chroot

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Exit values**

0
You specified -f, or **chown** successfully changed the ownership of all the specified files and directories.

1
Failure due to any of the following reasons:
- Inability to access a specified file.
- Inability to change the owner of a specified file.
- Inability to read the directory that contains the directory entry of the file.
- An unrecoverable error was encountered when using the -R option.

2
Failure due to any of the following reasons:
- The command line contained an incorrect option.
- The command line had too few arguments.
- An owner was specified with a user ID that the system did not recognize.

**Messages**

**function not implemented**
This error might occur if the directory is under automount control.

**Portability**
POSIX.2, UNIX systems.
The -f and -h options are an extension of the POSIX standard.

**Related information**
chgrp, chmod

**chroot - Change the root directory for the execution of a command**

**Format**
chroot directory command

**Description**
If you have appropriate privileges, the chroot command changes the root directory to the directory specified by the directory parameter of a specific command. The new root directory will also contain its children.
In order to use chroot, you must either be a superuser (UID=0), or have READ permission to the BPX.SUPERUSER resource profile in the FACILITY class.

The directory path name is always relative to the current root. If a nested chroot command is in effect, the directory path name is still relative to the current (new) root of the running process.

In order for your process to operate properly after the chroot is issued, you need to have in your new root all the files that your program depends on. For example, if your new root is /tmp and you issue an ls, you will get a not found error. To use ls with /tmp as your new root, you will need a /tmp/bin with ls in it before you issue the chroot command.

In addition, utilities that depend on locale-sensitive files (/usr/lib/nis/*) might be unsuccessful if these files are not in the new root file system.

After chroot is issued, your current working directory is the new root (directory), chroot does not change environment variables.

directory
   Specifies the new root directory.

command
   Specifies a command to run with the chroot command.

Examples

1. To run the ls command with the /tmp directory as the root file system, enter:

   ```
   mkdir /tmp/bin
   cp /bin/ls /tmp/bin
   chroot /tmp ls
   ```

2. To run a child shell with another file system as the root file system (assuming that /tmp is the mount point of a file system), enter:

   ```
   mkdir /tmp/bin
   cp /bin/sh /tmp/bin
   chroot /tmp sh       or       chroot /tmp /bin/sh
   ```

   This makes the directory name / (slash) refer to the /tmp for the duration of the /bin/sh command. It also makes the original root file system inaccessible. The file system on the /tmp file must contain the standard directories of a root file system.

   Running the sh command creates a child shell that runs as a separate process from your original shell. Press the END OF FILE (Crtl-D) key sequence or type exit to end the child shell and go back to where you were in the original shell. This restores the environment of the original shell, including the meanings of the . (current directory) and the / (root directory).

3. To create a file relative to the original root, not the new one, enter:

   ```
   chroot Directory Command > file
   ```

   For example, chroot /tmp ls > /bin/file will create the file in /bin/file.

   **Note:** Redirection is handled by the current shell before chroot is executed.

4. To create a file relative to the new root, enter:

   ```
   chroot Directory 'Command > file'
   ```

   For example, chroot /tmp 'ls > /bin/file' will create the file in /tmp/bin/file.

5. Examples of how the current root changes:

   Given the standard directories of the file system plus:

   ```
   # echo $PATH
   /bin
   # ls /tmp/bin
   bin file2 sh
   ```
# ls /tmp/bin/bin
file1 sh

# whence file2
#
# whence file1
#
# chroot /tmp 'whence file1'
#
# chroot /tmp 'type file2'
/bin/file2

# chroot /tmp/bin 'type file1'
/bin/file1

**Exit values**

0  
The command completed successfully

1  
Failure due to any of the following reasons:

* chroot seteuid failed.
* User not authorized to issue chroot.

2  
Failure due to any of the following reasons:

* Cannot chdir to directory specified.
* chroot cannot change root.
* Unable to execute the shell.
* Incorrect command syntax.

If the SHELL environment variable is set, chroot uses its value to invoke the shell.

### chtag - Change file tag information

**Format**

```
chtag  -b | -r [-hqRv] pathname ...
chtag  -c codeset [-hqRv] pathname ...
chtag  -m | -t [-c codeset] [-hqRv] pathname ...
chtag  -p [-hqRv] pathname ...
```

**Rule:** You must have write permission to the file or be a superuser in order to use chtag.

**Description**

chtag allows you to set, modify, remove, or display information in a file tag. A file tag is composed of a text flag (txtflag) and a coded character set.

**codeset**

Represents the coded character set in which text data is encoded. The code set can be used for uniformly encoded text files or files that contain mixed text/binary data.

**txtflag**

Indicates whether a file contains uniformly encoded or non-uniformly encoded text data.

* txtflag=ON indicates that the file has uniformly encoded text data.
* txtflag=OFF indicates that the file has non-uniformly encoded text data.
Only files with txtflag = ON and a valid code set are candidates for automatic conversion. If txtflag = OFF and a code set is associated with it, automatic conversion will not take effect. However, user applications can take advantage of the associated code set information and perform code set conversion by themselves.

For information about enabling automatic conversion, see converting files between code pages in z/OS UNIX System Services Planning.

Options

-b
Indicates that the file contains only binary (non-uniformly encoded) data. Automatic conversion is disabled with this option.

-b is mutually exclusive with the -c, -m, -t, or -r options.

-c codeset
Allows the user to modify the coded character set associated with the file. codeset can be a code set name that is known to the system or the numeric coded character set identifier (CCSID). If a code set name exists, the numeric CCSID associated with that name is used. -c is mutually exclusive with the -r and -b options. The command icnv -1 lists existing CCSIDs along with their corresponding code set names.

Modifying the code set associated with an untagged file without specifying -t causes the file to be marked as "mixed".

-h
Does not change file tag information if the file is a symbolic link or an external link.

-m
Indicates that the file contains mixed text and binary data. The data is not uniformly encoded, but to identify the encoding of portions of the file that are text, this option allows the specifications of a code set with the -c option. This option sets txtflag = OFF. When used without -c, the existing coded character set that is associated with the file is retained.

Automatic conversion is disabled with this option. However, user applications can independently convert any text data that resides in the file by knowing the code set associated with it. -m is mutually exclusive with the -b, -t and -r options.

Specifying -m without -c on an untagged file does not affect the tagging of the file.

-p
Prints file tag information that is associated with a file. If a code set name is not associated with the numeric CCSID in the file tag, the numeric CCSID is displayed instead.

The following example is a sample of the output you might see:

```
t IBM-1047 T=on   file1
- untagged T=on   file2
b binary   T=off  file3
m ISO-8859-1 T=off file4
- untagged T=off  file5
b binary   T=on   file6
```

where:

- **t** Text

- **b** Binary

- **m** Mixed


Note: Code sets that are aliases of each other exist, which might cause the test to fail because the file inquiry operator might return an alias of the code set you are testing.

-q
Suppresses warning messages.

-r
Removes any tagging information that is associated with the file and sets the status of the file to "untagged". This option disables automatic conversion for the files. -r is mutually exclusive with the -b, -c, -m, and -t options.

-R
Recursively changes the file tag information. For each pathname operand that names a directory, chtag changes the file tag information about all of the files in the file hierarchy below it. When -h is not specified, and symbolic links (or external links) are specified or encountered during the file hierarchy traversal, the links are followed, and the resolved file (or files in the directory) are changed.

-t
Indicates that the specified file contains pure (uniformly encoded) text data. Used alone, this option sets txtflag = ON and retains the existing coded character set associated with the file. To set or change the code set, use the -c option. Files that are tagged with this option and contain a valid code set are candidates for automatic conversion. -t is mutually exclusive with the -b, -m, and -r options.

-v
Gives verbose output. Displays what state the file tag is in, and what state the user is trying to change it to. This option is only useful for the -t, -b,-m, -r and -c options. The output is displayed in the following format:

<table>
<thead>
<tr>
<th>txtflag</th>
<th>Char Set Name</th>
<th>Char Set Type</th>
<th>--- &gt;</th>
<th>txtflag</th>
<th>Char Set Name</th>
<th>Char Set Type</th>
<th>Filename</th>
</tr>
</thead>
</table>

If a code set name is not associated with the numeric CCSID in the file tag, the numeric CCSID is presented instead. The following example is a sample of the output you might see:

chtag -mvc IBM-1047 file3.c

where:
A
ASCII
E
EBCDIC
?
Unknown

Examples

1. To specify a text file with IBM-1047 code set, issue:

   chtag -tc IBM-1047 filename

2. To specify a binary file, issue:

   chtag -b filename

3. To specify a file of mixed binary and text data, with a new code set of ISO8859-1, issue:

   chtag -mc ISO8859-1 filename
4. To remove the tag from a file issue:

   chtag -r filename

Usage notes

1. Table 5 on page 143 illustrates how the different combinations of txtflag and coded character set / CCSID affect a file's candidacy for automatic conversion. txtflag indicates whether this field is turned ON, OFF, binary, or untagged. txtflag indicates whether the stored coded character set is valid, invalid, or does not exist. Candidate for automatic conversion indicates whether this file is a candidate for automatic conversion.

   Table 5: Possible txtflag and CCSID combinations

<table>
<thead>
<tr>
<th>txtflag</th>
<th>Coded character set / CCSID</th>
<th>Candidate for automatic conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (on)</td>
<td>Defined</td>
<td>Yes (text file)</td>
</tr>
<tr>
<td>t (off)</td>
<td>Defined</td>
<td>No</td>
</tr>
<tr>
<td>b (off)</td>
<td>--</td>
<td>No</td>
</tr>
<tr>
<td>m (off)</td>
<td>Defined</td>
<td>No (mixed data)</td>
</tr>
<tr>
<td>-- (off)</td>
<td>--</td>
<td>No</td>
</tr>
</tbody>
</table>

2. The tagging of the following files is ignored:
   - /dev/null
   - /dev/random
   - /dev/urandom
   - /dev/zero

Exit values

0
   Successful completion.

1
   chtag failed to change the tag of a specified file for the following reasons:
   - Calling process does not have appropriate privileges to change file attributes.
   - An invalid txtflag / coded character set combination was issued.

2
   Incorrect command line syntax.

Related information

extattr, iconv, ls

**cksum - Calculate and display checksums and byte counts**

Format

cksum [-cprT] [file ... file ...]
Description

csum calculates and displays a checksum for each input file. A checksum is an error-checking technique used by many programs as a quick way to compare files that have been moved from one location to another to ensure that no data has been lost. It also displays the number of 8-bit bytes in each file.

If you do not specify any files on the command line, or if you specify - as the file name, csum reads the standard input.

When _UNIX03 is YES, the csum output has the space-separated form:

<table>
<thead>
<tr>
<th>checksum bytecount filename</th>
</tr>
</thead>
</table>

When _UNIX03 is unset or not YES, the csum output has the tab-separated form:

<table>
<thead>
<tr>
<th>checksum</th>
<th>bytecount</th>
<th>filename</th>
</tr>
</thead>
</table>

If the file operand is not specified, the path name and its leading white space is omitted.

Read error messages are controlled by the _UNIX03 variable.

- If csum fails with a read error and _UNIX03 is YES, it sends a diagnostic message to standard error, and does not show a checksum for that file.
- If _UNIX03 is unset or not YES, csum displays the checksum up to that point and marks the output line with FSUM6199 [read error].

csum continues processing files in either case.

All other error messages are sent to standard error.

Options

csum can calculate checksums in a variety of ways. The default is compatible with the POSIX standard. You can specify other algorithms with the following options. The POSIX standard does not recognize these algorithms; they are provided for compatibility with the UNIX sum command.

- \(-c\)
  Uses a standard 16-bit cyclic redundancy check (CRC-16).

- \(-i\)
  Uses the CCITT standard cyclic redundancy check (CRC-CCITT). Data communication network protocols often use a cyclic redundancy check to ensure proper transmission. This algorithm is more likely to produce a different sum for inputs—the only difference is byte order.

- \(-p\)
  Uses the POSIX checksum algorithm. This is the default.

- \(-r\)
  Enables the use of an alternate checksum algorithm that has the advantage of being sensitive to byte order.

- \(-t\)
  Produces a line containing the total number of bytes of data read as well as the checksum of the concatenation of the input files.

- \(-T\)
  Enables the automatic conversion of tagged files.

Localization

csum uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
Environment variables

• _UNIX03

For more information about the effect of _UNIX03 on the cksum command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Exit values

0  Successful completion.

1  Failure due to any of the following reasons:
   • Inability to open input file.
   • An error reading the input file.
   • Error turning off the automatic conversion of the input file.

2  Unknown command-line option.

Portability

POSIX.2, X/Open Portability Guide.
All the listed options are extensions of the POSIX standard.

Related information
cmp, diff, ls, sum, wc

clear - Clear the screen of all previous output

Format
clear

Description
The clear command clears the screen of all output and places the cursor at the top of the screen.

Localization
clear uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH
Environment variables

clear uses the following environment variables:

**TERM**
Contains the current terminal type.

**TERMINFO**
Contains the terminal information database, if different than the default.

Exit values

0
Successfully cleared the screen according to the current terminal's characteristics.

1
The terminal definition does not define a clear capability.

2
Syntax error.

3
The terminal definition specified by TERM is invalid.

4
Invalid terminfo capability.

Related information
tput

cmp - Compare two files

Format

cmp [-Bblsx] [-W option[option] ... file1 file2 [seek1 [seek2]]

Description

cmp compares two files. If either file name is -, cmp reads the standard input (stdin) for that file. By default, cmp begins the comparison with the first byte of each file. If you specify either seek1 or seek2 (or both), cmp uses it as a byte offset into file1 or file2 (respectively), and comparison begins at that offset instead of at the beginning of the files. The comparison continues, one byte at a time, until a difference is found. At that point, the comparison ends and cmp displays the byte number (in decimal) and the differing bytes (in octal format) for each difference occurred. cmp numbers bytes and lines beginning with 1.

Options

-B
Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-b
Compares single blocks at a time. Typically, cmp reads large buffers of data into memory for comparison.

-l
Causes the comparison and display to continue to the end. However, cmp does not attempt any resynchronization. cmp displays the byte number (in decimal) and the differing bytes (in octal format) for each difference found.

-s
Suppresses output and returns a nonzero status if the files are not identical.
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

```
filecodeset=codeset
```
Performs text conversion from one code set to another when reading from the file. The coded character set of the file is `codeset`. `codeset` can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The `filecodeset` and `pgmcodeset` options can be used on files with any file tag.

If `pgmcodeset` is specified but `filecodeset` is omitted, then the default program code set is `IBM-1047` even if the file is tagged with a different code set. If neither `filecodeset` nor `pgmcodeset` is specified, text conversion will not occur unless automatic conversion is enabled or the `_TEXT_CONV` environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

When specifying values for `filecodeset`, use the values that Unicode Service supports. For more information about supported code sets, see `z/OS Unicode Services User’s Guide and Reference`.

```
pgmcodeset=codeset
```
Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is `codeset`. `codeset` can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). The command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The `filecodeset` and `pgmcodeset` options can be used on files with any file tag.

If `filecodeset` is specified but `pgmcodeset` is omitted, then the default program code set is `IBM-1047`. If neither `filecodeset` nor `pgmcodeset` is specified, text conversion will not occur unless automatic conversion is enabled or the `_TEXT_CONV` environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If `filecodeset` or `pgmcodeset` is specified, then automatic conversion is disabled for this command invocation and the `-B` option is ignored if it is also specified. See `z/OS UNIX System Services Planning` for more information about automatic conversion.

```
Restriction: The only supported values for `pgmcodeset` are `IBM-1047` and `1047`.
```

```
-x
```
Displays the differing bytes shown by the `-l` option in hexadecimal format. Typically, `cmp` displays them in octal format.

**Examples**

1. To compare two files and display the first byte and line number of the difference:

   ```
   cmp myFile01 myFile02
   ```

2. To compare two text files that contain ASCII characters and display all the differences by byte number and octal byte format in EBCDIC, assuming that:

   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file):

   ```
   cmp -l -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047
   myAsciiFile01 myAsciiFile02
   ```
3. To compare two files and display the first byte and line number of the difference, assuming that automatic conversion has been enabled but the files are incorrectly tagged as UTF-8:

```bash
cmp -B myMisTaggedFile01 myMisTaggedFile02
```

**Localization**

`cmp` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Environment variables**

`cmp` uses the following environment variable:

`_TEXT_CONV`

Contains text conversion information for the command. The text conversion information is not used when either the `-B` option or the `filecodeset` or `pgmcodeset` option (`-W` option) is specified. For more information about text conversion, see [Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993](#).

**Exit values**

0

The files were identical.

1

The files were not identical.

2

Failure due to any of the following reasons:

- The command-line option is not correct.
- The code set is not valid.

3

Failure due to any of the following reasons:

- An error opening or reading an input file.
- Could not turn off automatic conversion.
- Could not perform requested text conversion.

**Messages**

Possible error messages include:

**EOF on filename**

`cmp` reached the end of the file on the specified file before reaching the end of the file on the other file.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

The `-B`, `-b`, `-W` and `-x` options and the `seek` pointers are extensions of the POSIX standard.
The `col` utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, do not use this utility for applications intended to be portable to other UNIX-branded systems.

**Description**

`col` processes control characters for vertical line feeds and writes the processed text to the standard output. If you do not specify any files, `col` reads from the standard input. Otherwise, `col` will read and process each specified file in sequence. It is intended to be used as a filter between a program such as `nroff` and an output device that cannot handle reverse line feeds.

Where possible, blank characters (spaces) are converted to tabs; tab stops are assumed to be every eight characters.

`col` also removes all escape sequences except for those shown in the following table. ESC is the ASCII escape character, octal code 033.

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII control character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backspace</td>
<td>010</td>
</tr>
<tr>
<td>Carriage-return</td>
<td>015</td>
</tr>
<tr>
<td>Newline</td>
<td>012</td>
</tr>
<tr>
<td>Vertical tab</td>
<td>013</td>
</tr>
<tr>
<td>SO</td>
<td>016</td>
</tr>
<tr>
<td>SI</td>
<td>017</td>
</tr>
<tr>
<td>Space</td>
<td>040</td>
</tr>
<tr>
<td>Tab</td>
<td>011</td>
</tr>
<tr>
<td>Reverse line feed</td>
<td>ESC-7</td>
</tr>
<tr>
<td>Reverse half-line feed</td>
<td>ESC-8</td>
</tr>
<tr>
<td>Forward half-line feed</td>
<td>ESC-9</td>
</tr>
</tbody>
</table>

The ASCII control characters SO and SI denote the beginning and end of text in an alternative character set. The set of each input character is remembered. `col` generates SO and SI characters as needed to output each character in the correct character set.

**Options**

- `-b`
  Ignores backspace (CRTL-H) characters. If two characters are supposed to appear in the same space, the first character is ignored and the second is output.

- `-f`
  Allows forward half-line motions. Typically, these are changed to forward full-line motions.
-x
Prevents conversion of spaces to tab characters.

Localization
col uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH

Usage notes
1. Because col ignores vertical motions that back up over the first line, you might get unexpected results if the first line contains superscripts.
2. Because -f allows escape sequences, it might cause unexpected results on terminals.

Exit values
0
Successful completion
1
Failure due to any of the following reasons:
• Incorrect command-line option.
• Insufficient memory.

Portability
UNIX systems.
This implementation does not handle double-byte characters.

: (colon) - Do nothing, successfully

Format
: [argument ...]
tcsh shell: :

Description
The : (colon) command is used when a command is needed, as in the then condition of an if command, but nothing is to be done by the command. This command simply yields an exit status of zero (success). This can be useful, for example, when you are evaluating shell expressions for their side effects.
colon is a special built-in shell command.
In the tcsh shell, colon performs as indicated for the z/OS version of : (colon).

Examples
: ${VAR="default value"}
sets VAR to a default value if and only if it is not already set.

**Localization**

colon uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Exit values**

Because this command always succeeds, the only possible exit status is:

- **0**
  - Successful completion.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

**Related information**

sh, tcsh, true

---

**comm - Show and select or reject lines common to two files**

**Format**

```
comm [-B123] [-W option[option] ... file1 file2
```

**Description**

comm locates identical lines within files that are sorted in the same collating sequence, and produces three columns; the first contains lines found only in the first file, the second lines only in the second file, and the third lines that are in both files. If you specify - in place of either file1 or file2, comm reads from the standard input (stdin).

**Options**

- **-1**
  - Suppresses lines that appear only in file1.

- **-2**
  - Suppresses lines that appear only in file2.

- **-3**
  - Suppresses lines that appear both in file1 and file2.

- **-B**
  - Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

- **-W option[option]...**
  - Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

    **filecodeset=codeset**
    - Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name that is known to the system or...
a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The `filecodeset` and `pgmcodeset` options can be used on files with any file tag.

If `pgmcodeset` is specified but `filecodeset` is omitted, then the default file code set is `ISO8859-1` even if the file is tagged with a different code set. If neither `filecodeset` nor `pgmcodeset` is specified, text conversion will not occur unless automatic conversion is enabled or the `_TEXT_CONV` environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If `filecodeset` or `pgmcodeset` is specified, then automatic conversion is disabled for this command invocation and the `-B` option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

If you are specifying values for `filecodeset`, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User's Guide and Reference.

`pgmcodeset=codeset`

Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is `codeset`. `codeset` can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The `filecodeset` and `pgmcodeset` options can be used on files with any file tag.

If `filecodeset` is specified but `pgmcodeset` is omitted, then the default program code set is `IBM-1047`. If neither `filecodeset` nor `pgmcodeset` is specified, text conversion will not occur unless automatic conversion is enabled or the `_TEXT_CONV` environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If `filecodeset` or `pgmcodeset` is specified, then automatic conversion is disabled for this command invocation and the `-B` option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

Restriction: The only supported values for `pgmcodeset` are `IBM-1047` and `1047`.

The options suppress individual columns. Thus, to list only the lines common to both files, use:

```
comm -12
```

To find lines unique to one file or the other, use:

```
comm -3
```

Observe that `comm -123` displays nothing.

**Examples**

1. To display the lines that are unique to each text file and the lines that are common to both text files:

```
comm myFile01 myFile02
```

2. To display the lines that are unique to a text file that contains UTF-8 characters, assuming that

   - The text files are untagged and you do not want to tag them or enable automatic conversion, and
   - You cannot alter the tag (for example, you are comparing untagged public text files or read-only text files)

then issue:

```
comm -23 -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File01 myUtf8File02
```
3. To display the lines that are common to both text files that contain EBCDIC characters, assuming that automatic conversion was enabled but the files are incorrectly tagged as ASCII:

```
comm -12 -B myMisTaggedFile01 myMisTaggedFile02
```

**Localization**

*comm* uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Environment variables**

*comm* uses the following environment variable:

**_TEXT_CONV**

Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

**Exit values**

0  
Successful completion.

1  
Failure due to any of the following reasons:

  - The code set is not valid.
  - Could not turn off automatic conversion.
  - Could not perform requested text conversion.

2  
Failure that generated a usage message, such as naming only one input file.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

The -B and -W options are extensions of the POSIX standard.

**Related information**

cmp, diff, sort, uniq

---

**command - Run a simple command**

**Format**

- `command [-p] command-name [argument...]`
- `command [-V|-v] command-name`
**Description**

`command` causes the shell to suppress its function lookup and execute the given `command` and arguments as though they made up a standard command line. In most cases, if `command-name` is not the name of a function, the results are the same as omitting command. If, however, `command-name` is a special built-in command, (see `sh`), some unique properties of special built-in commands do not apply:

- A syntax error in the command does not cause the shell running the command to stop.
- Variable assignments specified with the special built-in command do not remain in effect after the shell runs the command.

**Options**

- `-p`
  Searches for `command-name` using the system default PATH variable.

- `-v`
  Writes a string indicating the path name or command that the shell uses to invoke `command-name`.

- `-V`
  Writes a string indicating how the shell interprets `command-name`. If `command-name` is a command, a regular built-in command, or an implementation-provided function found using the PATH variable, the string identifies it as such and includes the absolute path name. If `command-name` is an alias, function, special built-in command, or reserved word, the string identifies it as such and includes its definition if it is an alias. If the command is a tracked alias, the string identifies it as cached.

**Examples**

Typically, you use `command` when you have a command that might have the same name as a function. For example, here’s a definition of a `cd` function that not only switches to a new directory but also uses `lc` to list the contents of that directory:

```bash
function cd {
    command cd $1
    lc
}
```

Inside the function, use `command` to get at the real `cd`. Otherwise, the `cd` function would call itself in an infinite recursion.

**Localization**

`command` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Usage notes**

`command` is a built-in shell command.

**Exit values**

If you specified `-v`, possible exit status values are:

0
  Successful completion
1 command could not find command-name, or an error occurred

2 Failure due to incorrect command-line argument

If you did not specify -v, possible exit status values are:

126 command found command-name, but failed to invoke it.

127 An error occurred in the command or it could not find command-name.

Otherwise, the exit status of command is the exit status of command-name.

Portability

POSIX.2.

Related information

sh

compress - Lempel-Ziv file compression

Format

compress [-cDdfVv] [-b bits] [file ...]

Description

compress compresses each input file by using Lempel-Ziv compression techniques. If you do not specify any input files, compress reads data from standard input (stdin) and writes the compressed result to standard output (stdout).

The output files have the same names as the input files but with a .Z suffix. For example, abc is compressed into abc.Z. If the .Z file already exists and you did not specify the -f option, compress gives an error and asks whether it should overwrite the existing file.

compress uses the modified Lempel-Ziv algorithm described in A Technique for High Performance Data Compression, Terry A. Welch, IEEE Computer, vol. 17, no. 6 (June 1984), pp.8-19. compress first replaces common substrings in the file by 9-bit codes starting at 257. After it reaches code 512, compress begins with 10-bit codes and continues to use more bits until it reaches the limit set by the -b option.

After attaining the bits limit, compress periodically checks the compression ratio. If it is increasing, compress continues to use the existing code dictionary. However, if the compression ratio decreases, compress discards the table of substrings and rebuilds it from scratch. This allows the algorithm to compensate for files, such as archives, where individual components have different information content profiles.

Options

-b bits

Limits the maximum number of bits of compression to bits. The value bits can be an integer from 9 to 16. The default is 16.

-c

Writes the output to stdout. When you use this option, you can only specify one file on the command line.
-D
   Allows an extra degree of compression to be done for files such as sorted dictionaries where
   subsequent lines normally have many characters in common with the preceding line.

-d
   Decompresses argument files instead of compressing them. This works by overlaying the compress
   program with the uncompress program. For this to work, uncompress must be available somewhere
   in your search path (given by the PATH environment variable). Decompressing files this way is slower
   than calling uncompress directly.

-f
   Forces compression even if the resulting file is larger or the output file already exists. When you do not
   specify this option, files that are larger after compression are not compressed. compress does not
   print an error message if this happens.

-V
   Prints the version number of compress.

-v
   Prints statistics giving the amount of compression achieved. Statistics give the name of each file
   that is compressed and the compression ratio, expressed as a percentage. If the file resulting from
   compression is larger than the original, the compression ratio is negative.

Localization
compress uses the following localization environment variables:
- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

Exit values
0
   Successful completion.
1
   Failure due to one of the following reasons:
   - Missing number of bits after the -b option.
   - Incorrect number of bits specified.
   - Failed to execute uncompress.
   - Unknown option.
   - Dictionary option: same count of string exceeded.
   - Output path or file name too long.
   - Cannot stat file.
   - Argument file not a regular file: unchanged.
   - Argument file has other links: unchanged.
   - No space for compression tables.
2
   One or more files were not compressed because the compressed version was larger than the original.
Limits
This implementation of compress is limited to a maximum of 16-bit compression.

Portability
A binary-compatible version of compress with more options is often found on UNIX systems.
The -D option is an extension to traditional implementations of compress. The -D, -d and -v options are extensions of the POSIX standard.
For portability, you should restrict the number of bits in the code (–b option) to a value between 9 and 14.

Related information
cpio, pack, pax, tar, uncompress, unpack, zcat

confighfs - Invoke the vfs_pf I wsctl function for HFS file systems

Format
confighfs [-l] [-v n] [-f n] [-q] [pathname] [-x[n] size pathname]

Note: The -l option signifies a lowercase -L, not an uppercase i.

Description
confighfs gives interactive shell users the ability to invoke the vfs_pfsctl function. The vfs_pfsctl function is used to pass control information to the PFS (physical file system). For more information about vfs_pfsctl, see the z/OS UNIX System Services File System Interface Reference. Detailed information about its use can be found in z/OS DFSMS Using Data Sets.
confighfs resides in the following directory: /usr/lpp/dfsms/bin/. This directory is not part of the default search path definition. Therefore, the directory must be included in the command specification when invoking the command.
You can only use the confighfs command when working with the DFSMS file system (HFS).
For the zFS file system, use the zfsadm command. For more information about the zFS file system, see z/OS Distributed File Service zFS Administration.

Options
-l
Query HFS limits.

Note: 1 signifies a lowercase L, not an uppercase i.
-v n
Set virtual storage max to n (where n is in MB). Requires superuser authority.
-f n
Set fixed storage min to n (where n is in MB). Requires superuser authority.
-q
Query your global statistics.
pathname
Query file system statistics for the file system containing each of the path names specified.
-x size pathname
Extend the specified file system, where size is the amount to be extended suffixed by the extend unit of M, T, or C (for megabytes, tracks, or cylinders), and the path name is a full or simple path name to a file or directory in the file system to extend. Requires superuser authority.
-xn size pathname
    Extend the specified file system to a new volume using the -x rules. Requires superuser authority.

The following are internal debug options:

-dn
    Prints incoming and outgoing pfsctl buffers (where n is 0, 1, or 2).

-t
    Skips issuing the pfsctl.

Examples

On systems running shared file systems, this command should only be issued on the server system (file system owner) for the file system pointed to by the path name. Issuing it on client systems results in fields of zeros.

1. To set virtual and fixed buffer limits for the HFS file system:

   > confighfs -v 128 -f 32

2. To extend the file system for your current directory 100 cylinders:

   > confighfs -x 100c .

3. If you need to get statistics for the root file system and the file system mounted over /tmp, you would do the following:

   > confighfs / /tmp .

   The period in examples two and three indicates the current directory.

Usage notes

1. If the HFS file system encounters an out of space condition during SYNC processing producing message IGW022S, then the following can result:

   a. If confighfs is used to successfully extend the file system (by specifying confighfs -x size pathname, for example) and the extent was large enough to accommodate the pages required to complete the SYNC processing, confighfs invokes the SYNC function again to complete its update and then resets the Out of Space error flag. It will no longer be necessary to unmount and remount the file system to use it further. After the error flag is reset, all file system functions will work properly again.

   b. If the extend size is not large enough to provide the amount of space required to complete the SYNC process, confighfs issues the following response:

   Inadequate space added to HFS. At least another nn tracks required.

   These results only apply when the IGW022S message indicates an Error Loc: EXTEND value. If it indicates an Error Loc: ARPN value, it goes into the out of space error state and requires an unmount followed by a mount to reset the error condition and make it reusable. The updates applied to the HFS since the last successful SYNC will also be lost.

2. Unlike most z/OS UNIX commands, which are located in /bin, confighfs is found in the /usr/lpp/dfsms/bin directory. You can symbolically link to the actual location of confighfs. The symbolic link is found in /usr/sbin:

   /usr/sbin/confighfs -> /usr/lpp/dfsms/bin/confighfs
configstk - Configure the AF_UEINT stack

Format

configstk [-s] Configuration_file_name

Description

configstk is used to configure the AF_UEINT stack. This command should initially be run from the /etc/rc script, which is run when z/OS UNIX System Services is initialized. It should also be run each time the AF_UEINT network topology changes after z/OS UNIX services have been initialized. This command requires superuser authority.

Options

-s

Does syntax checking only.

Files

configstk uses the following file:

Configuration_file_name

Specifies the configuration for the AF_UEINT stack. As with any system-wide configuration file, it should have the appropriate permissions set.

This file has two types of specifications, HOME and GATEWAY. Be careful when modifying the configuration file to ensure that the F_UEINT environment is not corrupted due to user error.

HOME ip_address BUFFERS(number) blocking

This statement is required but you can only specify it once. The entire statement must be on a single line.

ip_address

Defines the single virtual IP address to be used by all RS/6K clients when accessing the z/OS host, independent of how many RS/6K gateways are connected to a given z/OS image. This implementation differs from the standard IP model, which defines an IP address per physical adapter.

number

Defines the maximum number of 32 K page-fixed buffers (in OMVS private memory) that are to be used by the protocol stack. The number specified is be distributed equally among the read and write flows. As new ESCON fibers are added to the configuration, additional IO buffers are required. Thruput decreases and overhead increases if the number specified is too restrictive. You should initially specify a value of 10 times the number of defined gateways for low-to-average use and increase it proportionally as the number of users increase). The maximum number of buffers that are allocated is the larger of six times the number of active gateways, or the number specified. A decrease in the number is not honored until the next IPL.

blocking

Indicates whether the internal blocking algorithm should be activated for outgoing packets. The default is BLOCKING. Specifying NOBLOCKING causes the internal optimization routines, which attempt to group multiple packets into a single blocked I/O, to be bypassed (such as single packet per block written on demand). Specifying BLOCKING minimizes the z/OS overhead and maximizes the ESCON channel bandwidth, but can delay the packet delivery slightly.

GATEWAY device_number checksum

At least one of these statements is required and up to 32 can be specified. The entire statement must be on a single line. This statement maps the target RS/6K IP addresses to the gateway that will process the request. The device number to define the gateway must be the first of an even-odd pair of subchannels (both configured thru a single ESCON fiber) between the z/OS image and the RS/6K.
gateway. Multiple target IP addresses can be mapped to a given gateway. A given target IP address can be mapped to at most one gateway.

**device_number**
Specifies the hexadecimal address of device to be configured. This number must be four hexadecimal digits and must be an even number.

**checksum**
Indicates whether a reliable communications path exists between the communicating applications. Specify CHECKSUM if any portion of the path between the communicating applications is unreliable (such as a LAN). Specify NOCHECKSUM if the entire path is reliable (such as an SP2 fast switch or ESCON).

A list of IP addresses immediately follows this statement, one IP address per line. At least one IP address must be specified for each gateway device. Up to 256 IP addresses can be specified in the configuration file.

Blank lines are permitted and lines beginning with /* are treated as comments.

**Examples**

```plaintext
/* configure AF_UEINT sockets
/* name the ip address for this node, default to blocking enabled
home 10.32.166.20 buffers(20)

/* configure device 324
gateway 0324 nochecksum
10.34.166.20
10.34.166.24
10.34.166.26

/* configure device b28
gateway 0b28 checksum
10.36.166.20
10.36.166.22
10.36.166.24
10.36.166.26
```

**configstrm - Set and query the STREAMS physical file system configuration**

**Format**

```
configstrm [-bimv] [-h high_mem | ?] [-l loadmod] ... [-t trace_opt | ?] ... [-u loadmod]
```

**Note:** The l option signifies a lowercase L, not an uppercase i.

**Description**

`configstrm` sets and queries the STREAMS physical file system configuration. It can be used to view statistics and change configuration options for the STREAMS physical file system without changing your BPXPRMxx member and reIPLing.

**Options**

- **-b**
  Print current buffer pool utilization.

- **-h high_mem**
  Set and query the maximum allowed storage utilization and query the current utilization. `high_mem` is specified in kilobytes.

- **-i**
  Print internal diagnostic information.
-l loadmod
   Load a new device driver set.

   **Note:** l signifies a lowercase L, not an uppercase l.

-m
   Print device major information.

-t trace_opt
   Set and query trace options. The valid trace options are:
   
   all | none
      Enables or disables all trace points.
   
   proc | noproc
      Enables or disables procedure entry and exit trace points.
   
   data | nodata
      Enables or disables data trace points.
   
   nw | nonw
      Enables or disables NetWare trace points.
   
   code | nocode
      Enables or disables code trace points.
   
   diag | nodiag
      Enables or disables diagnostic trace points.

-u loadmod
   Unload a device driver.

-v
   Avoid output truncation when information is excessive.

Usage notes
1. Must be a superuser to use the **configstrm** command.
2. **configstrm** can be used to dynamically configure the physical file system for NetWare.

Examples
To display device information for the configured STREAMS device drivers, issue:

```bash
configstrm -m
```

**continue** - Skip to the next iteration of a loop in a shell script

Format
continueln

Description
**continue** skips to the next iteration of an enclosing `for`, `select`, `until`, or `while` loop in a shell script. If a number n is given, execution continues at the loop control of the nth enclosing loop. The default value of n is 1.

Usage notes
**continue** is a special built-in shell command.
Localization

continue uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


Exit values

0
  Successful completion.

1
  The value of \( n \) given was not an unsigned decimal greater than 0.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

Related information

break, sh, tcsh

copytree - Make a copy of a file hierarchy while preserving all file attributes

Format

/samples/copytree [-afos] sourcedir [targetdir]

Description

copytree is a REXX sample that enables you to use a number of z/OS UNIX capabilities. Included is a recursive routine to descend a hierarchical directory. You can also use it to accomplish the following tasks:

- Retrieve and set attributes for files.
- Read and write files.
- Read and set access control lists (ACLs).

copytree replicates a source tree starting under the source directory within a file system to a target directory. It:

- Tolerates errors when setting target attributes with messages.
- Tolerates errors in the source tree, skipping those files.
- Copies sparse files as sparse files.
- Handles both symbolic links and external links
- Does not cross mount points.
- Preserves file links.

copytree is installed in the z/OS UNIX file system. Run it as /samples/copytree.

To run it under TSO, copy /samples/copytree to a PDS where REXX execs can be run, or in a PDS to run under TSO.
Restriction: copytree cannot handle files greater than 1 GB in TSO/E.
Copy every attribute that can be set if you have sufficient authorization.

Options
Any combination of the option flags can be used, with no spaces between flags.

-a
   Specifies that the 30,000 node limit warning is not to be issued.
-f
   Specifies that full file reads are to be done when copytree is run in check mode.
-o
   Specifies that file ownership is not to be preserved.
-s
   Specifies that the effective UID is to be set to 0 before copytree is started.

<sourcedir>
The path name for the source directory where the copy begins. The path name must be used, not the file system name.

<targetdir>
The path name for the target directory. This directory must exist and must be empty. The permissions and other attributes of the target directory are not modified to be the same as the source directory. If <targetdir> is not specified, copytree runs in a mode to check the source file tree.

Exit values
0
   Successful completion
>0
   An error occurred
Any other value means that there were errors.

Related information
pax

cp - Copy a file

Format
If the variable _UNIX03=YES is set:

```bash
cp [-ACcimMUv] [-p |-F format|-B|-T|-X[-I]] [-s suffix] [-Z] [-O -u | c=odeset] file ... file ... directory
cp -r [-H|-L|-P] [-cfimp] [-Z] [-O -u | c=odeset] source... directory
cp -r [-H|-L|-P] [-cfimp][-Z] [-O -u | c=odeset] source... directory
```
If the variable _UNIX03 is unset or not YES:

```bash
cp [-ACcimMUv] [-p|F format|-B|-T|-X[-I]] [-S suffix] [-Z] [-O -u | -c=odeset] codeset file ... file ... directory
```
**Description**

`cp` copies files to a target named by the last argument on its command line. If the target is an existing file, `cp` overwrites it; if it does not exist, `cp` creates it. If the target file exists and does not have write permission, `cp` denies access and continues with the next copy.

If you specify more than two path names, the last path name (that is, the target) must be a directory. If the target is a directory, `cp` copies the sources into that directory with names that are given by the final component of the source path name.

You can also use `cp` to copy files to and from MVS data sets. If you specify more than one file to be copied, the target (last path name on command line) must be either a directory or a partitioned data set. If the target is an MVS partitioned data set, the source cannot be a UNIX directory.

`cp` does not support the copying to or from generation data groups (GDGs). To use those MVS data sets, the user must specify the real data set name. `cp` also does not support copying to a temporary PDSE.

When copying records, the string " \n" is copied the same way as the string "\n": both are read back as "\n", where "\n" indicates that z/OS XL C++ will write a record containing a single blank to the file (the default behavior of z/OS XL C/C++). All other blanks in your output are read back as blanks, and any empty (zero-length) records are ignored on input. However, if the environment variable _EDC_ZERO_RECLLEN is set to Y before `cp` is called, an empty record is treated as a single newline character and is not ignored. Also, if _EDC_ZERO_RECLLEN is set to Y, a single newline character is written to the file as an empty record, and a single blank will be represented by " \n".

You can copy:

- One file to another file in the working directory.
- One file to a new file on another directory.
- A set of directories and files to another place in your file system.
- A UNIX file to an MVS data set.
- An MVS data set to a file system.
- An MVS data set to an MVS data set.

**Options**

-A

Specifies that all suffixes (from the first period to the end of the target) be truncated. -A has precedence over -M and -C options. -s will be turned off if -A is the last option specified.

-B

Specifies that the data to be copied contains binary data. When you specify -B, `cp` operates without any consideration for <newline> characters or special characteristics of DBCS data. (This type of behavior is typical when data is copied across a UNIX system.) Because -B is mutually exclusive with -F, -x, and -T, you will get an error if you specify more than one of these options.

-C

Specifies truncating the file name to 8 characters to meet the restriction in the MVS data set member.

-c (UNIX to UNIX only)

Prompts you to change the disk if there is not enough room to complete a copy operation. This option has no effect on systems without disk drives. The parent directories must exist on the new target diskette.

-F format

Specifies whether the file is to be treated as binary, text, or record file format when copied; for text files, specifies the end-of-line delimiter. Also sets the file format to `format` if the target is a UNIX file. For text files, when copying from UNIX to MVS, the end-of-line delimiter is stripped. When copying from MVS to UNIX, the end-of-line delimiter is added. (Code page IBM-1047 is used to check for end-of-line delimiters.) Record file format files are treated as if they were binary files.

If -F is used when copying from UNIX to UNIX, `cp` sets only the target file format and does not replace the end-of-line delimiters.
If setting format fails, a warning is displayed but `cp` will continue to copy any remaining files that were specified to be copied.

-F is mutually exclusive with -B, -x, -p, and -T. If you specify one of these options with -F, you will get an error. If -F is specified more than once, the last -F specified is used.

For format, you can specify:

- **not**
  - Not specified.

- **bin**
  - Binary data.

- **rec**
  - Record. (File data consists of records with prefixes. The record prefix contains the length of the record that follows. From the shell command perspective, files with this format will be treated as if they were binary files.)

Or, you can specify the following text data delimiters:

- **nl**
  - Newline character.

- **cr**
  - Carriage return.

- **lf**
  - Line feed.

- **crlf**
  - Carriage return followed by line feed.

- **lfcr**
  - Line feed followed by carriage return.

- **crnl**
  - Carriage return followed by new line.

-F
  - Attempts to remove and replace a UNIX destination file that cannot be opened.

-H
  - Follows symbolic links that are specified as source operand on the command line. Symbolic links that are encountered in the tree traversal are not followed. This is the default behavior when the -R or -r option is specified but none of the -H, -L or -P options are specified.

  **Restriction:** This option can only be used with the -R or the -r option.

-i
  - When copying to a UNIX target, -i asks you if you want to overwrite an existing file, whether the file is read-only.

-L
  - Follows symbolic links that are specified as source operand on the command line and those encountered in the tree traverse.

  **Restriction:** This option can only be used with the -R or the -r option.

-M
  - Specifies that some characters of the file name are converted during the copy process between a UNIX file and an MVS data set member. Characters are converted as follows:
    - _ (underscore) in UNIX is converted to @ in MVS data set members and vice versa.
    - . (period) in UNIX is converted to # in MVS data set members and vice versa.
    - - (dash) in UNIX is converted to $ in MVS data set members and vice versa.
### -F (UNIX to UNIX only)
Sets the modification and access time of each destination file to that of the corresponding source file. Typically, `cp` sets the modification time of the destination file to the present.

### -I (UNIX to MVS only)
When the specified file has an alias and the file is copied from UNIX to MVS, the alias information is also copied. If the -I option is specified when the data set that is copied from MVS to z/OS UNIX does not have an alias, that option is ignored.

**Restriction:** The -I option can only be used with the -X option.

### -O u | c=odeset
Allow automatic conversion on source and target files.

- **-O u**
  If the target exists and is not empty or tagged, `cp` will not change the target's tag in order for the target to be a candidate for automatic conversion.

  For new targets and existing, untagged, empty files, this option has no effect and `cp` behaves the same as the default. For a description of the default behavior, see Automatic conversion and file tagging behavior for `cp`.

  If you are using `cp` to copy from a UNIXan MVS file to a data set, if the source is a tagged text file, then it might be a candidate for automatic conversion.

  If you are copying executables from or to MVS, the automatic conversion is disabled for both source and target.

- **-O c=odeset**
  For a detailed description of the behavior of this option on `cp`, see Automatic conversion and file tagging behavior for `cp`.

  `odeset` can be a code set name known to the system or the numeric coded character set identifier (CCSID). If a code set name exists, the numeric CCSID associated with that name is used. Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names.

  To prevent the corruption of text files, `cp` will fail if it cannot set the tag to text or code set.

  If automatic conversion is not set properly or if the source is not tagged properly, the target might end up with a tag code set that does not match the file content.

### -P
If _UNIX03 is YES, does not follow any symbolic links, neither those specified as source operand on the command line not those encountered in the tree traverse.

**Restriction:** This option can only be used with the -R or the -r option.

### -P params
If _UNIX03 is unset or not YES, then the -P option will be treated as specifying parameters needed to create a new sequential data set if one does not exist. You can specify the RECFM, LRECL, BLKSIZE, and SPACE in the format the CRTL fopen() function uses.

**SPACE=(units,(primary,secondary))** where the following values are supported for units:

- Any positive integer indicating BLKSIZE.
- CYL (mixed case).
- TRK (mixed case).

For example:

```
SPACE=(500,(100,500)) units, primary, secondary
SPACE=(500,100) units and primary only
```

For information about how to specify these parameters, see `z/OS XL C/C++ Programming Guide`. 

CRTL fopen() arguments: LRECL specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records. BLKSIZE specifies the maximum length, in bytes, of a physical block of records. RECFM refers to the record format of a data set and SPACE indicates the space attributes for MVS data sets.

-p (UNIX to UNIX only)
Preserves the modification and access times (as the -m option does). In addition, it preserves the file mode, file format, owner, and group owner, if authorized. It also preserves extended attributes. It preserves the ACLs of files and directories, if possible. The ACLs are not preserved if a file system does not support ACLs.

-p is mutually exclusive with -F. If you specify both, you will get an error message.

-R (UNIX to UNIX only)
Clones the source trees. cp copies all the files and subdirectories that are specified by source... into directory, making careful arrangements to duplicate special files (FIFO, character special). cp only follows symbolic link specified as source operand on the command line.

-r (UNIX to UNIX only)
Clones the source trees, but makes no allowances for special files (FIFO, character special). Consequently, cp attempts to read from a device rather than duplicate the special file. This is similar to, but less useful than, the preferred -R.

-s d=suffix|a=suffix
- d=suffix removes the specified suffix from a file.
- a=suffix appends the specified suffix to a file.

-S has precedence over -M and -C. It also turns off the -A option (if -S is the last specified option).

-T
Specifies that the data to be copied contains text data. See “Usage notes” on page 173 for details on how to treat text data. This option looks for IBM-1047 end-of-line delimiters, and is mutually exclusive with -F, -X, and -B. That is, you will get an error if you specify more than one of these options.

-T is ignored when you are copying data across UNIX file systems.

-U
Keeps file names in uppercase when copying from MVS data set members to UNIX files. The default is to make file names lowercase.

-v
Verbose

-W seqparms=params
Specifies the parameters that are needed to create a sequential data set if one does not exist. You can specify the RECFM, LRECL, BLKSIZE, and SPACE in the format the CRTL fopen() function uses.

SPACE=(units,(primary,secondary)) where the following values are supported for units:
- Any positive integer indicating BLKSIZE.
- CYL (mixed case).
- TRK (mixed case).

For example:

SPACE=(500,(100,500)) units, primary, secondary
SPACE=(500,100) units and primary only

For information about how to specify these parameters, see z/OS XL C/C++ Programming Guide.

CRTL fopen() arguments: LRECL specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records. BLKSIZE specifies the maximum length, in bytes, of a physical block of records. RECFM refers to the record format of a data set and SPACE indicates the space attributes for MVS data sets.
This option is the same as -P params with _UNIX03 unset or not YES. If multiple -P params and -W are specified, the value of the last one specified on the command will be used.

-X
Specifies that the data to be copied is an executable. Cannot be used with -F, -T, or -B.

-Z
Specifies that error messages are not to be displayed when setting ACLs on the target. The return code will be zero. If default behavior is used to set the file tag, failure is suppressed. For a description of the default behavior, see Automatic conversion and file tagging behavior for cp.

If you do not specify either -F | B | T or X, cp will first check the format of the MVS data set indicated and then try to determine the type of file.

The -P option on cp does not affect file tagging.

Automatic conversion and file tagging behavior for cp

The following tables describe the behavior of file tagging and automatic conversion for various source and target scenarios depending on whether the -O option is specified on the cp command.

<table>
<thead>
<tr>
<th>Table 6: Copying files to files by using the default (without the -O option).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S</strong></td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>File tagging</td>
</tr>
<tr>
<td>Automatic conversion</td>
</tr>
</tbody>
</table>
Table 7: Automatic conversion and file tagging behavior for cp: Copying files to files.

<table>
<thead>
<tr>
<th>Default (without -O option)</th>
<th>With -O u option</th>
<th>With -O c=code set option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If the target file system supports setting file tags...</strong></td>
<td><strong>If the target file system does not support setting file tags (For example, NFS)...</strong></td>
<td></td>
</tr>
<tr>
<td>File tagging</td>
<td>Target file is tagged the same as the source file.</td>
<td>An existing target's tag is unchanged. A new target is created with a tag according to the file system's attributes (MOUNT parameter can specify TAG).</td>
</tr>
<tr>
<td>Automatic conversion</td>
<td>Disabled for source and target files</td>
<td>Allowed for source and target files</td>
</tr>
</tbody>
</table>

Table 8: Automatic conversion and file tagging behavior for cp: Copying MVS data sets to files.

<table>
<thead>
<tr>
<th>Default (without -O option)</th>
<th>With -O u option</th>
<th>With -O c=code set option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If the SOURCE is text:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File tagging</td>
<td>Target is set to UNTAG</td>
<td>An existing target's tag is unchanged. A new target is created with a tag according to the file system's attributes (MOUNT parameter can specify TAG).</td>
</tr>
<tr>
<td>Automatic conversion</td>
<td>Disabled for target file</td>
<td>Allowed for target file (The target file is a candidate for automatic conversion when its TXTFLAG is tagged TEXT.)</td>
</tr>
<tr>
<td><strong>If the SOURCE is binary or executable:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File tagging</td>
<td>Target is set to UNTAG</td>
<td>Target's tag is unchanged</td>
</tr>
<tr>
<td>Automatic conversion</td>
<td>Disabled for target file</td>
<td></td>
</tr>
</tbody>
</table>

Shell command descriptions 169
Table 9: Automatic conversion and file tagging behavior for cp: Copying files to MVS data sets.

<table>
<thead>
<tr>
<th></th>
<th>Default (without -O option)</th>
<th>With -O u option</th>
<th>With -O c=code set option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If the SOURCE is text or binary:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File tagging</td>
<td>Not applicable for target data set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic conversion</td>
<td>Disabled for source file</td>
<td>Allowed for source file</td>
<td>Disabled for source file</td>
</tr>
<tr>
<td></td>
<td>(The source file is a candidate for automatic conversion when its TXTFLAG is tagged TEXT.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>If the SOURCE is executable:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File tagging</td>
<td>Not applicable for target data set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic conversion</td>
<td>Disabled for source file</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Limits and requirements**

In general:

1. To specify an MVS data set name, precede the name with double slashes (/ /). For example, to specify the fully qualified data set names 'turbo.gammalib' and 'turbo.gammalib(pgm1)', write:

   ```
   //'turbo.gammalib'
   //'turbo.gammalib(pgm1)'
   ```

   The same goes for data set names that are not fully qualified:

   ```
   //turbo
   ```

2. For PDS (partitioned data set) or PDSE (partitioned data set extended), to avoid parsing by the shell, the name should be quoted or minimally, the parenthesis should be escaped. For example, to specify 'turbo(pgm1)', use quotes:

   ```
   "//turbo(pgm1)"
   ```

   or escape the parenthesis:

   ```
   //turbo\(pgm1\)
   ```

   As indicated, a fully qualified name must be surrounded with single quotation marks (as is done within TSO). To prevent the single quotation marks from being interpreted by the shell, they must be escaped or the name must be placed within regular quotation marks. See the 'turbo.gammalib' examples.

3. If you specify a UNIX file as source and the MVS data set (target) does not exist, a sequential data set will be created. If the partitioned data set exists, the UNIX file will be copied to the partitioned data set member.

4. If source is an MVS data set and target is a UNIX directory, the UNIX directory must exist.

5. You cannot have a UNIX directory, partitioned data set, or sequential data set as source if the target is a partitioned data set.

6. To copy all members from a partitioned data set, you can specify the partitioned data set as source and a UNIX directory as target.

**MVS data set naming limitations:** The naming of MVS data sets has some limitations.

- Data set names can only contain uppercase alphabetic characters (A-Z). Lowercase characters will be converted to uppercase during any copies to MVS data sets.
• Data set names can contain numeric characters 0-9 and special characters @, #, and $.
• Data set names cannot begin with a numeric character.
• A data set member name cannot be more than 8 characters. If a file name is longer than 8 characters or uses characters that are not allowed in an MVS data set name, the file is not copied. You can use the -C option to truncate names to 8 characters.

Limitations: UNIX to MVS data set: The limitations are as follows:
1. If you specify a sequential MVS data set that is in undefined record format, the file is copied as binary.
2. If you specify a PDSE that is in undefined record format, the first file successfully copied determines in what format files will be copied. Note that PDSE does not allow a mixture of formats. If the first successfully copied file is an executable, the PDSE will have program objects only and all other files will fail. On the other hand, if the first file is data, then all files are copied as binary.
3. If you specify a PDS that is in undefined record format, UNIX executables are saved as PDS load modules. All other files are copied as binary.
4. If you specify an MVS data set that is either in variable length or fixed record length and you have not set the file format, text files are copied as text, binaries as binary, and executables as binary. (IBM-1047 end-of-line delimiters are detected in the data)
5. If you set the file format, the set value is used to determine if data is binary, text, or record file format.

Limitations: MVS data set to UNIX: The limitations are as follows:
1. If an UNIX file does not exist, one is created using 666 mode value, whether the data to be copied is binary or text:

<table>
<thead>
<tr>
<th>666 mode value:</th>
<th>owner(rw-)</th>
<th>group(rw-)</th>
<th>other(rw-)</th>
</tr>
</thead>
</table>

If the data to be copied is a shell script or executable in a PDS or PDSE with undefined record format, the UNIX file is created using 777 mode value:

<table>
<thead>
<tr>
<th>777 mode value:</th>
<th>owner(rwx)</th>
<th>group(rwx)</th>
<th>other(rwx)</th>
</tr>
</thead>
</table>

2. If a UNIX file exists and the file format is set, cp copies the file as that format. Otherwise,
   • Load modules (PDS) are stored as UNIX executables and program objects (PDSE) are copied because they are the same as executables;
   • Data within data sets of undefined record format are copied as binary if the data is not a program object or load module;
   • Data found within data sets of fixed length or variable record length are copied as text. (IBM-1047 end-of-line delimiters are detected in the data)

Limitations: MVS to MVS: The limitations are as follows:
1. Options -A, -C, -F, and -S are ignored.
2. If target and source are in undefined record format (and neither is a sequential data set), cp will attempt to copy the data as a load module. If that fails, then cp will copy the data as binary.
3. If target and source are in undefined record format and either is a sequential data set, cp copies the data as binary.
4. If the source has a fixed or variable record length and the target is in undefined record format, cp copies the data as binary.
5. If the source is in undefined record format and the target has a fixed or variable record length, cp copies the data as binary.
6. If both source and target are in fixed or variable record length, cp copies the data as text.

Limitations: Copying executables into a PDS: The limitations are as follows:
1. A PDS cannot store load modules that incorporate program management features.
2. c89, by default, produces objects using the highest level of program management.
3. If you plan on copying a load module to a PDS, you can use a prelinker, which produces output compatible with linkage editor. Output generated by a linkage editor generated can be stored in a PDS. Table 10 on page 172 is a quick reference for determining the correct use of options with *cp*.

### Table 10: Options that are allowed for *cp*: File to File and File (multiple files) to directory

<table>
<thead>
<tr>
<th>Source/target</th>
<th>Options allowed</th>
<th>Options ignored</th>
<th>Options failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX file/UNIX file</td>
<td>Ffip</td>
<td>ABCMPSTUX</td>
<td></td>
</tr>
<tr>
<td>UNIX file/sequential data set</td>
<td>BFipt</td>
<td>ACfMpSU</td>
<td>X</td>
</tr>
<tr>
<td>UNIX file/PDS or PDSE member</td>
<td>BFITX</td>
<td>ACfMPpSU</td>
<td></td>
</tr>
<tr>
<td>Sequential data set/UNIX file</td>
<td>BFITU</td>
<td>ACMPpS</td>
<td>X</td>
</tr>
<tr>
<td>Sequential data set/ sequential data set</td>
<td>BFipt</td>
<td>ACfMpSU</td>
<td>X</td>
</tr>
<tr>
<td>Sequential data set/PDS or PDSE member</td>
<td>BFIT</td>
<td>ACfMPpSU</td>
<td>X</td>
</tr>
<tr>
<td>PDS or PDSE member/UNIX file</td>
<td>BFITU</td>
<td>ACMPpS</td>
<td></td>
</tr>
<tr>
<td>PDS or PDSE member/ sequential data set</td>
<td>BFipt</td>
<td>ACfMpSU</td>
<td>X</td>
</tr>
<tr>
<td>PDS or PDSE member/PDS or PDSE member</td>
<td>BFITX</td>
<td>ACfMPpSU</td>
<td></td>
</tr>
<tr>
<td>UNIX file/UNIX directory</td>
<td>ACfipS</td>
<td>BMPTUX</td>
<td></td>
</tr>
<tr>
<td>PDS or PDSE member/UNIX directory</td>
<td>BFIMSTUX</td>
<td>ACMPp</td>
<td></td>
</tr>
<tr>
<td>UNIX file/partitioned data set</td>
<td>ABCfIMSTX</td>
<td>fPpU</td>
<td></td>
</tr>
<tr>
<td>PDS or PDSE member/ partitioned data set</td>
<td>BFITX</td>
<td>ACfMPpSU</td>
<td></td>
</tr>
<tr>
<td>Partitioned data set/UNIX directory</td>
<td>ABCfIMSTUX</td>
<td>Pp</td>
<td></td>
</tr>
</tbody>
</table>

The tables that follow indicate the kind of copies that are allowed by using *cp*.

### Table 11: Copies that are allowed for *cp*: File to File

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX file, sequential data set, or partitioned data set member</td>
<td>UNIX file, sequential data set, or partitioned data set member</td>
<td>Yes</td>
</tr>
<tr>
<td>UNIX directory</td>
<td>UNIX directory</td>
<td>No (unless <em>cp</em> is used with -r or -r)</td>
</tr>
<tr>
<td>Partitioned data set</td>
<td>UNIX directory (dir) NOTE: results in each member of data set are moved to dir.</td>
<td>Yes</td>
</tr>
<tr>
<td>UNIX directory</td>
<td>Partitioned data set</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 11: Copies that are allowed for cp: File to File (continued)

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioned data set</td>
<td>Partitioned data set</td>
<td>No</td>
</tr>
<tr>
<td>UNIX file or partitioned data set member</td>
<td>UNIX directory (must exist) or partitioned data set</td>
<td>Yes</td>
</tr>
<tr>
<td>Partitioned data set member</td>
<td>Partitioned data set</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 12: Copies allowed for cp: File... (multiple files) to directory

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any combination of UNIX file or partitioned data set member</td>
<td>UNIX directory or Partitioned data set</td>
<td>Yes</td>
</tr>
<tr>
<td>Any combination of UNIX directory or sequential data set</td>
<td>Partitioned data set or UNIX directory</td>
<td>No</td>
</tr>
<tr>
<td>Partitioned data set</td>
<td>UNIX directory</td>
<td>Yes</td>
</tr>
<tr>
<td>Partitioned data set</td>
<td>Partitioned data set</td>
<td>No</td>
</tr>
</tbody>
</table>

Usage notes

You cannot copy a UNIX file to a PDS or PDS/E data set with the cp command if the PDS or PDS/E data set has already been opened by another user or job. For example, if the data set posix.test.pdse is being viewed by ISPF browse, the following cp command will fail:

```
cp test_file "//'posix.test.pdse(member)'"
```

UNIX to MVS:

1. To copy from UNIX to a partitioned data set, you must allocate the data set before doing the cp.
2. If an MVS data set does not exist, cp will allocate a new sequential data set of variable record format.
3. For text files, all <newline> characters are stripped during the copy. Each line in the file ending with a <newline> character is copied into a record of the MVS data set. If text file format is specified or already exists for the source file, that file format will be used for end-of-line delimiter instead of <newline>. Note that cp looks for IBM-1047 end-of-line delimiters in data.

You cannot copy a text file to an MVS data set that has an undefined record format.

- For an MVS data set in fixed record format, any line copied longer than the record size will cause cp to fail with a displayed error message and error code. If the line is shorter than the record size, the record is padded with blanks.
- For an MVS data set in variable record format: Any line copied longer than the largest record size will cause cp to fail with a displayed error message and error code. Record length is set to the length of the line.
4. For binary files, all copied data is preserved:

- For an MVS data set in fixed record format, data is cut into chunks of size equal to the record length. Each chunk is put into one record. The last record is padded with blanks.
- For an MVS data set in variable record format, data is cut into chunks of size equal to the largest record length. Each chunk is put into one record. The length of the last record is equal to length of the data left.
• For an MVS data set in undefined record format, data is cut into chunks of size equal to the block size. Each chunk is put into one record. The length of the last record is equal to the length of the data left.

5. For load modules, the partitioned data set specified must be in undefined record format. Otherwise the executable will not be copied.

6. If more than one file name is the same, the file is overwritten on each subsequent copy.

7. If a UNIX file name contains characters that are not allowed in an MVS data set, it will not be copied. If the UNIX file name has more than 8 characters, it can not be copied to an MVS data set member. (See the -ACMS options for converting file names)

8. You are not allowed to copy files into data sets with spanned records.

9. PDSE cannot have a mixture of program objects and data members. PDS allows mixing, but it is not recommended.

10. Special files such as external links and FIFO will not be copied to MVS data sets. However, you can copy character special files to MVS data sets.

11. If a file is a symbolic link, cp copies the resolved file, not the link itself.

12. UNIX file attributes are lost when copying to MVS. If you want to preserve file attributes, you should use the pax utility.

**MVS to UNIX:**

1. If the target UNIX file exists, the new data overwrites the existing data. The mode of the file is unchanged (except the S_ISUID and S_ISGID bits are turned off).

2. If the specified UNIX file does not exist, it will be created using 666 mode value if binary or text (this is subject to umask). If the data to be copied is a shell script or executable, the UNIX file will be created with 777 mode value (also subject to umask).

3. For an MVS data set in variable record format RECFM(VB) or undefined record format RECFM(U), trailing blanks are preserved when copying from MVS to UNIX. For an MVS data set in fixed record format, trailing blanks are not preserved when copying from MVS to UNIX.

4. When you copy MVS data sets to text files in the z/OS UNIX file system, a <newline> character is appended to the end of each record. If trailing blanks exist in the record, the <newline> character is appended after the trailing blanks. If the file format option is specified or the target file has the file format set, that file format is used as the end-of-line delimiter instead of <newline>.

5. When you copy MVS data sets to UNIX binary files, the <newline> character is not appended to the record.

6. You cannot use cp to copy data sets with spanned record lengths.

7. Due to an XL C/C++ runtime restriction, when copying a file from a file system to an MVS sequential data set with the same name and case, you must prefix the file in the file system with "./". For example:

   ```
   cp ./SMPL.DATA "//SMPL.DATA"
   ```

**Examples**

1. If _UNIX03 is unset or not 'YES, to specify -P params for a nonexisting sequential target:

   ```
   cp -P "RECFM=U,space=(500,100)" file "//turbo.gammalib"
   ```

   This **cp** command is equivalent to:

   ```
   cp -W "seqparms='RECFM=U,space=(500,100)'" file "//turbo.gammalib"
   ```

2. To copy file *f1* to a fully qualified sequential data set 'turbo.gammalib' and treat it as a binary:

   ```
   cp -F bin f1 "//turbo.gammalib"
   ```
3. To copy all members from a fully qualified PDS 'turbo.gammalib' to an existing UNIX directory `dir`:

   ```
   cp "//'turbo.gammalib'" dir
   ```

4. To drop .c suffixes before copying all files in UNIX directory `dir` to an existing PDS 'turbo.gammalib':

   ```
   cp -S d=.c dir/* "//'turbo.gammalib'"
   ```

Localization

The `cp` command uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_COLLATE`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_SYNTAX`
- `NLSPATH`

Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.

Environment variables

The `cp` command uses the following environment variable when copying records to or from MVS data sets:

```_EDC_ZERO_RECLEN```

If set to `Y` before calling `cp`, an empty record (zero-length) is treated as a single newline character and is not ignored. Also, a single newline character is written to the file as an empty record, and a single blank will be represented by "\n". If you do not set this environment variable when copying records, then the string "\n" is copied the same way as the string "\n": both are read and written as "\n", where "\n" indicates that z/OS XL C/C++ will write a record containing a single blank to the file (the default behavior of z/OS XL C/C++). All other blanks in the output are read back as blanks, and any empty records are ignored.

The `cp` command also uses the following environment variable:

```_UNIX03```

For more information about the effect of `_UNIX03` on this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Exit values

0

Successful completion

1

Failure due to any of the following:

- An argument had a trailing slash (/) but was not a directory.
- Inability to find a file.
- Inability to open an input file for reading.
- Inability to create or open an output file.
- A read error occurred on an input file.
- A write error occurred on an output file.
- The input and output files were the same file.
• An unrecoverable error when using -r or -R. Possible unrecoverable -r or -R errors include:
  – Inability to access a file.
  – Inability to change permissions on a target file.
  – Inability to read a directory.
  – Inability to create a directory.
  – A target that is not a directory.
  – Source and destination directories are the same.

2
Failure due to any of the following:
• An incorrect command-line option.
• Too few arguments on the command line.
• A target that should be a directory but isn’t.
• No space left on target device.
• Insufficient memory to hold the data to be copied.
• Inability to create a directory to hold a target file.

Messages
Possible error messages include:

cannot allocate target string
  cp has no space to hold the name of the target file. Try to release some memory to give cp more space.

name is a directory (not copied)
  You did not specify -r or -R, but one of the names you asked to copy was the name of a directory.

target name?
  You are attempting to copy a file with the -i option, but there is already a file with the target name. If you have specified -F, you can write over the existing file by typing y and pressing <Enter>. If you do not want to write over the existing file, type n and press <Enter>. If you did not specify -F and the file is read-only, you are not given the opportunity to overwrite it.

source name and target name are identical
  The source and the target are actually the same file (for example, because of links). In this case, cp does nothing.

unreadable directory name
  cp cannot read the specified directory; for example, because you do not have appropriate permission.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The -F and -m options are extensions of the POSIX standard.

Related information
cat, cpio, ln, mv, rm

**cpio - Copy in/out file archives**

**Format**

cpio [-o] [-aBcvyz] [-C blocksize] [-O file] [-V volpat]
The `cpio` utility is fully supported for compatibility with older UNIX systems. However, it is recommended that the `pax` utility be used instead because it might provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

**Description**

`cpio` reads and writes files called `cpio archives`. A cpio archive is a concatenation of files and directories preceded by a header giving the file name and other file system information. With `cpio`, you can create a new archive, extract contents of an existing archive, list archive contents, and copy files from one directory to another.

**Options**

Every call to `cpio` must specify one and only one of the following selector options:

- **-i**
  
  Reads an existing archive (created with the `-o` option) from the standard input (stdin). Unless you specify the `-t` option, `cpio` extracts all files matching one or more of the given `pattern` arguments from the archive. Patterns are the same as those used by file name generation (see `sh`). When you do not specify a `pattern` argument, the default pattern `*` is used; as a result, `cpio` extracts all files.

- **-o**
  
  Writes a new archive to the standard output (stdout), using the list of files read from stdin. Such a list might be produced by the `ls` or `find` commands. For example:
  
  ```bash
  ls . | cpio -o >arch
  ```
  
  uses `ls` to list the files of the working directory and then pipes this list as input to `cpio`. The resulting archive contains the contents of all the files, and is written to `arch`.

- **-p**
  
  Is shorthand for:
  
  ```bash
  cpio -o | (cd directory; cpio -i)
  ```
  
  where `cpio -i` is performed in the given directory. You can use this option to copy entire file trees.

Consult the syntax lines to determine which of the following additional options can be applied with a particular selector option:

- **-a**
  
  Resets the access time (of each file accessed for copying to the archive) to what it was before the copy took place.

- **-B**
  
  Uses buffers of 5120 bytes for input and output rather than the default 512-byte buffers.

- **-b**
  
  Causes 16-bit words to be swapped within each longword and bytes to be swapped within each 16-bit word of each extracted file. This facilitates the transfer of information between different processor architectures. This is equivalent to specifying both the `-s` and `-s` options.

- **-C blocksize**
  
  Sets the buffer size to a specified block size, rather than the default 512-byte buffers.

- **-c**
  
  Reads and writes header information in ASCII form. Normally, `cpio` writes the header information in a compact binary format. This option produces an archive more amenable to transfer through nonbinary streams (such as some data communication links) and is highly recommended for those moving data between different processors.
**-d**
Forces the creation of necessary intermediate directories when they do not exist.

**-f**
Inverts the sense of pattern matching. More precisely, cpio extracts a file from the archive if and only if it does not match any of the pattern arguments.

**-I file**
Causes input to be read from the specified file, rather than from stdin.

**-l**
Gives permission to create a link to a file rather than making a separate copy.

**-m**
Resets the modification time of an output file to the modification time of the source file. Normally, when cpio copies data into a file, it sets the modification time of the file to the time at which the file is written. This option has no effect on directories.

**-o file**
Causes output to be written to the specified file, rather than to stdout.

**-q**
Assumes that all created files are text. This means that any \r (carriage return) characters are stripped, and only the \n (newlines) are retained.

Do not use the -q option for converting text to a system-independent format, because that would require all files to be read twice.

**-r**
Lets you rename files as cpio works. When extracting, cpio displays the name of the component it is about to extract and gives you the chance to specify a name for the extracted file. If you enter . as the name, cpio processes the file or directory with no modification to the name. If you press Enter, cpio skips the file.

**-s**
For portability reasons, swaps pairs of 16-bit words within longwords (a 32-bit or 64-bit word) only when extracting files. This option does not affect the headers.

**-t**
Prevents files extraction, producing instead a table of file names contained in the archive. See the description of the -v option.

**-u**
Copies an archive file to a target file even if the target is newer than the archive. Normally, cpio does not copy the file.

**-v volpat**
Provides automatic multivolume support. cpio writes output to files, the names of which are formatted using volpat. The current volume number replaces any occurrence of # in volpat. When you invoke cpio with this option, it asks for the first number in the archive set, and waits for you to type the number and a carriage return before it proceeds with the operation. cpio issues the same sort of message when a write error or read error occurs on the archive; the reasoning is that this kind of error means that cpio has reached the end of the volume and should go on to a new one.

**-v**
Provides more verbose information than usual. cpio prints the names of files as it extracts them from or adds them to archives. When you specify both -v and -t, cpio prints a table of files in a format similar to that produced by the ls –l command.

**-y**
When used with -v, does not ask for a volume number to begin with, but does ask if it gets a read or write error.
-z
Performs Lempel-Ziv compression. Output is always a 16-bit compression. On input, any compression up to 16-bit is acceptable.

Usage notes
1. Use the pax command if you need to use multibyte patterns when searching for file names.
2. The POSIX 1003.1 standard defines formats for cpio archives that limit the UIDs and GIDs that can be stored to the maximum value of 262143. Larger values are not properly restored.
3. The byte and word swapping done by the -b, -s, and -r options is effective only for the file data written. With or without the -c option, header information is always written in a machine-invariant format.
4. The POSIX 1003.1 standard defines formats for cpio archives that limit the modification time to a maximum value of 7777777777 (octal) for ASCII headers and FFFFFFFF (hexadecimal) for binary headers. Modification times that are larger than the maximum value are not properly saved. Modification times beyond 03:14:07 UTC on January 19, 2038 but less than or equal to the maximum value, are not properly saved or restored on z/OS V2R2 and earlier systems. They also might not be properly saved or restored on non-z/OS systems. However, on systems later than z/OS V2R2, such values are properly saved. Due to these restrictions, the pax utility with the pax interchange format is the preferred method for processing time values beyond 03:14:07 UTC on January 19, 2038.

Localization
cpio uses the following localization environment variable:
- LANG
- LC_ALL
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH

Exit values
0
Successful completion.
1
Failure due to any of the following reasons:
- An incorrect option.
- Incorrect command-line arguments.
- Out of memory.
- Compression error.
- Failure on extraction.
- Failure on creation.

Portability
The -q, -v, -y, and -z options are specific to the z/OS shell.

Related information
compress, cp, dd, find, ls, mv, pax, tar, cpio, uncompress
The pax file format description in Appendix H, “File formats,” on page 969.
cron daemon - Run commands at specified dates and times

**Format**

cron

**Description**

cron is a clock daemon that runs commands at specified dates and times. You can specify regularly scheduled commands as described in crontab. You can also submit jobs that are to be run only once using the at command. cron runs commands with priorities and limits set by the queuedefs file. cron uses the value from queuedefs to lower the priority for non-UID=0 users only. The priority is unchanged for UID=0 users.

cron only examines crontab files and at command files when initializing or when a file changes using crontab or at. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

The setuid bit for cron should never be set; however, it must be started by a user with appropriate privileges to issue the setuid call for any UID. Because cron never exits, it should only be run once, normally during the system initialization process. cron automatically forks and runs itself in the background, in a new shell session. cron uses the pid file to prevent more than one cron running at the same time.

When matching the date and time expressions given in crontab entries, cron uses the time zone in effect when the system started the daemon. As a result, you should ensure that the TZ environment variable is set at this time. For information about setting the TZ environment variable, see Appendix I, “Format of the TZ environment variable,” on page 987. For at jobs, cron uses the value of TZ in effect when you submitted the job.

at, batch, and crontab submit jobs to cron; the data for those jobs can contain double-byte characters. When the jobs are executed, the data in the jobs are interpreted in the locale that cron is using. Because it is strongly recommended that cron be started in the POSIX locale, double-byte characters in the jobs may not be interpreted correctly. You can get around this by calling the setlocale() system call in the job itself.

The crontab, batch, and at job files store information about the MVS identity and the UNIX identity (the real UID) of the user who created the jobs. The cron daemon uses that information to set up the environment in which to run the jobs as follows:

- The MVS identity is set to the user's MVS identity.
- The UNIX real and effective UIDs are set to the user's real UID.

cron handles the following externally generated signals in a special way:

**SIGTERM**

Causes cron to exit. You can cause cron to exit with the following command:

```bash
kill -TERM pid
```

where `pid` is the cron's PID number. To find the cron's PID number, you can use:

```bash
ps -e | grep cron
```

**SIGUSR1**

Is sent by either at or crontab to indicate a new at job or an updated crontab entry. cron does not delete at jobs until they finish running. If the cron daemon is terminated while at jobs are running, cron runs them again when the daemon is restarted.

**SIGUSR2**

Writes internal cron queue information to the log file.
The following is an example of output to a `cron` log from `kill -USR2 5`. The output was written to the log on a test system when the `queuedefs` job limit of 5 was exceeded. The number of jobs that are running is 5 (the limit is 500):

```
Queue 'c' 5j2n15w:
queued 4, running 5, jobs 5
Next try for queued jobs 13 seconds
RUNNING: uid/gid: 0/512: pid 3354441: sleep 10000
RUNNING: uid/gid: 0/512: pid 38587972: echo start; sleep 10000
RUNNING: uid/gid: 0/512: pid 67108876: echo start; sleep 10000
RUNNING: uid/gid: 0/512: pid 67108879: echo start; sleep 10000
QUEUED: uid/gid: 0/512: echo Hello!
QUEUED: uid/gid: 0/512: echo Hello!
QUEUED: uid/gid: 0/512: echo Hello!
```

cron uses a number of files in the `/usr/lib/cron` directory to determine which users may and may not use the at and crontab commands.

- The file `at.allow` contains the list of users who have permission to use at.
- The file `at.deny` contains the list of users who do not have permission to use at.

If these files do not exist, only the superuser can use the at command. To allow all users access to at, there must be a null at.deny file and no at.allow file.

cron uses the files `cron.allow` and `cron.deny` in a similar manner.

- `cron.allow` contains the list of users who have permission to use crontab.
- `cron.deny` contains the list of users who do not have permission to use crontab.

If these files do not exist, only the superuser can use crontab. To allow all users access to crontab, there must be a null cron.deny file and no cron.allow file.

### Files

cron uses the following files which reside in a system-defined directory:

- `/etc/mailx.rc`
  Although cron does not use this file directly, cron may call mailx which uses this file for configuration settings. Specifically, for cron to deliver messages properly, this file should contain a valid setting for the mailx sendmail variable. This file is generally copied from `/samples/mailx.rc`.

- `/usr/spool/cron`
  The main cron directory.

- `/usr/spool/cron/atjobs`
  A directory containing at files.

- `/usr/spool/cron/crontabs`
  A directory containing crontab files.

- `/usr/spool/cron/log`
  A file that maintains a history of the commands being run. The systems administrator should truncate this file periodically.

- `/usr/spool/cron/pid`
  A file that cron uses to ensure that only one version of cron is currently running on the system. This file must be unique per system which is particularly important when you are setting up a sysplex. See the section in z/OS UNIX System Services Planning about customizing cron for a read-only file system.

- `/usr/lib/cron/at.allow`
  Contains a list of the users who can use the at command (one per line).

- `/usr/lib/cron/at.deny`
  Contains a list of the users who cannot use the at command (one per line).

- `/usr/lib/cron/cron.allow`
  Contains a list of the users who can use the crontab command (one per line).
crontab

/usr/lib/cron/cron.deny
Contains a list of the users who cannot use the crontab command (one per line).

/usr/lib/cron/queuedefs
The queue description file (see the description of queuedefs in “queuedefs - Queue description for at, batch, and cron” on page 976).

Related information
at, crontab, mailx

Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.

For more information about customizing cron, see the section on customizing the cron, uucp, and mail utilities for a read-only file system in z/OS UNIX System Services Planning. Also see the section about customizing the cron and uucp utilities in z/OS UNIX System Services Planning.

crontab - Schedule regular background jobs

Format

crontab [-e|-l|-r] [-u user] [file]

Description

crontab creates or changes your crontab entry. The crontab is a system facility that automatically runs a set of commands for you on a regular schedule. For example, you might set up your crontab entry so it runs a job every night at midnight, or once a week during low-use hours. This job could perform regular maintenance chores, for example, backing up files or getting rid of unnecessary work files.

To set up a crontab entry, use:

\[ \text{crontab file} \]

If you omit the file argument, crontab takes input from standard input. In this mode, you must provide your entire crontab file. This replaces any other existing crontab entries. If you issue crontab with no options, do not enter the end-of-file character or you will end up with an empty crontab file. Press INTERRUPT instead.

Input consists of six fields, which are separated by blanks. All blank lines and any input that contains a # as the first non-blank character are ignored. The first five give a date and time in the following form:

- A minute, expressed as a number from 0 through 59.
- An hour, expressed as a number from 0 through 23.
- A day of the month, expressed as a number from 1 through 31.
- A month of the year, expressed as a number from 1 through 12.
- A day of the week, expressed as a number from 0 through 6 (with 0 standing for Sunday).

Requirement: Always use a system default time zone. Your system administrator can tell you what it is. The cron daemon does not use the value of the TZ environment variable when crontab is invoked.

Any of these fields may contain an asterisk (\ast) standing for all possible values. For example, if you have an \ast as the day of the month, the job runs every day of the month. A field can also contain a set of numbers separated by commas, or a range of numbers, with the first number followed by a minus sign - followed by the second number. If you give specific days for both day of the month and day of the week, the two are ORed together. Here are some examples:

\[ \begin{align*}
0 & 0 * * * \quad \text{-- Midnight every day} \\
0 & 0 * * 1-5 \quad \text{-- Midnight every weekday}
\end{align*} \]
The sixth field of a crontab entry is a string that your shell executes at the specified time. When the shell executes this string, it sets the HOME, LOGNAME, PATH, and SHELL environment variables to default values for you.

If the string in your crontab entry contains percent characters %, the shell interprets them as newline characters, splitting your string in several logical lines. The first logical line (up to the first %) is interpreted as the command you want to execute; subsequent logical lines are used as standard input to the command. If any real (not logical) line in the file is blank or begins with #, the shell ignores the line (treats it as a comment).

To obtain the output of the command in your crontab entry, redirect the standard output and the standard error into a file. If you do not do this, the system mails you the output from the command.

at, batch, and crontab submit jobs to cron; the data for those jobs may contain double-byte characters. When the jobs are run, the data in the jobs is interpreted in the locale that cron is using. Because it is strongly recommended that cron be started in the POSIX locale, double-byte characters in the jobs may not be interpreted correctly. You can get around this by calling setlocale() in the job itself.

Options
-e
Let's you edit your crontab entry. crontab invokes an editor to edit the entry. if you have an EDITOR environment variable defined, crontab assumes that the variable's value is the name of the editor you want to use. if you do not have the EDITOR environment variable defined, crontab uses v1.

if you do not have a crontab entry, crontab sets up a blank entry for you. when you exit from the editor, crontab uses the edited entry as your new entry.

-l
Displays your current crontab entry on standard output.

-r
Removes (deletes) your current crontab entry.

-u user
Uses the crontab entry of user. the user specified has to be the same username that the crontab entry was created under in /usr/spool/cron/crontabs. this requires the appropriate privileges.

You can specify only one of the -e, -l, or -r options.

Environment variables

crontab uses the following environment variables:

EDITOR
Specifies the editor that the -e option invokes. The default editor is v1.

HOME
Is set to your user ID's home directory (not necessarily the current value of HOME when the commands in your crontab entry are run.

LOGNAME
Is set to your user ID when the commands in your crontab entry are run.

PATH
Is set to a system-wide default value when the commands in your crontab entry are run.

TZ
Is not used in time calculations. The cron daemon does, however, use this variable when cron is first started, typically when the system is started.
Localization

`crontab` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


Exit values

0
Successful completion.

1
Returned if the command fails for any reason. In this case, `crontab` does not change your crontab entry.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
The `-u` option is an extension to the POSIX standard.

Related information

at, batch, bg, cron

The queuedef file format description in Appendix H, “File formats,” on page 969.

csplit - Split text files

Format

```
csplit [-Aaks] [-f prefix] [-n number] file arg arg ...
```

Description

csplit takes a text file as input and breaks up its contents into pieces, based on criteria given by the `arg` value on the command line. For example, you can use `csplit` to break up a text file into chunks of ten lines each, then save each of those chunks in a separate file. See “Splitting criteria” on page 185 for more information. If you specify `-` as the `file` argument, `csplit` uses the standard input (stdin).

The files created by `csplit` normally have names of the form

```
xxnumber
```

where `number` is a 2-digit decimal number that begins at zero and increments by one for each new file that `csplit` creates.

`csplit` also displays the size, in bytes, of each file that it creates.

Options

-A
Uses uppercase letters in place of numbers in the number portion of created file names. This generates names of the form `xxA`, `xxB`, and so on.
-a
Uses lowercase letters in place of numbers in the number portion of created file names. This generates names of the form xxaa, xxab, and so on.

-f prefix
Specifies a prefix to use in place of the default xx when naming files. If it causes a file name longer than NAME_MAX bytes, an error occurs and csplit exits without creating any files.

-k
Leaves all created files intact. Normally, when an error occurs, csplit removes files that it has created.

-n number
Specifies the number of digits in the number portion of created file names.

-s
Suppresses the display of file sizes.

Splitting criteria

csplit processes the args on the command line sequentially. The first argument breaks off the first chunk of the file, the second argument breaks off the next chunk (beginning at the first line remaining in the file), and so on. Thus each chunk of the file begins with the first line remaining in the file and goes to the line given by the next arg.

arg values can take any of the following forms:

/regexp/
Takes the chunk as all the lines from the current line up to but not including the next line that contains a string matching the regular expression regexp. After csplit obtains the chunk and writes it to an output file, it sets the current line to the line that matched regexp.

/regexp/offset
Is the same as the previous criterion, except that the chunk goes up to but not including the line that is a given offset from the first line containing a string that matches regexp. The offset can be a positive or negative integer. After csplit has obtained the chunk and written it to an output file, it sets the current line to the line that matched regexp.

Note: This current line is the first one that was not part of the chunk just written out.

%regexp%
Is the same as /regexp/, except that csplit does not write the chunk to an output file. It simply skips over the chunk.

%regexp%offset
Is the same as /regexp/offset, except csplit does not write the chunk to an output file.

linenumber
Obtains a chunk beginning at the current line and going up to but not including the linenumberth line. After csplit writes the chunk to an output file, it sets the current line to linenumber.

{number}
Repeats the previous criterion number times. If it follows a regular expression criterion, it repeats the regular expression process number more times. If it follows a linenumber criterion, csplit splits the file every linenumber lines, number times, beginning at the current line. For example,

csplit file 10 {10}

obtains a chunk from line 1 to line 9, then every 10 lines after that, up to line 109.

Errors occur if any criterion tries to grab lines beyond the end of the file, if a regular expression does not match any line between the current line and the end of the file, or if an offset refers to a position before the current line or past the end of the file.
Localization

csplit uses the following localization variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


Exit values

0
Successful completion.

1
Failure due to any of the following reasons:
- csplit could not open the input or output files.
- A write error on the output file.

2
Failure due to any of the following reasons:
- Unknown command-line option.
- The prefix name was missing after -f.
- The number of digits was missing after -n.
- The input file was not specified.
- No arg values were specified.
- The command ran out of memory.
- An arg was incorrect.
- The command found end-of-file before it was expected.
- A regular expression in an arg was badly formed.
- A line offset/number in an arg was badly formed.
- A \{number\} repetition count was misplaced or badly formed.
- Too many file names were generated when using -n.
- Generated file names would be too long.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.

The -A and -a options are extensions to the POSIX standard.

Related information

awk, sed

For more information about regexp, see Appendix C, “Regular expressions (regexp),” on page 945.
ctags - Create tag files for ex, more, and vi

Format

ctags [-aBFwx] [-f tagfile] sourcefile ...

Description

ctags creates a file named tags in the current directory. It summarizes the locations of various objects in the C source files named on the command line. All files with a .c or .h suffix are treated as C source files.

For C source code, ctags summarizes function, macro and typedef definitions. See “tags - Format of the tags file” on page 977 for a description of the format of the tags file.

The tags file is used by ex, more, and vi to support the tag command. The tag command can be used to edit the file containing a name in the tags file.

For ex and vi, the command is:

: tag name

For more, the command is:

:t name

After these commands are run, the tags file is searched for name. If it is found, the file associated in the tags file with that name is loaded and the line containing the name is made the current line.

Options

-a
  Appends output to the existing tags file rather than overwriting the file.

-B
  Produces a tags file that searches backward from the current position to find the pattern matching the tag.

-F
  Searches for tag patterns in the forward direction. This is the default.

-f
  Generates a file named tagfile rather than the default tags file.

-w
  Suppresses warning messages.

-x
  Produces a report on the standard output. The report gives the definition name, the line number of where it appears in the file, the name of the file in which it appears, and the text of that line. ctags arranges this output in columns and sorts it in order by tag name according to the current locale’s collation sequence. This option does not produce a tags file.

Localization

ctags uses the following localization environment variables:

• LANG
• LC_ALL
• LC_COLLECT
• LC_CTYPE
Files

ctags uses the following file:

tags
  Output tags file.

Usage notes

1. It can be difficult to recognize a function definition in C source code. Because ctags does not know which C preprocessor symbols are defined, there may be some misplaced function definition information if sections of code within #if...#endif are not complete blocks.

2. ctags invokes the sort internally.

3. ctags makes special provision for the main() function, which may occur in several C source files. The tags file contains an entry for the first main() routine found. For all occurrences of main(), including the first, the tags file contains an entry for Mname, where name is the name of the input source file, with the .c suffix and any leading path name components removed. For example, a tags file created for a C source code file named foo.c would contain an entry for Mfoo, which represents the main() routine in foo.c.

4. ctags uses sort to sort the file by tag name, according to the POSIX locale's collation sequence.

Exit values

0
  Successful completion

1
  Failure due to any of the following reasons:
  • Unknown command-line option.
  • Cannot create the output file.
  • Cannot open the output file.
  • One of the input files was unrecognized.

Portability

POSIX.2, X/Open Portability Guide, 4.2BSD and later.
This utility only understands characters from the POSIX locale.
The -B, -F, and -w options are extensions to the POSIX and XPG standards.

Related information

more, sort, vi
See the tags file format description in “tags - Format of the tags file” on page 977.

cu - Call up another system (stub only)

Format

cu [-dehot] [-l device_name] [-s speed] [system_name | phone_num]
cu -n [-dehot] [-l device_name] [-s speed]
Note: The **cu** utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, this utility should be avoided for applications intended to be portable to other UNIX-branded systems.

**Description**

**cu** connects to remote systems specified in the UUCP configuration file. You can use it for simple terminal connections, or to do simple file transfer with no error checking.

**cu** is recognized, but its functions are disabled. Traditionally, it is used for simple terminal connections to remote systems specified in the UUCP configuration file. **cu** requires a direct connection (such as with a modem) to the remote system, but this is not supported by z/OS.

---

**cut** - Cut out selected fields from each line of a file

**Format**

```
cut -b list [-Bn] [-W option[option] ... [file...]]
cut -c list [-B] [-W option[option] ... [file...]]
cut -f list [-d char] [-Bs] [-W option[option] ... [file...]]
```

**Description**

**cut** reads input from files, each specified with the **file** argument, and selectively copies sections of the input lines to the standard output. If you do not specify any **file**, or if you specify a file named **-**, **cut** reads from standard input.

**Options**

- **-B**
  
  Disables the automatic conversion of tagged files. This option is ignored if the **filecodeset** or **pgmcodeset** options (-W option) are specified.

- **-b list**
  
  Invokes byte position mode. After this comes a list of the byte positions you want to display. This list might contain multiple byte positions, separated by commas (,) or blanks or ranges of positions separated by dashes (-). Since the list must be a single argument, shell quoting is necessary if you use blanks. You can combine these to allow selection of any byte positions of the input.

  When you use the -b option with double-byte characters, you must also specify the -n option if you want to ensure that entire characters are displayed. If you do not specify the -n option, **cut** assumes that the low byte of a range is the first byte of a character and that the high byte of a range is the last byte of a double-byte character, possibility resulting in the misinterpretation of the characters represented by those byte positions.

- **-c list**
  
  Invokes character-position mode. After this comes a list of character positions to retain in the output. This list can contain many character positions, separated by commas (,) or blanks or ranges of positions separated by a dash (-). Since the list must be a single argument, shell quoting is necessary if you use blanks. You can combine these to allow selection of any character positions of the input.

- **-d char**
  
  Specifies **char** as the character that separates fields in the input data; by default, this is the horizontal tab.

- **-f list**
  
  Invokes field delimiter mode. After this comes a list of the fields you want to display. You specify ranges of fields and multiple field numbers in the same way you specify ranges of character positions and multiple character positions in -c mode.
-n
Does not split characters. If the low byte in a selected range is not the first byte of a character, cut extends the range downward to include the entire character; if the high byte in a selected range is not the last byte of a character, cut limits the range to include only the last entire character before the high byte selected. If -n is selected, cut does not list ranges that do not encompass an entire character, and these ranges do not cause an error.

-s
Does not display lines that do not contain a field separator character. Normally, cut displays lines that do not contain a field separator character in their entirety.

-W option[,...]
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

filecodeset=\[codeset\]
Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

pgmcodeset=\[codeset\]
Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

Examples
1. To print a list that contains the dates that the files were created and the file names of files in the working directory:

   ls -al | cut -c 42-48,54-66

2. To display the first field of each line of a file containing ASCII characters to the standard output (stdout), assuming that
• The text file is untagged and you do not want to tag it or enable automatic conversion, and
• You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only
  text file)

then issue:

```
cut -f 1 -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 myAsciiFile
```

3. To display the second byte of each line of a file containing EBCDIC characters to the standard output
   (stdout), assuming that automatic conversion has been enabled but the text file is incorrectly tagged
   as UTF-8:

```
cut -b 2 -B myMisTaggedFile
```

**Localization**

`cut` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Environment variables**

`cut` uses the following environment variable:

`_TEXT_CONV`

Contains text conversion information for the command. The text conversion information is not used
when either the -B option or the `filecodeset` or `pgmcodeset` option (-W option) is specified. For
more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX
shell commands,” on page 993.

**Exit values**

0

Successful completion

1

Failure due to any of the following reasons:

• Cannot open the input file.
• Out of memory.
• The code set is not valid.
• Could not turn off automatic conversion.
• Could not perform requested text conversion.

2

Failure due to any of the following reasons:

• An incorrect command-line argument.
• You did not specify any of the -b, -c, or -f options.
• You omitted the `list` argument.
• Badly formed `list` argument.
Portability
POSIX.2, X/Open Portability Guide, UNIX System V.
The -B and -W options are extensions of the POSIX standard.

Related information
paste, uname

cxx - Compile C and C++ source code, link-edit and create an executable file

See c89/xlc or man xlc.

When working in the shell, to view man page information about cxx, type: man c89 or man xlc.

date - Display the date and time

Format
date [-cu ] [+format]

Description
date displays the operating system's concept of the current date and time. The following example shows the default format of the date:

Wed Feb 26 14:01:43 EST 1986

Options
date accepts the following options:

-c
Displays the date and displays the time according to Greenwich Mean Time (Coordinated Universal Time) using CUT as the time zone name.

-u
Displays the date and displays the time according to Greenwich Mean Time (Coordinated Universal Time) using GMT as the time zone name.

If the argument to date begins with a + character, date uses format to display the date. date writes all characters in format, except for the % and the character that immediately follows it, directly to the standard output. After date exhausts the format string, it outputs a newline character. The % character introduces a special format field similar to the printf() function in the C library. date supports the following field descriptors:

%A
The full weekday name (for example, Sunday).

%a
The three-letter abbreviation for the weekday (for example, Sun).

%B
The full month name (for example, February).

%b
The three-letter abbreviation for the month name (for example, Feb).

%C
The first two digits of the year (00 to 99).
The local representation of the date and time (see `%D` and `%T`).

 `%D`  
The date in the form `mm/dd/yy`.  

 `%d`  
The two-digit day of the month as a number (01 to 31).  

 `%e`  
The day of the month in a two-character, right-aligned, blank-filled field.  

 `%H`  
The two-digit hour (00 to 23).  

 `%h`  
The three-letter abbreviation for the month (for example, Jun).  

 `%I`  
The hour in the 12-hour clock representation (01 to 12).  

 `%j`  
The numeric day of the year (001 to 366).  

 `%M`  
The minute (00 to 59).  

 `%m`  
The month number (01 to 12).  

 `%n`  
The newline character.  

 `%p`  
The local equivalent of AM. or PM.  

 `%r`  
The time in AM-PM notation (11:53:29 a.m.).  

 `%S`  
The seconds (00 to 61). There is an allowance for two leap seconds.  

 `%T`  
The time (14:53:29).  

 `%t`  
A tab character.  

 `%U`  
The week number in the year, with Sunday being the first day of the week (00 to 53).  

 `%W`  
The week number in the year, with Monday being the first day of the week (00 to 53).  

 `%w`  
The weekday number, with Sunday being 0.  

 `%X`  
The local time representation (see `%T`).  

 `%x`  
The local date representation (see `%D`).  

 `%Y`  
The year.  

 `%y`  
The two-digit year.  

 `%Z`  
The time zone name (for example, EDT).  

 `%%`  
A percent-sign character.
The **date** command also supports the following modified field descriptors to indicate a different format as specified by the locale indicated by LC_TIME. If the current locale does not support a modified descriptor, **date** uses the unmodified field descriptor value.

- **%EC**
  The name of the base year (period) in the alternative representation of the current locale.

- **%Ec**
  The current locale's alternative date and time representation.

- **%Ex**
  The current locale's alternative date representation.

- **%EY**
  The full alternative year representation.

- **%Ey**
  The offset from %EC (year only) in the alternative representation of the current locale.

- **%Od**
  The day of the month using the alternative numeric symbols of the current locale.

- **%Oe**
  The day of the month using the alternative numeric symbols of the current locale.

- **%OH**
  The hour (24-hour clock) using the alternative numeric symbols of the current locale.

- **%OI**
  The hour (12-hour clock) using the alternative numeric symbols of the current locale.

- **%OM**
  The minutes using the current locale's alternative numeric symbols.

- **%Om**
  The month using the current locale's alternative numeric symbols.

- **%OS**
  The seconds using the current locale's alternative numeric symbols.

- **%OU**
  The week number of the year (0–53) (with Sunday as the first day of the week) using the alternative numeric symbols of the current locale.

- **%OW**
  The week number of the year (0–53) (with Monday as the first day of the week) using the alternative numeric symbols of the current locale.

- **%Ow**
  The weekday as a number using the current locale's alternative numeric symbols (Sunday=0).

- **%Oy**
  The year (offset from %C) using the current locale's alternative numeric symbols.

**Examples**

The command:

```
date '+%a %b %e %T %Z %Y'
```

produces the date in the default format as shown at the start of this command description.

**Environment variables**

**date** uses the following environment variables:

- **TZ**
  Gives the time zone for **date** to use when displaying the time. This option is ignored if you specify either the –c or the –u option.
For information about setting the local time zone with the TZ environment variable, see Appendix I, “Format of the TZ environment variable,” on page 987.

Localization

date uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_TIME
• NLSPATH


Exit values

0  Successful completion.

>0  Failure due to any of the following reasons:

• An incorrect command-line option.
• Too many arguments on the command line.
• A bad date conversion.
• A formatted date that was too long.
• You do not have permission to set the date.

Messages

Possible error messages include:

Bad format character x
A character that follows % in the format string was not in the list of field descriptors.

No permission to set date
The system denied you the right to set the date.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.
The -c option is an extension of the POSIX standard.

Related information

touch

Appendix I, “Format of the TZ environment variable,” on page 987 also explains how to set the local time zone with the TZ environment variable.

dbgld - Create a module map for debugging
Format

dbgld [option] ...file

Description

The compiler creates a .dbg file for each compilation unit if the DEBUG compiler option is specified. The path names of all of the .dbg files are then stored in the module, which is an executable file or a DLL. The dbgld command opens all of the .dbg files associated with the module and stores all of the functions, global variables, external types, and source files in a single module map file with a .mdbg extension. In addition, the contents of all of the .dbg files are packaged together into this same .mdbg file. The dbgld command only needs to be executed once after binding.

Debuggers that support demand load can use the .mdbg file for faster access to debug information. For more information on using the module map to improve debugger performance, see z/OS Common Debug Architecture User's Guide.

If the original source files are not available at debugging time, you can add the source file contents to the .mdbg file before the source files are relocated. For example, the source files are moved into a different directory or the compilation and debugging are performed on different machines. When invoking the dbgld command, you can specify the -c option because the source file contents cannot be captured into the .mdbg file by the dbgld command by default. A debugger that supports captured source can then retrieve the source file contents from the .mdbg file.

Options

option

-c

Adds captured source file to the module map, which consists of all files that contain executable lines of code.

-cf

Adds captured source file to the module map, which consists of all files regardless if they contain executable lines of code.

-v

Writes the version information to standard error.

file

Is the module name, which can be:

- The absolute path name of a z/OS UNIX file.
- The relative path name of a z/OS UNIX file.
- A fully qualified MVS data set (PDS member).

The output of the dbgld command is a file with the name of the module followed by a .mdbg extension. The file will always be written in the current directory. For example, if the module name is /mypath/mymodule, a file called mymodule.mdbg will be created in the current directory. If the file already exists, it will be overwritten.

Restrictions

The following restrictions apply to the use of dbgld:

- The source files must be compiled with the DEBUG compiler option.
- The name of a valid module must be passed into the dbgld utility. The module must be bound with the EDIT=YES binder option, which is the default. An error message will be generated if EDIT=NO.
• The .dbg files associated with the module must exist in the directories where they were during compilation. Otherwise, they will not be added to the module map and no debug information will be available to the compilation units via the module map during debugging. An error message will be generated for each .dbg file that is not found.

• Because the dbgld command always creates the .mdbg file in the current directory, the command must be run from a directory that has write permission.

• Source files compiled with NOGOFF and NOLONGNAME are not processed by the utility. If the entire module is made up of these compilation units, an error message will be generated to indicate that no debug information was found. Compile your application with LONGNAME or GOFF to mitigate this restriction.

Example
The following example shows how to compile hello1.c and hello2.c and create a module map in a file called hello.mdbg.

```
xlc -g hello1.c hello2.c -o hello
dbgld hello
```

The following example shows how to compile hello1.c and hello2.c and create a module map in a file called hello.mdbg which contains captured source.

```
xlc -g hello1.c hello2.c -o hello
dbgld -c hello
```

The following example shows how to display the version information for the dbgld utility and the Common Debug Architecture run times when creating the module map.

```
dbgld -v hello
```

The output is:

```
CDA0000I Utility(dbgld ) Level(level name)
CDA0000I Library(elf ) Level(level name)
CDA0000I Library(dwarf ) Level(level name)
CDA0000I Library(ddpi ) Level(level name)
```

If the `-g` option is missing during compilation, hello.mdbg is not generated and a warning message is printed, as shown in the following example:

```
xlc hello1.c hello2.c -o hello
dbgld hello
```

The output is a warning message stating that no debug information was found in hello.

Exit values
The exit values for dbgld are:

0  Successful completion
4  Warning
8  Error
12  Severe error
dbx - Use the debugger

Format

$dbx$ [options] [executable-file [program-arguments ...]]
$dbx$ [options] [attach-type] process-id
$dbx$ [options] -C core-file

Description

$dbx$ is a source-level debugger for z/OS UNIX System Services. It provides an environment to debug and run C and C++ programs, as well as performing machine level debug. You can carry out operations such as the following:

- Examine object files.
- Run a program in a controlled environment.
- Set breakpoints at selected statements or run the program one line at a time.
- Debug using symbolic variables and display them in their correct format.
- View an MVS dump.
- Attach to a running program, and perform debugging operations.

The $executable-file$ argument is an load module produced by a compiler. To perform source-level debugging, you need to compile your executable with symbolic information. This is accomplished by specifying the -g or -Wc,debug compiler flags on the compiler command line.

**Note:** If the object file is not compiled with the -g or -Wc, debug option on the c89/cc/c++ command, or if the user compiles with optimization, the capabilities of the $dbx$ command will be reduced.

The $core-file$ argument is an MVS dump that exists as a file in the z/OS UNIX file system or in an MVS data set.

$dbx$ allows the end user to customize its behavior via two files that are processed during initialization. These are .dbxsetup. Each file can contain a list of $dbx$ subcommands that will be run before the $dbx$ prompt is displayed. During startup, $dbx$ will first search for these files in the current working directory and then in the user’s $HOME$ directory. If a file is found, it is parsed and the search for that specific file terminates. Use a text editor to create a .dbxsetup or .dbxinit file.

Any subcommand in the .dbxsetup file are executed before the debug target program is loaded. This allows the user to tailor $dbx$’s operational behavior during this phase of the $dbx$ startup process. Any subcommand in the .dbxinit file are executed just before the $dbx$ prompt is displayed.

You can use the man command to view descriptions of $dbx$ subcommands. To do this, you must prefix all subcommands with $dbx$. For example, to view a description of the $dbx$ alias subcommand, you would enter the following:

```
man dbxalias
```

$dbx$ supports SVC dumps and SYSMDUMP dumps. It does not process SYSUDUMP dumps or CEEDUMP dumps. (The dump data set must be FB with a record length of 4160.)

Attach-types

**-a ProcessID**

Attaches the debug program to a running process. The debug program becomes active as soon as the process wakes up. To attach the debug program, you need authority to use the kill command on this process.

**-A ProcessID**

Reattaches the debug program to a running process that is already being debugged by $dbx$. Use this option to reattach a child process that was created when a debugged parent process did a fork while
multiprocess debugging mode was active. To reattach to the debug program, you need authority to use the kill command on this process.

Options
-b
  Suppresses processing of .dbxsetup and .dbxinit files (bare startup).
-c script
  Runs dbx subcommands from a specified script file before reading from standard input.
-C dump-filename
  Puts dbx in dump reading (core file) processing mode.
-d
  Deprecated. This option is ignored and remains only for compatibility purposes.
-f
  Deprecated. This option is ignored and remains only for compatibility purposes.
-F
  Starts debug target in a different address space than the one dbx currently resides in; or starts dbx in its own address space when attaching to a running target program.
-h
  Prints the dbx command syntax.
-I directory
  Appends the given directory to the list of directories searched for source and debug files. The default list contains the working directory and the directory containing the object files. The search path can also be set with the use subcommand.
-m dbxmode
  Instructs dbx to start in a specific mode:
  • Specifying -m4 forces dbx to run in 31-bit mode, even on a machine capable of running it in 64-bit mode.
  • Specifying -m8 forces dbx to run in 64-bit mode. If this is impossible, dbx will terminate.
-p ipaddress | name[:port]
  Tells dbx to open a socket and connect to the ipAddress:port or machineName:port which is assumed to be a GUI that supports remote debugging. For example: dbx -p 9.123.456.78:8001 mypgm. In this mode, the command-line prompt is not displayed and the user cannot enter dbx subcommands through the command-line interface. In addition, the following dbx subcommands are not supported when entered via a GUI debug console (command line) interface: detach, edit, multproc, object, quit, rerun, run, sh.
  If :port is not specified, port 8001 is used as the default.
-q
  Suppresses some of the dbx startup messages. Only the dbx version information is printed before the prompt is displayed. All other messages are suppressed. This option does not affect the verbosity of normal dbx operation.
-r
  Runs the object file immediately. If it ends successfully, the dbx debug program is exited. Otherwise, the debug program is entered and the reason for termination is reported.
  Unless -r is specified, the dbx command prompts the user and waits for a command. However, you can specify program arguments on the .dbxsetup files that are used by dbx and .dbxinit even when -r is not used. For example:
  
  ```
  dbx myprog arg1 arg2 arg3
  ```
-u
  Deprecated. This option is ignored and remains only for compatibility purposes.
Expression handling

Specify expressions in dbx with a subset of C and Pascal syntax. A prefix * (asterisk) or a postfix ^ (circumflex) denotes indirection. Specify portions of an array by separating the lower and upper bounds with .. (two periods).

Use [ ] (square brackets) or () (parentheses) to enclose array subscripts. Use the field reference operator . (period) with pointers and records. The field reference operator . (period) makes the C operator → unnecessary (although it is supported).

When displaying variables and expressions, the dbx command resolves names first using the static scope of the current function. The dynamic scope is used if the name is not defined in the first scope. If static and dynamic searches do not yield a result, an arbitrary symbol is chosen and the system prints the message (using Module.Variable). The Module.Variable construction is the name of an identifier qualified with a block name. Override the name resolution procedure by qualifying an identifier with a block name. Source files are treated as modules named by the file name without the language suffix (such as the .c suffix on a C language program).

The dbx command debug program checks types of expressions. Override types of expressions by using TypeName (Expression). When there is no corresponding named type, use the &TypeName special construct to represent a pointer to the named type. Represent a pointer to enum, struct, or union tag with the $$TagName construct.

A condition can be any dbx expression that evaluates to a true or false value. This pertains to four dbx subcommands: stop, stopi, trace, and tracei.

The following operators are valid in expressions:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp (exponentiation) Algebraic</td>
<td>+ (addition), - (subtraction), * (multiplication), / (floating point division), div (integral division), mod (modulo division), exp (exponentiation)</td>
</tr>
<tr>
<td>Bitwise</td>
<td>? (unary minus), ? (bitwise or), &amp; (bitwise and), ? (bitwise xor), ~ (one’s complement), &lt;&lt; (bitwise left shift), &gt;&gt; (bitwise right shift), bitand (bitwise and), xor (bitwise xor)</td>
</tr>
<tr>
<td>Logical</td>
<td>?? (logical or), &amp;&amp; (logical and), ! (logical not), or, and, not</td>
</tr>
<tr>
<td>Comparison</td>
<td>&lt; (less than), &gt; (greater than), &lt;= (less than or equal to), &gt;= (greater than or equal to), &lt;&gt; (not equal to), != (not equal to), = (equal to), == (equal to)</td>
</tr>
<tr>
<td>Other</td>
<td>sizeof</td>
</tr>
</tbody>
</table>

Files

a.out

The object file a.out is the default name of an executable file produced by the compiler. If no executable is specified on the dbx command line, then dbx will look for an a.out file to debug.
.dbxinit
Contains initial commands.

.dbxsetup
Contains initial commands.

.dbxhistory
Contains a listing of recently used dbx commands. Used by the history dbx subcommand.

Examples
1. This example shows you how to attach dbx to a running process. To do this, it is useful to insert a sleep(30) call into the program to be debugged. This will give you enough time to issue the ps -Aef shell command to find the process ID of the program. Once you have the process ID, issue the shell command: dbx -a process-id.

Before starting the server, set the following four environment variables:

- `export _BPX_PTRACE_ATTACH=yes` (tells the kernel to always load DLLs and executables into read/write storage)
- `export _CEE_RUNOPTS="test(all)"` (tells Language Environment to load the CEEEVDBG debugger exit and send dbx debug events)
- `export _CEE_DEBUG_FILENAME31=/bin/dbx31vdbg` (the path to the Language Environment debugger exit for 31-bit programs)
- `export _CEE_DEBUG_FILENAME64=/bin/dbx64vdbg` (the path to the Language Environment debugger exit for 64-bit programs)

Use dbx commands to set breakpoints, step through program statements, print variables, work with threads, examine storage, and actions as needed.

2. Example of creating the .dbxinit file in your home directory:

   ```bash
   alias nsf "use /sandbox3/UNIX_notes/CGOOD/notes/nsf/"
   alias asc "set $asciichars ; set $asciistrings"
   alias ebc "unset $asciichars ; unset $asciistrings"
   set $repeat
   set $history_unique
   set $hold_next
   set $showbases
   ```

3. Sample dbx commands issued after starting server and seeing the sleeping for 30 seconds message for server process ID 50331876:

   ```bash
   /sandbox3/UNIX_notes/CGOOD/notes/os> dbx -a 50331876
   FDBX0278: Waiting to attach to process 50331876 ...
   FDBX0609: dbx for MVS.
   FDBX0699: Compiled: Sep 28 2001 10:22:24 GMT as BFP
   FDBX0400: OS level: 12.00 03, LE level: 4.1.2 with CWIs.
   FDBX0100: Type 'help' for help.
   FDBX0099: reading symbolic information ...
   FDBX0900: reading symbols for
   /sandbox3/UNIX_notes/CGOOD/usr/lpp/lotus/notes/latest/os390/server ...
   FDBX0637: XPLink module found
   FDBX0900: reading symbols for
   /sandbox3/UNIX_notes/CGOOD/usr/lpp/lotus/notes/latest/os390/libnotes ...

   attached in sleep at 0xebcd024 ($t1)
   sleep() at 0xebcd024
   unnamed block $b64, line 873 in "meminit.c"
   MemoryInit1(), line 873 in "meminit.c"
   OSInitExt() at 0x1000cdc4
   ServerMain() at 0xf80ac38
   main() at 0xebe0ac30
   main() at 0x80a12e
   () at 0xebeb2f4a
   () at 0x6f8e976
   0xebcd024 (+0xff3c3024) 47060003  nop X'3'($r6,)
   (dbx) stop at "meminit.c":875
   [1] stop at "meminit.c":875
   (dbx) c
   [1] stopped in unnamed block $b64 at line 875 in file "meminit.c" ($t1)
   (dbx) where
   unnamed block $b64, line 875 in "meminit.c"
   ```
MemoryInit1(), line 875 in "meminit.c"
OSInitExt() at 0x1000cdc4
ServerMain() at 0xf80ac38
main() at 0xf80a12e
.() at 0xebe2f4a
() at 0x6f8e976
(dbx) list 872,875
872    __printf_a("sleeping for 30 seconds time to dbx\n");
873    sleep(30);
874    __printf_a("Done sleeping \n");
875    if (loc_num_of_segs > MAX_NUM_OF_SEGM)
(dbx) print loc_num_of_segs
4
(dbx) &loc_num_of_segs/8x
0fc55af8:  0000 0004 0000 01b0 0400 0000 0f9d 57f8
(dbx) 0xfc55af8/8x
0fc55af8:  0000 0004 0000 01b0 0400 0000 0f9d 57f8
(dbx) n
Stopped in unnamed block $b64 at line 971 in file "meminit.c" ($t1)
971   DoAgain2:
(dbx) n
Stopped in unnamed block $b64 at line 972 in file "meminit.c" ($t1)
972      hMMShMemId = shmget(ShmemAccessKey,
FirstSegSize, IPC_CREAT|IPC_EXCL|loc_shm390flags|perms);

Usage notes

The following dbx subcommands can have their output redirected to a file.

- alias
- args
- condition
- dump
- examine
- list
- listfiles
- listfuncs
- listi
- map
- mutex
- onload
- readwritelock
- rerun
- registers
- run
- sh
- status
- thread
- whatis
- where
- whereis
- which

The following dbx subcommands will return an error message in GUI mode (-p).

- detach
- edit
- multproc
Related information

c89/cc/c++

? subcommand for dbx: Search backward for a pattern

Format

? [RegularExpression]

Description
The ? subcommand searches backward in the current source file for the pattern specified by the RegularExpression argument. Entering the ? subcommand with no arguments causes dbx to search backward for the previous regular expression.

Usage notes
The ? subcommand can be run only while the dbx debug program is running.

Examples
1. To search backward in the current source file for the letter z, enter:
   
   ?z

2. To repeat the previous search, enter:
   
   ?

Related information
The / (search) subcommand.

/ subcommand for dbx: Search forward for a pattern

Format

/ [RegularExpression]

Description
The / subcommand searches forward in the current source file for the pattern specified by the RegularExpression argument. Entering the / subcommand with no arguments causes dbx to search forward for the previous regular expression.

Usage notes
The / subcommand can be run only while the dbx debug program is running.
**Examples**

1. To search forward in the current source file for the number 12, enter:

   / 12

2. To repeat the previous search, enter:

   /

**Related information**

The ? (search) subcommand.

**alias subcommand for dbx: Display and assign subcommand aliases**

**Format**

alias [name] [string]

**Description**

The alias subcommand creates aliases for dbx subcommands. The *name* argument is the alias being created. The *string* argument is a series of dbx subcommands that, after the execution of this subcommand, can be referred to by *name*. If the alias subcommand is used without aliases, it displays all current aliases.

**Usage notes**

The alias subcommand can be run only while the dbx debug program is running.

**Examples**

1. To set tracef1 to be an alias for trace in f1, enter:

   alias tracef1 "trace in f1"

2. To define a stopf alias with *file* and *line* arguments to allow shorthand for setting a breakpoint within a file, enter:

   alias stopf(file, line) "stop at \"file\":line"

**args subcommand for dbx: Display program arguments**

**Format**

args

**Description**

The args subcommand displays the argument count and a list of arguments that are passed to the user's program when dbx starts debugging the program.

**Usage notes**

The args subcommand can be run only while the dbx debug program is running.
Examples
To display the current arguments, enter:

```
args
```

Related information
The `rerun` and `run` subcommands.

**asid subcommand for dbx: Display a list of address spaces**

**Format**

```
asid [set]
```

**Description**

The `asid` subcommand displays a list of all address spaces in the dump file. The current address space is defaulted to the one that contains the current process; `dbx` uses it as the context for the `pid` subcommand. You can use the `set` option to switch the current `dbx` address space.

**Usage notes**

The `asid` subcommand can be run only in dump debugging mode.

**Example**

1. To display all address spaces, issue the following command:

```
asid
```

2. To switch the `dbx` focus address space, issue the following command:

```
asis set
```

Then type an index in the address space identifier (ASID) list.

**Related information**

The `list` and `pid` subcommands.

**assign subcommand for dbx: Assign a value to a variable**

**Format**

```
assign [variable=expression]
```

**Description**

The `assign` subcommand assigns the value specified by the `expression` argument to the variable specified by the `variable` argument.

**Usage notes**

1. The `assign` subcommand can be run only while the `dbx` debug program is running.
2. Functions cannot be specified with the `expression` argument.
Examples

1. To assign the value 5 to a variable \(x\), enter:
   
   ```
   assign x = 5
   ```

2. To assign the value of a variable \(y\) to a variable \(x\), enter:
   
   ```
   assign x = y
   ```

3. To assign a value to a storage location, enter:
   
   ```
   assign 0x02e0f7f0 = 0xff
   ```

4. To assign a value to a register, enter:
   
   ```
   assign $r7 = 123
   ```

5. To change the `exit_status` of a specific thread, enter:
   
   ```
   assign $t1.exit_status=&$void(0x2d95840);
   ```

case subcommand for dbx: Change how dbx interprets symbols

Format

case [default | mixed | lower | upper]

Description

The `case` subcommand changes how the `dbx` debug program interprets symbols. Use `case` if a symbol needs to be interpreted in a way not consistent with the default behavior.

Entering `case` with no parameters displays the current case mode.

Options

default

   Varies with the current language.

mixed

   Causes symbols to be interpreted as they actually appear.

lower

   Causes symbols to be interpreted as lowercase.

upper

   Causes symbols to be interpreted as uppercase.

Usage notes

The `case` subcommand can be run only while the `dbx` debug program is running.

Examples

1. To instruct `dbx` to interpret symbols as they actually appear, enter:
   
   ```
   case mixed
   ```

2. To instruct `dbx` to interpret symbols as uppercase, enter:
   
   ```
   case upper
   ```
catch subcommand for dbx: Start trapping a signal

Format
catch [signalnumber | signalname]

Description
The catch subcommand starts the trapping of a specified signal before that signal is sent to the application program. This subcommand is useful when the application program being debugged handles such signals as interrupts. The signal to be trapped can be specified by number or by name using either the signalnumber or the signalname argument, respectively. Signal names are case-insensitive, and the SIG prefix is optional. All signals are caught by default except the SIGDUMP, SIGHUP, SIGCHLD, SIGALRM, and SIGKILL signals. If no arguments are specified, the current list of signals to be caught is displayed.

Usage notes
The catch subcommand can be run only while the dbx debug program is running.

Examples
1. To display a current list of signals to be caught by dbx, enter:
   ```
catch
   ```
2. To trap signal SIGALRM, enter:
   ```
catch SIGALRM
   ```
or:
   ```
catch ALRM
   ```
or:
   ```
catch 14
   ```

Related information
The ignore subcommand.

clear subcommand for dbx: Remove all stops at a specified source line

Format
clear sourceline

Description
The clear subcommand removes all stops at a specified source line. The sourceline argument can be specified in two formats:

- As an integer
- As a file name string followed by a : (colon) and an integer

Usage notes
The clear subcommand can be run only while the dbx debug program is running.
Examples
To remove breakpoints set at line 19, enter:

```
clear 19
```

Related information
The cleari and delete subcommands.

cleari subcommand for dbx: Remove all breakpoints at an address

Format
```
cleari address
```

Description
The cleari subcommand clears all the breakpoints at the address specified by the `address` argument.

Usage notes
The cleari subcommand can be run only while the dbx debug program is running.

Examples
1. To remove a breakpoint set at address 0X100001B4, enter:
```
cleari 0x100001b4
```
2. To remove a breakpoint set at the address of the `main()` procedure, enter:
```
cleari &main
```

Related information
The clear and delete subcommands.

condition subcommand for dbx: Display a list of active condition variables

Format
```
condition &lbrk;number &ellip;&rbrk;
condition wait
condition nowait
```

Description
The condition subcommand displays a list of active condition variables for the application program. All active condition variables are listed unless you use the `number` parameter to specify the condition variables you want listed. You can also select condition variables with or without waiters by using the wait or nowait options.

In order to capture the condition variables, dbx must be debugging your program before the condition variable is created. You must have coded your application in one of the following ways:

- Add the following line at the top of the C program:
```
#pragma runopts(TEST(ALL))
```

Or:
• Code an assembler program, CEEUOPT, to invoke the CEEXOPT macro, which specifies TEST(ALL). For examples of how to code this program, see z/OS Language Environment Programming Guide.

**Usage notes**
The condition subcommand can be run only while the dbx debug program is running.

**Examples**
1. To display all condition variables, enter:
   ```
   condition
   ```
2. To display condition variables number 1 and number 4, enter:
   ```
   condition 1 4
   ```
3. To display all condition variables with waiters, enter:
   ```
   condition wait
   ```
4. To display all condition variables without waiters, enter:
   ```
   condition nowait
   ```

**cont subcommand for dbx: Continue program execution**

**Format**
```
cont [signalnumber | signalname]
```

**Description**
The `cont` subcommand continues the execution of the program from the current stopping point until either the program finishes, another breakpoint is reached, a signal is received that is trapped by the dbx command, or an event occurs (such as a fork, an exec, or a program abend).

If a signal is specified, either by the number specified in the `signalnumber` argument or by the name specified in the `signalname` argument, the program continues as if that signal had been received by the focus thread.

If a signal is not specified, the dbx debug program variable `$sigblock` is set, and a signal caused the debugged program to stop, then the program resumes execution. If a signal is not specified, the dbx debug program variable `$sigblock` is not set, and a signal caused the debugged program to stop, then typing in the `cont` command with no signal causes the program to continue as if it had received the original signal.

Signal names are not case-sensitive, and the SIG prefix is optional. If no signal is specified, the program continues as if it had not been stopped.

**Usage notes**
The `cont` subcommand can be run only while the dbx debug program is running.

**Examples**
1. To continue program execution from the current stopping point, enter:
   ```
   cont
   ```
To continue program execution as though it had received the signal **SIGQUIT**, enter:

```
cont SIGQUIT
```

**Related information**
The step, next, goto, and skip subcommands.

### delete subcommand for dbx: Remove traces and stops

**Format**
```
delete [all | number...]
```

**Description**
The delete subcommand removes traces and stops from the program. You can specify the traces and stops to be removed through the `number` arguments, or you can remove all traces and stops by using the `all` option. To display the numbers associated by the dbx debug program with a trace or stop, use the status subcommand.

**Options**
```
all
```
Removes all traces and stops.

**Usage notes**
The delete subcommand can be run only while the dbx debug program is running.

**Examples**
1. To remove all traces and stops from the program, enter:

```
delete all
```

2. To remove traces and stops for event number 4, enter:

```
delete 4
```

**Related information**
The status, clear, and cleari subcommands.

### detach subcommand for dbx: Continue program execution without dbx control

**Format**
```
detach [signalnumber | signalname]
```

**Description**
The detach subcommand continues the execution of a program from the current stopping point without control of dbx.

If a signal is specified, either by the number specified in the `signalnumber` argument or by the name specified in the `signalname` argument, the program continues without dbx control as if that signal had been received by the focus thread. If the signal is not specified, the program continues with no signal and without dbx control.
Signal names are not case-sensitive, and the SIG prefix is optional. If no signal is specified, the program continues without dbx control as if it had not been stopped.

Usage notes

• The detach subcommand can be run only while the dbx debug program is running.
• The detach subcommand is not supported in GUI (socket) mode (-p).

Examples

1. To continue program execution from the current stopping point without dbx in control, enter:

   detach

2. To continue program execution without dbx control as though it had received the signal SIGQUIT, enter:

   detach SIGQUIT

Related information

The cont subcommand.

display memory subcommand for dbx: Display the contents of memory

Format

address, address/[mode]
address/[count][mode]
[b|Bd|Bf|Bq|c|d|D|f|g|h|i|I|ld|lo|lx|o|O|q|s|S|o|W|X][>file]

Description

The display memory subcommand displays the contents of memory. The display starts at the first address, and terminates at either the second address or until count items are printed. If the address is ":", the address following the one most recently printed is used. The mode specifies how memory is to be printed; if it is omitted the previous mode specified is used. The initial mode is "X".

• The range of memory displayed is controlled by specifying:

  --
  Two address arguments, in which case all lines between those two addresses are displayed (address/address), or

  --
  One address argument, where the display starts, and count, which determines the number of lines displayed from address/address/count).

• Used in place of the first address argument, this displays from the point where you left off (see example "3" on page 213).

• Symbolic addresses are specified by preceding the name with an & (ampersand).

• Registers are denoted by "$rN", "$frN" or "$drN", where N is the number of the register.

• Addresses may be expressions made up of other addresses and the operators +, -, and * indirection.
Any expression enclosed in parentheses is interpreted as an address.

The format in which the memory is displayed is controlled by the *mode* argument. The default for the *mode* argument is the current mode. The initial value of *mode* is *X*. The possible modes include:

- **b**
  - Print a byte in octal format.
- **Bf**
  - Print single precision real number in binary floating point.
- **Bg**
  - Print a double precision real number in binary floating point.
- **Bq**
  - Print a long double precision real number in binary floating point.
- **c**
  - Print a byte as a character.
- **C**
  - Print a *wchar_t* character.
- **d**
  - Print a short word in decimal.
- **D**
  - Print a long word in decimal.
- **Df**
  - Print single precision real number in decimal floating point.
- **Dg**
  - Print a double precision real number in decimal floating point.
- **Dq**
  - Print a long double precision real number in decimal floating point.
- **f**
  - Print a single precision real number in hexadecimal floating point.
- **g**
  - Print a double precision real number in hexadecimal floating point.
- **h**
  - Print a byte in hexadecimal format.
- **ha**
  - Print a byte in hexadecimal format and ASCII
- **he**
  - Print a byte in hexadecimal format and EBCDIC
- **i**
  - Print the machine instruction.
- **I**
  - Print a *wint_t* character.
- **ld**
  - Print a long long in signed decimal.
- **lo**
  - Print a long long in octal format.
- **lu**
  - Print a long long in unsigned decimal
- **lx**
  - Print a long long in hexadecimal format
Print a short word in octal format

Print a long word in octal format

Print a long double precision real number in hexadecimal floating point

Print a string (terminated by a null byte).

Print a wchar_t string.

Print a wint_t string.

Print a short word in hexadecimal format.

Print a long word in hexadecimal format.

Options

>file
  Redirects output to the specified file.

Usage notes

The display memory subcommand can be run only while the dbx debug program is running.

Examples

1. To display one long word of memory content in hexadecimal format starting at the address 0X3FFFE460, enter:

   0x3fffe460 / x

2. To display 2 bytes of memory content as characters starting at the address of variable y, enter:

   &y/2c

3. To display from the point where you left off, when using . (period) in place of one of the addresses, enter:

   0x100 / 2              which displays 2 words starting at x'100'

followed by:

   . / 3                  which displays 3 words starting at x'108'

Related information

See also: cleari, gotoi, registers, stepi, nexti, tracei, and stopi commands.

down subcommand for dbx: Move the current function down the stack

Format

down [count]
**dbx: down**

**Description**
The down subcommand moves the current function down the stack *count* number of levels. The current function is used for resolving names. The default for the *count* argument is 1.

**Usage notes**
The down subcommand can be run only while the dbx debug program is running.

**Examples**
1. To move one level down the stack, enter:
   ```
   down
   ```
2. To move three levels down the stack, enter:
   ```
   down 3
   ```

**Related information**
The up and where subcommands.

**dump subcommand for dbx: Display the names and values of variables in a procedure**

**Format**
```
dump [procedure] [file]
```

**Description**
The dump subcommand displays the names and values of all variables in the specified procedure. If the *procedure* argument is . (dot), all active variables are displayed. If the *procedure* argument is not specified, the current procedure is used. If the *file* option is used, the output is redirected to the specified file.

**Options**
- `>file`
  
  dump output to the specified file.

**Usage notes**
The dump subcommand can be run only while the dbx debug program is running. dump redirects output to the specified file.

**Examples**
1. To display names and values of variables in the current procedure, enter:
   ```
   dump
   ```
2. To display names and values of variables in the **add_count** procedure, enter:
   ```
   dump add_count
   ```
3. To redirect names and values of variables in the current procedure to the **var.list** file, enter:
   ```
   dump > var.list
   ```
edit subcommand for dbx: Invoke an editor

**Format**

edit [procedure | file]

**Description**

The edit subcommand invokes an editor on the specified file. The file can be specified through the *file* argument or through the *procedure* argument (in which case the editor is invoked on the file containing that procedure). If no file is specified, the editor is invoked on the current source file. The default editor is the ed editor. Override the default by resetting the EDITOR environment variable to the name of the desired editor.

**Usage notes**

- The edit subcommand can be run only while the dbx debug program is running.
- The edit subcommand is not supported in GUI (socket) mode (-p).

**Examples**

1. To invoke an editor on the current source file, enter:

   `edit`

2. To invoke an editor on the *main.c* file, enter:

   `edit main.c`

3. To invoke an editor on the file containing the *do_count* procedure, enter:

   `edit do_count`

**Related information**

The ed editor.

The list subcommand for the dbx command.

file subcommand for dbx: Change the current source file

**Format**

file [file]

**Description**

The file subcommand changes the current source file to the file specified by the *file* argument; it does not write to that file. If the *file* argument is not specified, the file subcommand displays the name of the current source file.

**Usage notes**

The file subcommand can be run only while the dbx debug program is running.

**Examples**

1. To change the current source file to the main.c file, enter:

   `file main.c`
2. To display the name of the current source file, enter:

```
file
```

**func subcommand for dbx: Change the current function**

**Format**

```
func [procedure]
```

**Description**

The `func` subcommand changes the current function to the procedure or function specified by the `procedure` argument. If the `procedure` argument is not specified, the default current function is displayed. Changing the current function implicitly changes the current source file to the file containing the new function; the current scope used for name resolution is also changed.

**Usage notes**

The `func` subcommand can be run only while the dbx debug program is running.

**Examples**

1. To change the current function to the do_count procedure, enter:

```
func do_count
```

2. To display the name of the current function, enter:

```
func
```

**goto subcommand for dbx: Run a specified source line**

**Format**

```
goto sourceline
```

**Description**

The `goto` subcommand causes the specified source line to be run next. Typically, the source line must be in the same function as the current source line. To override this restriction, use the `set` subcommand with the `$unsafegoto` variable.

**Usage notes**

- The `goto` subcommand can be run only while the dbx debug program is running.
- While dbx allows the changing of the next instruction to be run, unpredictable results might occur in the program being debugged. Whether the unpredictable results occur depends on where the program is currently stopped, the current state of the program, and where the program is to resume running.

**Examples**

To change the next line to be executed to line 6, enter:

```
goto 6
```

**Related information**

The `cont`, `gotoi`, and `set` subcommands.
gotoi subcommand for dbx: Change the program counter address

Format

gotoi address

Description

The gotoi subcommand changes the program counter address to the address specified by the address argument.

Usage notes

• The gotoi subcommand can be run only while the dbx debug program is running.
• While dbx allows the changing of the next instruction to be run, unpredictable results might occur in the program being debugged. Whether the unpredictable results occur depends on where the program is currently stopped, the current state of the program, and where the program is to resume running.

Examples

To change the program counter address to address 0X100002B4, enter:

```
gotoi 0x100002b4
```

Related information

The goto subcommand.

help subcommand for dbx: Display a subcommand synopsis

Format

help [subcommand] [topic]

Description

The help subcommand displays a synopsis of common dbx subcommands.

Usage notes

The help subcommand can be run only while the dbx debug program is running.

Examples

To obtain a synopsis of common dbx subcommands, enter one of the following commands:

```
help
help subcommand
help topic
```

The help subcommand with no arguments lists available dbx subcommands and topics.

The help subcommand, where subcommand is one of the dbx subcommands, displays a synopsis and brief description of the subcommand. help topic (where topic is execution, files, threads, scope, usage, or variables) displays a synopsis and brief description of the topic.
history subcommand for dbx: Display commands in a history list

**Format**

```
history
```

**Usage notes**

The `history` subcommand displays the commands in the history list. As each command is entered, it is appended to the history list. A mechanism for history substitution is provided through the exclamation (!) operator. The allowable forms are `!!` for a previous command, `!n` for the `n`th command, and `!string` for the previous command that starts with `string`. The number of commands retained and displayed is controlled by the dbx internal variable `$historywindow`.

ignore subcommand for dbx: Stop trapping a signal

**Format**

```
ignore [signalnumber | signalname]
```

**Description**

The `ignore` subcommand stops the trapping of a specified signal before that signal is sent to the program. This subcommand is useful when the program being debugged handles such signals as interrupts.

The signal to be trapped can be specified by:

• Number, with the `signalnumber` argument
• Name, with the `signalname` argument

Signal names are not case-sensitive. The `SIG` prefix is optional.

If neither the `signalnumber` nor the `signalname` argument is specified, all signals except the SIGDUMP, SIGHUP, SIGCHLD, SIGALRM, and SIGKILL signals are ignored by default. The dbx debug program cannot catch SIGKILL or SIGDUMP. If no arguments are specified, the list of currently ignored signals is displayed.

**Usage notes**

The `ignore` subcommand can be run only while the dbx debug program is running.

**Examples**

To cause dbx to ignore alarm clock timeout signals sent to the program, enter:

```
ignore sigalrm
```

or:

```
ignore alrm
```

or:

```
ignore 14
```

**Related information**

The `catch` subcommand.
**list subcommand for dbx: Display lines of the current source file**

**Format**

```
list [procedure | SourcelineExpression] [SourcelineExpression]
```

**Description**

The list subcommand displays a specified number of lines in the source file. The number of lines displayed are specified in one of two ways:

- By specifying a procedure using the `procedure` argument. In this case, the list subcommand displays lines before the first executable line of source in the specified procedure and until the list window is filled.
- By specifying a starting and ending source line number using the `SourcelineExpression` argument. Use the current filename or source filename if specified.

The `SourcelineExpression` argument should consist of a valid line number followed by an optional + or - and an integer. In addition, a `SourcelineExpression` of `$` can be used to denote the current line number, and a `SourcelineExpression` of `@` can be used to denote the next line number to be listed.

All lines from the first line number specified to the second line number specified, inclusive, are then displayed, provided these lines fit in the list window.

If the second source line is omitted, ten lines are printed, beginning with the line number specified in the `SourcelineExpression` argument.

If the `list` subcommand is used without arguments, the default number of lines are printed, beginning with the current source line. The default is 10.

To change the number of lines to list by default, set the special debug program variable, `$listwindow`, to the number of lines you want. Initially, `$listwindow` is set to 10.

**Usage notes**

The `list` subcommand can be run only while the dbx debug program is running.

**Examples**

1. To list the lines 1 through 10 in the current file, enter:

   ```
   list 1,10
   ```

2. To list 10, or `$listwindow`, lines around the first executable line in the `main` procedure, enter:

   ```
   list main
   ```

3. To list 11 lines around the current line, enter:

   ```
   list $-5,$+5
   ```

**Related information**

The `edit`, `listi`, `move`, and `set` subcommands.

**listfiles subcommand for dbx: Display the list of source files**

**Format**

```
listfiles [loadmap-index]
```
**dbx: listfuncs**

**Description**
The `listfiles` subcommand displays the list of files associated with each module in the load map.
If the `listfiles` subcommand is used without arguments, the files for every module in the load map will be listed.

**Usage notes**
The `listfiles` subcommand can be run only while the `dbx` debug program is running.

**Examples**
1. To list all files in all modules, enter:
   ```
   listfiles
   ```
2. To list the files only for module with index 0 in the loadmap, enter:
   ```
   listfiles 0
   ```

**Related information**
The `listfuncs` and `map` subcommands.

**listfuncs subcommand for dbx: Display the list of functions**

**Format**
```
listfuncs [filename]
```

**Description**
The `listfuncs` subcommand displays a list of functions associated with each file in the program.
If the `listfuncs` subcommand is used without arguments, the function for every file in the program will be listed.

**Usage notes**
The `listfuncs` subcommand can be run only while the `dbx` debug program is running.

**Examples**
1. To list all functions in all files, enter:
   ```
   listfuncs
   ```
2. To list the function only for file `mypgm.c`, enter:
   ```
   listfuncs mypgm.c
   ```

**Related information**
The `func` subcommand.

**listi subcommand for dbx: List instructions from the program**

**Format**
```
listi [procedure | at | sourceline | address | [,address]
```
Description
The listi subcommand displays a specified set of instructions from the source file. The instructions displayed are specified by:

- Providing the *procedure* argument, in which case the listi subcommand lists instructions from the beginning of *procedure* until the list window is filled.
- Using the *at sourceline* option, in which case the listi subcommand displays instructions beginning at the specified source line and continuing until the list window is filled.
- Specifying a beginning and ending address using the *address* arguments, in which case all instructions between the two addresses, inclusive, are displayed.

If the listi subcommand is used without options or arguments, the next $listwindow instructions are displayed. To change the current size of the list window, use the set $listwindow=value subcommand.

Options

**at sourceline**
- Specifies a starting source line for the listing.

Usage notes
The listi subcommand can be run only while the dbx debug program is running.

Examples
1. To list the next 10, or $listwindow, instructions, enter:
   
   listi

2. To list the machine instructions beginning at source line 10, enter:
   
   listi at 10

3. To list the instructions between addresses 0X10000400 and 0X10000420, enter:
   
   listi 0x10000400, 0x10000420

Related information
The list and set subcommands.

map subcommand for dbx: Display load characteristics

**Format**
map [>

**Description**
The map subcommand displays characteristics for each loaded portion of the program. This information includes the name, text origin, text length, text end, text subpool, data origin, data length, data subpool, and file descriptor for each loaded module. The data origin, data length, data subpool, and file descriptor do not contain meaningful information.

**Options**

>
- Redirects output to the specified file.
dbx: move

Usage notes
The map subcommand can be run only while the dbx debug program is running.

Examples
To examine the characteristics of the loaded portions of the application, enter:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
</tr>
</tbody>
</table>

move subcommand for dbx: Display or change the next line to be shown with the list command

Format
move
move sourceline
move function

Description
The move subcommand changes the next line to be displayed to the line specified by the sourceline argument. This subcommand changes the value of the @ variable. The sourceline argument can either be a line number in the current file, or a function name. Omitting the sourceline argument will display the current line number.

Usage notes
The move subcommand can be run only while the dbx debug program is running.

Examples
To change the next line to be listed to line 12, enter:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>move 12</td>
</tr>
</tbody>
</table>

To change the next line to be listed to be the function main, enter:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>move main</td>
</tr>
</tbody>
</table>

To display the current line number, enter:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>move</td>
</tr>
</tbody>
</table>

Related information
The list subcommand.

multproc subcommand for dbx: Enable or disable multiprocess debugging

Format
multproc
multproc [on]
multproc [off]
multproc [parent]
multproc [child]
Description
The `multproc` subcommand alters the way `dbx` behaves when the process that is being debugged issues a `fork()` runtime call. By default, multiprocess debugging is disabled when `dbx` is started. If no options are specified, the `multproc` subcommand returns the current status of multiprocess debugging.

Options
- **on**
  - `dbx` will notify the user that a fork has occurred, provide the PID of the new child process, and follow the parent process.
- **off**
  - `dbx` will ignore any forks that occur.
- **parent**
  - `dbx` will notify the user that a fork has occurred and follow the parent process.
- **child**
  - `dbx` will notify the user that a fork has occurred and follow the child process.

Usage notes
- The `multproc` subcommand can be run only while the `dbx` debug program is running.
- The `multproc` subcommand is not supported in GUI (socket) mode (`-p`).

Examples
1. To check the current status of multiprocess debugging, enter:
   ```
   multproc
   ```
2. To have `dbx` notify the user that the process being debugged has forked, enter:
   ```
   multproc on
   ```
3. To have `dbx` ignore all forks by the process being debugged, enter:
   ```
   multproc off
   ```

Related information
The `fork()` function.

mutex subcommand for `dbx`: Display a list of active mutex objects

Format
- `mutex [number ...]`
  - `mutex lock`
  - `mutex unlock`
  - `mutex wait`
  - `mutex nowait`

Description
The `mutex` subcommand displays a list of active mutex objects for the application program. All active mutex objects are listed unless you use the `number` parameter to specify the mutex objects you want listed. You can also select only locked or unlocked mutexes, or mutexes with or without waiters, by using the `lock`, `unlock`, `wait`, or `nowait` options.
In order to capture the mutex variables, dbx must be debugging your program before the mutex variable is created. You must have coded your application in one of the following ways:

- Add the following line at the top of the C program:
  ```c
  #pragma runopts(TEST(ALL))
  ```

Or:

- Code an assembler program, CEEUOPT, to invoke the CEEXOPT macro, which specifies TEST(ALL). For examples of how to code this program, see z/OS Language Environment Programming Guide.

Usage notes

The mutex subcommand can be run only while the dbx debug program is running.

Examples

1. To display all mutex objects, enter:
   ```
   mutex
   ```

2. To display mutex objects number 1 and number 4, enter:
   ```
   mutex 1 4
   ```

3. To display all locked mutex objects, enter:
   ```
   mutex lock
   ```

4. To display all unlocked mutex objects, enter:
   ```
   mutex unlock
   ```

5. To display all mutex objects with waiters, enter:
   ```
   mutex wait
   ```

6. To display all mutex objects without waiters, enter:
   ```
   mutex nowait
   ```

next subcommand for dbx: Run the program up to the next source line

Format

```next [number]```

Description

The next subcommand runs the application program up to the next source line. The number argument specifies the number of times the next subcommand runs. If the number argument is not specified, next runs once only.

Usage notes

1. The next subcommand can be run only while the dbx debug program is running.
2. If the $hold_next variable is defined, dbx holds all threads except the focus thread during a next subcommand. Then dbx unholds the threads after the next subcommand finishes.
Examples
1. To continue execution up to the next source line, enter:
   
   ```plaintext
   next
   ```

2. To continue execution up to the third source line following the current source line, enter:
   
   ```plaintext
   next 3
   ```

Related information
The `cont`, `goto`, `nexti`, and `stepi` subcommands.

nexti subcommand for dbx: Run the program up to the next machine instruction

Format

```plaintext
nexti [number]
```  

Description
The `nexti` subcommand runs the application program up to the next instruction. The `number` argument specifies the number of times the `nexti` subcommand is to be run. If the `number` argument is not specified, `nexti` runs only once.

Usage notes
1. The `nexti` subcommand can be run only while the dbx debug program is running.
2. If the `$hold_next` variable is defined, dbx holds all threads except the focus thread during a `nexti` subcommand. Then dbx unholds the threads after the `nexti` subcommand finishes.

Examples
1. To continue execution up to the next machine instruction, enter:
   
   ```plaintext
   nexti
   ```

2. To continue execution up to the third machine instruction following the current machine instruction, enter:
   
   ```plaintext
   nexti 3
   ```

Related information
The `gotoi`, `next`, and `stepi` subcommands.

object subcommand for dbx: Load an object file

Format

```plaintext
object filename
```  

Description
The object subcommand loads the specified object file for execution, without the overhead of reloading dbx.
Usage notes

- The object subcommand can be run only while the dbx debug program is running.
- The object subcommand is not supported in GUI (socket) mode (-p).

Examples

To complete debugging of the current program, and to start debugging a new program without reloading dbx, enter:

```plaintext
object myprog
```

onload subcommand for dbx: Evaluate stop/trace after DLL load

Format

```plaintext
onload delete [all|number...]
onload list
onload stop at sourceline
onload stop in procedure
onload trace at sourceline
onload trace in procedure
```

Description

The onload subcommand defers building of stop or trace events until the `procedure` or `sourceline` is defined in the program dbx is debugging. dbx will evaluate the onload list after a DLL is loaded and generate stop/trace events if the `procedure` or `sourceline` is now known to dbx after symbolics for the DLL are processed. If the `procedure` or `sourceline` is already known to dbx, then a normal stop or trace event will be generated and no event will be added to the onload list.

Usage notes

The onload subcommand can be run only while the dbx debug program is running.

Examples

To defer the building of a stop or trace event, enter:

```plaintext
onload stop in myfunc
onload stop in myclass::memfunc
onload stop at "mypgm.c":52
onload trace in myfunc
onload trace in myclass::memfunc
onload trace in "mypgm.c":52
```

Related information

The stop and trace subcommands.

plugin subcommand for dbx: Pass the specified command to the plug-in parameter

Format

```plaintext
plugin[name[command]]
```
Description
The plugin subcommand passes the command specified by the command parameter to the plug-in specified by the name parameter. If no parameters are specified, the names of all available plug-ins are displayed.

Usage notes
The plugin subcommand can be run only while the dbx debug program is running.

Examples
1. To list all available plug-ins, enter:
   
   plugin

2. To invoke the subcommand help of a plug-in named sample, enter:
   
   plugin sample help

3. To invoke the subcommand interpret 0x12345678 of a plug-in named xyz, enter:
   
   plugin xyz interpret 0x12345678

See the pluginload subcommand and also the topic on Developing for dbx Plug-in Framework in z/OS UNIX System Services Programming Tools.

pluginload subcommand for dbx: Load the specified plug-in

Format
pluginload file

Description
The pluginload subcommand loads the plug-in specified by the file parameter. The file parameter should specify a path to the plug-in.

Usage notes
The pluginload subcommand can be run only while the dbx debug program is running.

Examples
1. To load the plug-in named sample located at /home/user/dbx_plugins/libdbx_sample.dll, enter:
   
   pluginload /home/user/dbx_plugins/libdbx_sample.dll

See the pluginload subcommand and also the topic on developing for the dbx plug-in framework in z/OS UNIX System Services Programming Tools.

pluginunload subcommand for dbx: Unload the specified plug-in

Format
pluginunload name

Description
The pluginunload subcommand unloads the plug-in specified by the name parameter. The name parameter should specify a name of plug-in that is currently loaded.


**Usage notes**

The pluginunload subcommand can be run only while the dbx program is running.

**Examples**

1. To unload the plug-in named sample, enter:

   ```
   pluginunload sample
   ```

   See the plugin subcommand and the pluginload subcommand. Also see the topic on developing for the dbx plug-in framework in *z/OS UNIX System Services Programming Tools*.

---

**print subcommand for dbx: Print the value of an expression**

**Format**

`print [expression,…] [(parameters)]`

**Description**

The `print` subcommand prints the value of a list of expressions, specified by the `expression` arguments.

**Usage notes**

The `print` subcommand can be run only while the dbx debug program is running.

**Examples**

1. To display the value of `x` and the value of `y` shifted left 2 bits, enter:

   ```
   print x, y << 2
   ```

2. To display a specific condition variable, enter:

   ```
   print $c1
   ```

3. To display the number of waiters for a specific mutex object, enter:

   ```
   print $m1.num_wait
   ```

4. To display the exit value of a specific thread, enter:

   ```
   print $t1.exit_status
   ```

**Related information**

The `assign` and `set` subcommands.

---

**prompt subcommand for dbx: Change the dbx command prompt**

**Format**

`prompt ["string"]`

**Description**

The `prompt` subcommand changes the dbx command prompt to the string specified by the `string` argument.
Usage notes
The prompt subcommand can be run only while the dbx debug program is running.

Examples
To change the prompt to dbx>, enter:

```
prompt "dbx>"
```

pid subcommand for dbx: Display a list of processes

Format
pid [set]

Description
The pid subcommand displays a list of all processes in the address space. The current process is defaulted to the one that is causing the problem. dbx then uses it as the context for other dbx subcommands such as thread. You can use the set option to switch the current dbx process.

Usage notes
The pid subcommand can be run only in dump debugging mode.

Example
1. To display all processes in the address space, issue:

   `pid`

2. To switch the dbx focus address space, issue:

   `pid set`

   Then type an index in the pid list.

Related information
The asid and list subcommands.

quit subcommand for dbx: End the dbx debugging session

Format
quit

Description
The quit subcommand ends the dbx debugging session.

Usage notes
• The quit subcommand can be run only while the dbx debug program is running.
• The quit subcommand is not supported in GUI (socket) mode (-p).
dbx: readwritelock

Examples
To exit the dbx debug program, enter:

quit

readwritelock subcommand for dbx: Display a list of active read/write lock objects

Format

readwritelock [number ...]
readwritelock lock
readwritelock unlock
readwritelock holder
readwrite noholder

Description

The readwritelock subcommand displays a list of active read/write lock objects for the application program. All active read/write lock objects are listed unless you use the number parameter to specify the read/write lock objects you want listed. You can also select only locked or unlocked read/write locks, or read/write locks with or without holders, by using the lock, unlock, holder, or noholder options.

In order to capture the read/write lock variables, dbx must be debugging your program before the read/write lock variable is created. You must have coded your application in one of the following ways:

- Add the following line at the top of the C program:

  ```
  #pragma runopts(TEST(ALL))
  ```

  Or:

- Code an assembler program, CEEUOPT, to invoke the CEEXOPT macro, which specifies TEST(ALL). For examples of how to code this program, see z/OS Language Environment Programming Guide.

  Or:

- Specify test(all) in the _CEE_RUNOPTS environment variable:

  ```
  export _CEE_RUNOPTS="test(all)"
  ```

Usage notes

The readwritelock subcommand can be run only while the dbx debug program is running.

Examples

1. To display all read/write lock objects, enter:

   ```
   readwritelock
   ```

2. To display read/write lock objects number 1 and number 4, enter:

   ```
   readwritelock 1 4
   ```

3. To display all locked read/write lock objects, enter:

   ```
   readwritelock lock
   ```

4. To display all unlocked read/write lock objects, enter:

   ```
   readwritelock unlock
   ```
5. To display all read/write lock objects with holders, enter:

   readwritelock holder

6. To display all read/write lock objects without holders, enter:

   mutex noholders

**record subcommand for dbx: Append user's commands to a file**

**Format**

`record filename`

**Description**

The `record` subcommand appends the user's command lines to the specified file until a `record` command is entered with no parameters.

The `record` subcommand is started by specifying a file name on the `record` command. A second `record` command with no parameters will stop the current `record` process and close the file.

**Usage notes**

The `record` subcommand can be run only while the dbx debug program is running.

**Examples**

1. To start recording the dbx commands to file `/tmp/mycmds`, enter:

   `record /tmp/mycmds`

2. To stop the previous recording to file `/tmp/mycmds`, enter:

   `record`

**Related information**

The `source` subcommand.

**registers subcommand for dbx: Display the value of registers**

**Format**

`registers [>file]`

**Description**

The `registers` subcommand displays the values of general-purpose registers, system control registers, floating-point registers, and the current instruction register, such as the program status word (PSW) for z/OS.

- General-purpose registers are denoted by the `$rnumber` variable, where the `number` argument indicates the number of the register.
- Floating-point registers are denoted by the `$fnumber` variable. By default, the floating-point registers are not displayed. To display the floating-point registers, use the `unset $noflregs` dbx subcommand.
- Vector registers can be displayed or assigned individually by vector element type using the following predefined register names:
  - `$vr0` through `$vri31`, for vector registers of type vector int.
– $vr0c through $vr31c, for vector char.
– $vr0s through $vr31s, for vector short.

Assigning a vector register can either be done a whole register at a time, for example, assign $vr0 = $vr31, or must be done an individual element at a time, using array subscripts. For example, to assign the third short element of vr15, assign $vr15s[2] = <expression>. To assign the sixteenth character element of vr30, assign $vr30c[15] = <expression>.

Options
>file
   Redirects output to the specified file.

Usage notes for the registers subcommand of dbx
1. The registers subcommand can be run only while the dbx debug program is running.
2. Vector registers can be displayed or assigned individually by vector element type using the following predefined register names:
   • $vr0 through $vr31, for vector registers of type vector int
   • $vr0c through $vr31c, for vector char
   • $vr0s through $vr31s, for vector short

Assigning a vector register can either be done a whole register at a time, for example, assign $vr0 = $vr31, or must be done an individual element at a time, using array subscripts. For example, to assign the third short element of vr15, assign $vr15s[2] = <expression>.

Examples
To display the registers, enter:

```
registers
```

Related information
The set and unset subcommands.

rerun subcommand for dbx: Begin running a program with the previous arguments

Format
```
rerun [arguments] [<file | >file | 2>file | >>file | 2>>file | >&file | >>&file]
```

Description
The rerun subcommand begins execution of the object file. The values specified with the arguments argument are passed as command-line arguments. If the arguments argument is not specified, the arguments from the last run or rerun subcommand are reused.

Options
<file
   Redirects input so that input is received from file.

>file
   Redirects output to file.

2>file
   Redirects standard error to file.
dbx: return

>>file
Append redirected output to file.

2>>file
Append redirected standard error to file.

>&file
Redirects output and standard error to file.

>>&file
Appends output and standard error to file.

Usage notes
• The `rerun` subcommand can be run only while the dbx debug program is running.
• The `rerun` subcommand is not supported in GUI (socket) mode (-p).

Examples
To rerun the program with the previously entered arguments, enter:

`rerun`

Related information
The `run` subcommand.

return subcommand for dbx: Continue running a program until a return is reached

Format
`return [procedure]`

Description
The `return` subcommand causes the program to run until a return to the procedure specified by the `procedure` argument is reached. If the `procedure` argument is not specified, execution ceases when the current procedure returns.

Usage notes
The `return` subcommand can be run only while the dbx debug program is running.

Examples
1. To continue execution to the calling routine, enter:

`return`

2. To continue execution to the `main` routine, enter:

`return main`

run subcommand for dbx: Run a program

Format
`run [arguments] [<file | >file | 2>file | >>file | 2>file | >&file | >>&file]`
**dbx: set**

**Description**
The `run` subcommand begins execution of the object file. The values specified with the `arguments` argument are passed as command-line arguments.

**Options**

<file>
Redirects input so that input is received from file.

>file
Redirects output to file.

2>file
Redirects standard error to file.

>>file
Appends redirected output to file.

2>>file
Appends redirected standard error to file.

>&file
Redirects output and standard error to file.

>>&file
Appends output and standard error to file.

**Usage notes**
- The `run` subcommand can be run only while the `dbx` debug program is running.
- The `run` subcommand is not supported in GUI (socket) mode (`-p`).

**Examples**
To run the application with the arguments `blue` and `12`, enter:

```
run blue 12
```

**Related information**
The `run` subcommand.

**set subcommand for dbx: Define a value for a dbx variable**

**Format**

```
set [variable=expression]
```

**Description**
The `set` subcommand defines a value, which is specified by the `expression` argument, for the `dbx` debug program variable, which is specified by the `variable` argument. The name of the variable must not conflict with names in the program being debugged. A variable is expanded to the corresponding expression within other commands. If the `set` subcommand is used without arguments, the variables currently set are displayed.

**Variables**
The following variables are set with the `set` subcommand:

- `$asciichars`
  Any `dbx` operation that displays the value of a character will interpret the binary representation of the character as ASCII.
$asciistrings
Any dbx operation that displays the value of a string will interpret the binary representation of the string as ASCII.

$c<n>
Condition variables

$catchbp
Catches breakpoints during the execution of the next command.

$charset
Converts character strings before displaying them. The character strings are converted from the code page srcCodePage to destCodePage. The destCodePage must be IBM-1047. The default setting is not to convert the character strings.

$commandedit
Enables the command line facility.

$current
Defined as a constant with the value of the focus thread.

$cv_events
Notifies the user but does not stop when a condition variable event is processed. The following trace information is sent to the user for the different events:

```
(dbx) cont
.
  cv initialize, object=0x2e04567
  cv wait, object=0x2e04567, mutex=0x2d04567, thid=0x0102030405060708
  cv unwait, object=0x2e04567, mutex=0x2d04567, thid=0x0102030405060708
  cv destroy, object=0x2e04567
```

$dll_loads
Set by default. When set, dbx processes symbolics for DLLs as they are loaded.

$dll_loadstop
Set by default. When set, dbx stops the function call that caused the DLL to be loaded. If the DLL was loaded due to a variable reference or an explicit load, dbx stops at the source line that caused the DLL to be loaded.

$expandunions
Displays values of each part of variant records or unions.

$expressionexhaustivesearch
Searches all scopes in a user’s program to determine and verify the scope for an expression. Selecting this option might degrade performance.

$flprecision
Determines the precision in bytes of floating-point registers when the values of the register are displayed; for example, in expressions or during assignment. Valid values are 4, 8 or 16.

$fr<n>
Hexadecimal floating-point register.

$frb<n>
Binary floating-point register

$frd<n>
Decimal floating-point register

$hexchars
Prints characters in hexadecimal format.

$hexin
Input is interpreted in hexadecimal format.
Restriction: The $hexin variable is only supported in dbx command-line mode and does not affect the interpretation of GUI input. If the user of the GUI debugger wants input to be interpreted in hexadecimal format, the input must be prefixed with "0x".

\$hexinta
Prints integers in decimal format instead of hexadecimal format.

\$historypage
Specifies the number of history items to be traversed when using the page up and page down keys.

\$history_unique
Prevents consecutive duplicate commands from being saved to the history list.

\$historywindow
Specifies the number of commands to display and retain in the history list.

\$hold_next
Automatically holds all threads except the focus thread during the next, nexti, step, or stepi command execution. If not set, all threads resume execution and may reach the breakpoint set by the next, nexti, step, or stepi command execution.

\$l<n>
Read/write locks variables.

\$listwindow
Specifies the number of lines to list around a function and to list when the list subcommand is used without parameters.

\$lv_events
When set, dbx notifies the user but does not stop when a read/write lock object event is processed. The following trace information is sent to the user for the different events:

(dbx) cont

lv initialize, object=0x2d04567
lv wait, object=0x2d04567, thid=0x0102030405060708
lv unwait, object=0x2d04567, thid=0x0102030405060708
lv lock, object=0x2d04567, thid=0x0102030405060708
lv unlock, object=0x2d04567, thid=0x0102030405060708
lv relock, object=0x2d04567, thid=0x0102030405060708
lv unrelock, object=0x2d04567, thid=0x0102030405060708
lv destroy, object=0x2d04567

\$m<n>
Specifies mutex variables.

\$maxstring
Specifies the maximum number of characters to be displayed when printing a string. String printing stops when $maxstring characters are printed. Set to zero to completely display strings. The default value is zero.

\$mv_events
When set, dbx notifies the user but does not stop when a mutex object event is processed. The following trace information is sent to the user for the different events:
$noargs
Omits arguments from subcommands, such as where, up, down, and dump.

$noflbregs
Does not display the binary floating point representation of the floating point registers with the registers subcommand.

$nofldregs
Does not display the decimal floating point representation of the floating point registers with the registers subcommand.

$noflregs
Does not display the hexadecimal floating point representation of the floating point registers with the registers subcommand.

$novregs
When set, omits the display of vector registers from the registers subcommand.

$octin
Interprets input in octal format. The $octin variable is only supported in dbx command-line mode and does not affect the interpretation of GUI input. If the user of the GUI debugger wants input to be interpreted in octal format, the input must be prefixed with "0".

$octints
Prints integers in octal format.

$pc
Program counter register.

$psw
First word of the program status word register.

$psw0
First word of the program status word register.

$psw1
Second word of the program status word register.

$r<n>
General register.

$r_precision
Sets the amount of precision, in bytes, to use when displaying an integer value. Possible values are 4 and 8.

$repeat
Repeats the previous command if no command was entered.

$showbases
Displays the base class data when a derived class is printed.

$showcodelines
Indicates the lines where the debugger can stop and where breakpoints can be set.

$sigblock
Blocks all signals from reaching the program being debugged.
**$sticky_debug**
When set, dbx will recognize sticky bit programs and DLLs in the loadmap.

**$tv_events**
Notifies the user but does not stop when a thread object event is processed. Trace information similar to the following example is sent to the user for the different events:

```plaintext
bdb: cont.

IPT create, thid=0x1234567890123456, stack=5200
IPT exit, thid=0x1234567890123456
tv create, thid=0x1234567890123456, created thid=0x1234567890123422, stack=5200
tv created, thid=0x1234567890123456, stack=5200
tv exit, thid=0x1234567890123456
tv wait, thid=0x1234567890123456, joining thid=0x1234567890123422
tv unwait, thid=0x1234567890123456, joined thid=0x1234567890123422
```

**$unsafeassign**
Turns off strict type checking between the two sides of an `assign` subcommand.

**$unsafebounds**
Turns off subscript checking on arrays.

**$unsafegoto**
Turns off the `goto` subcommand destination checking.

**Usage notes**

1. The $unsafe variables limit the usefulness of the dbx debug program in detecting errors.
2. The `set` subcommand can be run only while the dbx debug program is running.

**Examples**

1. To change the default number of lines to be listed to 20, enter:
   ```plaintext
   set $listwindow=20
   ```
2. To disable type checking on the `assign` subcommand, enter:
   ```plaintext
   set $unsafeassign
   ```

**Related information**

The `unset` subcommand.

**sh subcommand for dbx: Pass a command to the shell for execution**

**Format**

```plaintext
sh [command]
```

**Description**

The `sh` subcommand passes the command specified by the `command` parameter to the shell for execution. The SHELL environment variable determines which shell is used. The default is the `sh` shell. If no argument is specified, control is transferred to the shell.
Usage notes
• The `sh` subcommand can be run only while the `dbx` debug program is running.
• The `sh` subcommand is not supported in GUI (socket) mode (-p).

Examples
1. To run the `ls` command, enter:
   `sh ls`
2. To escape to a shell, enter:
   `sh`

skip subcommand for `dbx`: Continue from the current stopping point

Format
`skip [number]`

Description
The `skip` subcommand continues execution of the application program from the current stopping point. A number of breakpoints equal to the value of the `number` argument are skipped, and execution then ceases when the next breakpoint is reached or when the program finishes. If the `number` argument is not specified, it defaults to a value of 1.

Usage notes
The `skip` subcommand can be run only while the `dbx` debug program is running.

Examples
To continue execution until the second breakpoint is encountered, enter:
`skip 1`

Related information
The `cont` subcommand.

source subcommand for `dbx`: Read subcommands from a file

Format
`source file`

Description
The `source` subcommand reads `dbx` subcommands from the file specified by the `file` argument.

Usage notes
The `source` subcommand can be run only while the `dbx` debug program is running.
To read the dbx subcommands in the cmdfile file, enter:

```
source cmdfile
```

**status subcommand for dbx: Display the active trace and stop subcommands**

**Format**

```
status [>file]
```

**Description**

The status subcommand displays the trace and stop subcommands currently active. The > option sends the output of the status subcommand to a file specified in the file argument.

**Options**

>file

Redirects output to file.

**Usage notes**

The status subcommand can be run only while the dbx debug program is running.

**Examples**

1. To display the currently active trace and stop subcommands, enter:

```
status
```

2. To stop at line 52 only when thread $t2 reaches that line, enter:

```
stop at 52 if $t2==$current
```

**Related information**

The clear, delete, stop, and trace subcommands.

**step subcommand for dbx: Run one or more source lines**

**Format**

```
step [number]
```

**Description**

The step subcommand runs source lines of the program. Specify the number of lines to be run with the number argument. If the number argument is omitted, it defaults to a value of 1.

**Usage notes**

1. The step subcommand can be run only while the dbx debug program is running.
2. If the $hold_next variable is defined, dbx holds all threads except the focus thread during a step subcommand. Then dbx unholds the threads after the step subcommand finishes.
Examples
1. To continue execution for one source line, enter:
   step
2. To continue execution for five source lines, enter:
   step 5

Related information
The cont, goto, next, and stepi subcommands.

steipi subcommand for dbx: Run one or more machine instructions

Format
steipi[number]

Description
The steipi subcommand runs instructions of the program. Specify the number of instructions to be run in the number argument. If the number argument is omitted, it defaults to 1.

Usage notes
1. The steipi subcommand can be run only while the dbx debug program is running.
2. If the $hold_next variable is defined, dbx holds all threads except the focus thread during a steipi subcommand. Then dbx unholds the threads after the steipi subcommand finishes.

Examples
1. To continue execution for one machine instruction, enter:
   steipi
2. To continue execution for five machine instructions, enter:
   steipi 5

Related information
The gotoi, nexti, and step subcommands.

stop subcommand for dbx: Stop execution of a program

Format
stop if condition
stop [variable] at ["filename":]sourceline [if condition]
stop [variable] in procedure [if condition]
stop variable [if condition]

Description
The stop subcommand stops execution of the program when certain conditions are fulfilled. The program is stopped when:
The condition is true, if the if condition option is used.
The sourceline line number is reached, if the at sourceline option is used.
The procedure is called, if the in procedure option is used.
The variable is changed, if the variable argument is specified.

The dbx debug program associates event numbers with each stop subcommand. To view these numbers, use the status subcommand. To turn stop off, use the delete or clear subcommand.

Options

current line

at ["filename"] sourceline
Specifies the source line number in either the specified filename or the file that is currently being debugged. If a specific file name is specified, the filename must be enclosed with quotation marks and a colon must separate the "filename" from the sourceline. For example:

```
stop at "myfile":1234
```

if condition
Specifies the condition, such as true.
in procedure
Specifies the procedure to be called.

Usage notes

The stop subcommand can be run only while the dbx debug program is running.

Examples

1. To stop execution at the first executable statement in the main procedure, enter:

   

   stop in main

2. To stop execution when the value of the x variable is changed on line 12 of the execution, enter:

   

   stop x at 12

3. To stop execution at a specified line (line 23), when the value of the variable (myvar) is greater than 2, enter:

   

   stop at 23 if myvar > 2

4. To stop at line 52 only when thread $t2 reaches that line, enter:

   

   stop at 42 if $t2==$current

        Related information
The stopi, delete, clear, and trace subcommands.

stopi subcommand for dbx: Stop at a specified location

Format

stopi address [if condition]
stopi [address] at address [if condition]
stopi [address] in procedure [if condition]
The stopi subcommand sets a stop at the specified location.

- With the `if` condition option, the program stops when the condition is true.
- With the `address` argument, the program stops when the contents of `address` change.
- With the `ataddress` option, a stop is set at the specified address.
- With the `inprocedure` option, the program stops when the procedure specified with the `procedure` argument is called.

### Options

**if**

Specifies the condition, such as `true`.

**inprocedure**

Specifies the procedure to be called.

**ataddress**

Specifies the machine instruction address.

### Usage notes

The `stopi` subcommand can be run only while the dbx debug program is running.

### Examples

1. To stop execution at address 0X100020F0, enter:
   ```
   stopi at 0x100020f0
   ```

2. To stop execution when the contents of address 0X100020F0 change, enter:
   ```
   stopi 0x100020f0
   ```

3. To stop at address 0x2d04567 only when thread $t2 reaches that address, enter:
   ```
   stopi at 0x2d04567 if $t2=$current
   ```

### Related information

The `stop` subcommand.

**thread subcommand for dbx: Display a list of active threads**

### Format

```
thread [number ...]
thread hold [number...]
thread unhold [number ...]
thread info [number ...]
thread current [number ...]
thread activ
thread async
thread dead
thread pcanc
```
**dbx: thread**

**Description**

The thread subcommand displays a list of active threads for the application program. All active threads are listed unless you use the `number` parameter to specify the threads you want listed. You can also select threads by their states using the `activ`, `async`, `dead`, or `pcanc` options.

You can use the `info` option to display full information about a thread, and threads can be held or unheld with the `hold` or `unhold` options. The focus thread is defaulted to the running thread; `dbx` uses it as the context for normal `dbx` subcommands such as `register`. You can use the `current` option to switch the `dbx` focus thread.

**Examples**

1. To display all thread objects, enter:
   ```
   thread
   ```

2. To display thread objects number 1 and 2, enter:
   ```
   thread 1 2
   ```

3. To display all active threads, enter:
   ```
   thread activ
   ```

4. To display all threads in dead state, enter:
   ```
   thread dead
   ```

5. To display all threads in async state (that is, threads with a cancelability type of `PTHREAD_INTRASYNCHRONOUS`) that are waiting to be scheduled, enter:
   ```
   thread async
   ```

   Because this thread was created with the `PTATASYNCHRONOUS` attribute and the limit was reached, this thread was queued for execution. For example, if the thread limit is set to ten and there are 12 threads, two of them will be shown as async for the `dbx` thread command.

6. To display all threads in pcanc state (that is, threads that have been requested to be canceled by `pthread_cancel()`), enter:
   ```
   thread pcanc
   ```

7. To hold all threads, enter:
   ```
   thread hold
   ```

8. To hold thread number 1 and 4, enter:
   ```
   thread hold 1 4
   ```

9. To unhold thread number 1 and 4, enter:
   ```
   thread unhold 1 4
   ```

10. To display the focus thread, enter:
    ```
    thread current
    ```

11. To set the focus thread to thread number 1, enter:
    ```
    thread current 1
    ```
12. To get information about thread number 3, enter:

```
thread info 3
```

**trace subcommand for dbx: Print tracing information**

**Format**

```
trace [if condition]
trace procedure [if condition]
trace [variable] at sourceline [if condition]
trace [variable] in procedure [if condition]
trace sourceline [if condition]
trace expression at sourceline [if condition]
```

**Description**

The `trace` subcommand prints tracing information for the specified procedure, function, source line, expression, or variable when the program runs. A condition can be specified. The `dbx` debug program associates a number with each `trace` subcommand. To view these numbers, use the `status` subcommand. To turn tracing off, use the `delete` subcommand.

**Options**

- **at sourceline**
  
  Specifies the source line at which to find the expression being traced.

- **if condition**
  
  Specifies a condition for the beginning of the trace. The trace begins only if `if condition` is true.

- **in procedure**
  
  Specifies the procedure in which to find the procedure or variable being traced.

**Usage notes**

The `trace` subcommand can be run only while the `dbx` debug program is running.

**Examples**

1. To trace each call to the `printf()` procedure, enter:

   ```
   trace printf
   ```

2. To trace each execution of line 22 in the `hello.c` file, enter:

   ```
   trace "hello.c":22
   ```

3. To trace changes to the `x` variable within the `main` procedure, enter:

   ```
   trace x in main
   ```

4. To trace at line 52 only when mutex `$m1` is not held, enter:

   ```
   trace at 52 if $m2.lock == 0
   ```

**Related information**

The `tracei` subcommand.
tracei subcommand for dbx: Turn on tracing

Format

```
tracei [if condition]
tracei address [at address] [if condition]
tracei address [in procedure] [if condition]
tracei expression at address [if condition]
```

Description

The `tracei` subcommand turns on tracing when:

- The contents of the storage at the address change, if the `address` argument is specified.
- The instruction at the specified address is executed, if the `at address` option is specified.
- The procedure specified by `procedure` is active, if the `in procedure` option is included.
- The condition specified by the `condition` argument is true, if the `if condition` option is included.

Options

- **at address**
  Specifies an address. Tracing is enabled when the contents of this address change.

- **if condition**
  Specifies a condition, the meeting of which causes tracing to be enabled.

- **in procedure**
  Specifies a procedure. Tracing is enabled when this procedure is active.

Usage notes

The `tracei` subcommand can be run only while the dbx debug program is running.

Examples

1. To trace each instruction run, enter:
   ```
   tracei
   ```

2. To trace each time the instruction at address 0X100020F0 is run, enter:
   ```
   tracei at 0x100020f0
   ```

3. To trace each time the contents of memory location 0X20004020 change while the `main` procedure is active, enter:
   ```
   tracei 0x20004020 in main
   ```

4. To trace at address 0x2d04567 only when thread $t2 reaches that address, enter:
   ```
   tracei at 0x2d04567 if $t2=$current
   ```

Related information

The `trace` subcommand.
unalias subcommand for dbx: Remove an alias

Format
unalias name

Description
The unalias subcommand removes the alias specified by the name argument.

Usage notes
The unalias subcommand can be run only while the dbx debug program is running.

Examples
To remove an alias named printx, enter:

```
unalias printx
```

Related information
The alias subcommand.

unset subcommand for dbx: Delete a variable

Format
unset name

Description
The unset subcommand deletes the debug program variable associated with the name specified by the name argument.

Usage notes
The unset subcommand can be run only while the dbx debug program is running.

Examples
To delete the variable inhibiting the display of floating-point registers, enter:

```
unset $noflregs
```

Related information
The set subcommand.

up subcommand for dbx: Move the current function up the stack

Format
up [count]

Description
The up subcommand moves the current function up the stack count number of levels. The current function is used for resolving names. The default for the count argument is 1.
dbx: use

Usage notes
The up subcommand can be run only while the dbx debug program is running.

Examples
1. To move the current function up the stack two levels, enter:
   ```
   up 2
   ```
2. To display the current function on the stack, enter:
   ```
   up 0
   ```

Related information
The down subcommand.

use subcommand for dbx: Set the list of directories to be searched

Format
use [directory...]

Description
The use subcommand sets the list of directories to be searched when the dbx debug program looks for source files. If the use subcommand is specified without arguments, the current list of directories to be searched is displayed.

If the C primary source is in an MVS data set, the use subcommand can be specified with a double-slash (//) argument to indicate that the source file be sought outside the file system.

Usage notes
The use subcommand can be run only while the dbx debug program is running.

Examples
1. To change the list of directories to be searched to the working directory, the parent directory, and /tmp, enter:
   ```
   use . .. /tmp
   ```
2. To change the list of directories to be searched to look for the C source as an MVS data set, enter:
   ```
   use //
   ```

Related information
The edit and list subcommands.

whatis subcommand for dbx: Display the type of program components

Format
whatis name
Description
The `whatis` subcommand displays the declaration of `name`, where the `name` argument designates a variable, procedure, or function name, optionally qualified with a block name.

Usage notes
1. Variables declared with the `const` attribute (in C programs) are displayed without the `const` attribute.
2. The `whatis` subcommand can be run only while the `dbx` debug program is running.

Examples
1. To display the declaration of the `x` variable, enter:
   
   ```bash
   whatis x
   ```

2. To display the declaration of the `main` function, enter:

   ```bash
   whatis main
   ```

3. To display the declaration of the `x` variable within the `main` function, enter:

   ```bash
   whatis main.x
   ```

4. To display the declaration of a specific condition variable, `$c1`, enter:

   ```bash
   whatis $c1
   ```

5. To display the declaration of a specific mutex object, `$m1`, enter:

   ```bash
   whatis $m1
   ```

6. To display the declaration of a specific thread, `$t1`, enter:

   ```bash
   whatis $t1
   ```

`where` subcommand for `dbx`: List active procedures and functions

Format
```
where [>file]
```

Description
The `where` subcommand displays a list of active procedures and functions. By using the `>file` option, you can redirect the output of this subcommand to the specified file.

Options
```
>flag
```
Redirects output to the specified file.

Usage notes
The `where` subcommand can be run only while the `dbx` debug program is running.

Examples
To display the list of active routines, enter:

```bash
where
```
Related information
The up and down subcommands.

whereis subcommand for dbx: Display the full qualifications of symbols

Format
whereis [-exhaustive] identifier

Description
The whereis subcommand displays the full qualifications of all the symbols whose names match the specified identifier. The order in which the symbols print is not significant.

Options
-exhaustive
Specifies that whereis is to search for symbols in all compile units. This option only applies when a module map is being used. If one is being used, then the default is to use quick mode, which searches for symbols only in the current compile unit and global lists. In that case, using this option might degrade performance. If a module map is not being used, then this subcommand always searches for symbols in all compile units.

Usage notes
1. The whereis subcommand can be run only while the dbx debug program is running.

Examples
1. To display the qualified names of all symbols named x, enter:
   
   ```
   whereis x
   ```
   
   An example of a possible output is:
   
   ```
   "x1.c".x
   FDBX9998: whereis quick mode will only search symbol x in current CU and global lists. To list all symbol x, please use whereis -exhaustive x, but that will cause all debug data files to be loaded and performance will be degraded.
   ```

2. To display the qualified names of all symbols named x in exhaustive mode, enter:
   
   ```
   whereis -exhaustive x
   ```
   
   An example of a possible output is:
   
   ```
   
   
   ."x1.c".x
   ."x2.c".x
   ```

Related information
The which subcommand.
which subcommand for dbx: Display the full qualification of an identifier

Format

```
which identifier
```

Description

The `which` subcommand displays the full qualification of the given identifier. The full qualification consists of a list of the outer blocks with which the identifier is associated.

Usage notes

The `which` subcommand can be run only while the `dbx` debug program is running.

Examples

To display the full qualification of the `x` symbol, enter:

```
which x
```

Related information

The `whereis` subcommand.

dd - Convert and copy a file

Format

```
dd [bs=size] [cbs=size] [conv=conversion] [count=n] [ibs=size] [if=file] [imsg=string] [iseek=n] [obs=size] [of=file] [omsg=string] [seek=n] [skip=n]
```

Description

`dd` reads and writes data by blocks. It can convert data between formats. It is frequently used for such devices as tapes that have discrete block sizes, or for fast multisector reads from disks. `dd` performs conversions to accommodate nonprogrammable terminals, which require deblocking, conversion to and from EBCDIC, and fixed-length records.

`dd` processes the input data as follows:

1. `dd` reads an input block.
2. If this input block is smaller than the specified input block size, `dd` pads it to the specified size with null bytes. When you also specify a block or unblock conversion, `dd` uses spaces instead of null bytes.
3. If you specified `bs=s` and requested no conversion other than `sync` or `noerror`, `dd` writes the padded (if necessary) input block to the output as a single block and omits the remaining steps.
4. If you specified the `swab` conversion, `dd` swaps each pair of input bytes. If there is an odd number of input bytes, `dd` does not attempt to swap the last byte.
5. `dd` performs all remaining conversions on the input data independently of the input block boundaries. A fixed-length input or output record may span these boundaries.
6. `dd` gathers the converted data into output blocks of the specified size. When `dd` reaches the end of the input, it writes the remaining output as a block (without padding if `conv=sync` is not specified). As a result, the final output block might be shorter than the output block size.
Options

bs=size
Sets both input and output block sizes to size bytes. You can suffix this decimal number with w, b, k, or x number, to multiply it by 2, 512, 1024, or number, respectively. You can also specify size as two decimal numbers (with or without suffixes) separated by x to indicate the product of the two values. Processing is faster when ibs and obs are equal, since this avoids buffer copying. The default block size is 1B. bs=size supersedes any settings of ibs=size or obs=size.

If you specify bs=size and you request no other conversions than noerror, notrunc, or sync, dd writes the data from each input block as a separate output block. If the input data is less than a full block and you did not request sync conversion, the output block is the same size as the input block.

cbs=size
Sets the size of the conversion buffer that is used by various conv options.

conv=conversion[, conversion, ...]
conversion can be any one of the following:

ascii
Converts EBCDIC input to ASCII for output; it is provided for compatibility purposes only.

To copy a file and convert between a shell code page and ASCII, use iconv, not dd.

block
Converts variable-length records to fixed-length records. dd treats the input data as a sequence of variable-length records (each terminated by a newline or an EOF character) independent of the block boundaries. dd converts each input record by first removing any newline characters and then padding (with spaces) or truncating the record to the size of the conversion buffer. dd reports the number of truncated records on standard error. You must specify cbs=size with this conversion.

Note: When working with double-byte characters, dd truncates the record after the last complete double-byte character that will fit in the conversion buffer. dd then pads the record with spaces if it is shorter than the conversion buffer size.

convfile
Uses convfile as a translation table if it is not one of the conversion formats listed here and it is the name of a file of exactly 256 bytes.

You can perform multiple conversions at the same time by separating arguments to conv with commas; however, some conversions are mutually exclusive (for example, ucase and lcase).

Note:
1. When you specify one or more of the character set conversions (ascii, ebcdic, ibm, or convfile), dd assumes that all characters are single-byte characters, regardless of the locale. Do not use these conversions with double-byte character sets.

2. When working with DBCS text, dd treats the input and output files as character strings and handles DBCS characters correctly (no splitting and retaining of proper shift states). This happens only if any of the conversion options (block, unblock, ucase, or lcase) are specified. Otherwise, DBCS strings can be corrupted with the seek, count, or iseek processing.

ebcdic
Converts ASCII input to EBCDIC for output; it is provided for compatibility purposes only.

To copy a file and convert between a shell code page and ASCII, use iconv, not dd.

ibm
Like ebcdic, converts ASCII to EBCDIC; it is provided for compatibility purposes only.

To copy a file and convert between code page 01047 (used in the z/OS shell) and ASCII, use iconv, not dd.
lcase
  Converts uppercase input to lowercase.

noerror
  Ignores errors on input.

notrunc
  Does not truncate the output file. dd preserves blocks in the output file that it does not explicitly write to.

swab
  Swaps the order of every pair of input bytes. If the current input record has an odd number of bytes, this conversion does not attempt to swap the last byte of the record.

sync
  Specifies that dd is to pad any input block shorter than ibs to that size with NUL bytes before conversion and output. If you also specified block or unblock, dd uses spaces instead of null bytes for padding.

ucase
  Converts lowercase input to uppercase.

unblock
  Converts fixed-length records to variable-length records by reading a number of bytes equal to the size of the conversion buffer, deleting all trailing spaces, and appending a newline character. You must specify cbs=size with this conversion.

count=n
  Copies only n input blocks to the output.

ibs=size
  Sets the input block size in bytes. You specify it in the same way as with the bs option.

if=file
  Reads input data from file. If you don't specify this option, dd reads data from standard input.

ims=string
  Displays string when all data has been read from the current volume, replacing all occurrences of %d in string with the number of the next volume to be read. dd then reads and discards a line from the controlling terminal.

iseek=n
  seeks to the n-th block of the input file. The distinction between this and the skip option is that isseek does not read the discarded data. There are some devices, however, such as tape drives and communication lines, on which seeking is not possible, so only skip is appropriate.

obs=size
  Sets the output block size in bytes. You specify it in the same way as the bs value. The size of the destination should be a multiple of the value chosen for size. For example, if you choose obs=10K, the destination's size should be a multiple of 10K.

of=file
  Writes output data to file. If you don't specify this option, dd writes data to standard output. dd truncates the output file before writing to it, unless you specified the seek=n operand. If you specify seek=n, but do not specify conv=notrunc, dd preserves only those blocks in the output file over which it seeks. If the size of the seek plus the size of the input file is less than the size of the output file, this can result in a shortened output file.

omsg=string
  Displays string when dd runs out of room while writing to the current volume. Any occurrences of %d in string are replaced with the number of the next volume to be written. dd then reads and discards a line from the controlling terminal.

seek=n
  Initially seeks to the n-th block of the output file.

  Note: Use caution when working with DBCS characters and the seek option. Seeking into the output file that contains DBCS characters can cause the DBCS string in the output file to be corrupted. Be
sure that the seek count is not aligned with an existing DBCS string in the output file. Otherwise, part of the existing DBCS string either is written over with single-byte data or has extra shift codes from the input file’s DBCS data.

**skip=n**

Reads and discards the first n blocks of input.

**Examples**

Entering:

```
dd if=in of=out conv=ascii cbs=80 ibs=6400 obs=512
```

converts 80-byte fixed-length EBCDIC card images in 6400-byte input blocks to variable-length ASCII lines, 512 bytes to the output block.

**Localization**

dd uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Exit values**

0  
Successful completion.

1  
Failure due to any of the following reasons:
  - I/O errors on read/write.
  - Incorrect command-line option.
  - Incorrect arguments to a conversion.

2  
Failure resulting in a usage message such as:
  - An option that should contain = does not.
  - Unknown or incorrect command-line option.

**Messages**

Possible error messages include:

**badly formed number number**

A value specified as a number (for example, a block size) does not have the form of a number as recognized by dd. For example, you may have followed the number with a letter that dd does not recognize as a block-size unit (w, b, k).

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

The conv=convfile, iseek, insg, and omsg options plus the w suffix described in the bs= option are all extensions of the POSIX standard.
**df - Display the amount of free space in the file system**

**Format**
```
df [-kPStv][file ...]
```

**Description**
`df` shows the amount of free space that is left on a file system. Space can have the following values:

**Space Used**
Total amount of space that is allocated to existing files in the file system.

**Space Free**
Total amount of space available in file system for the creation of new files by unprivileged users.

**Space Reserved**
Space reserved by the system, which is not normally available to a user.

**Total Space**
Includes space used, space free, and space reserved.

`df` measures space in units of 512-byte disk sectors. You can specify a particular file system by naming any file name on that file system. If you do not give an argument, `df` reports space for all mounted file systems known to the system, in the following format:

* File system root.
* File system name.
* Space available and total space.
  The total space reported is the space in the already allocated extents (primary and any already allocated secondary extents) of the data set that holds this file system. Therefore, the total space might increase as new extents are allocated.
* Number of free files (inodes).
  This number is only meaningful for file systems created using DFSMS 1.3.0 and later. For file systems created with earlier versions of DFSMS, this number is always 4 294 967 295.
* File system status.

**Tip:** For zFS file systems, the `df` command might not provide sufficient information to indicate whether a file system is running out of space. For complete information about zFS space usage, use the `zfsadm aggrinfo -long` command. See *z/OS Distributed File Service zFS Administration* for more information.

**Options**

- `-k`
  Uses 1024-byte (1 KB) units instead of the default 512-byte units when reporting space information.

- `-P`
  Lists complete information about space used, in the following order:
  * File system name.
  * Total space.
  * Space used.
  * Space free.
  * Percentage of space used.
  * File system root.
Display SMF accounting fields.

Display total allocated file slots, in addition to the total number of free files that are already displayed.

Lists more detailed information about the file system status.

- File system root.
- File system name.
- Space available and total space.
- Number of free files (inode).
- File system status.
- File system type, mode bits, and device number.
- File system mount parm data.
- File system mount tag value.
- Whether ACLs are supported by the security product and file system.
- Aggregate name, if one exists.
- File system ID issuing a quiesce request.
- User name and effective UID of the user who mounted the file system, if it was a nonprivileged user mount.

For systems in a shared file system environment, the following additional fields are displayed:

- File system ID (owner/mounted file system server)
- File system automove status (yes-Y, no-N, include-I, exclude-E or unmount-U)
- File system client status
- System list and include/exclude indicator, if the system list exists
- PFS normal status, if one exists
- PFS exception status, if one exists

Usage notes

1. Some information might not be available if the underlying physical file system (PFS) could not be provided because of a heterogeneous environment or Protocol limitation.

Example: In the following example, the df command that displays an NFS Version 2 mount point shows 0 under the Files field.

```
df /fsdirs2
Mounted on     Filesystem         Avail/Total     Files       Status
/fsdirs2       (fs2)           8065166/8112960  0
Available
```

Examples

If you issue a df -v on a file system whose owner is participating in shared file system, status information such as the following example is displayed:

```
Mounted on     Filesystem         Avail/Total     Files       Status
/u/billyjc     (OMVS.ZFS.BILLYJC)  365824/3165120  4294924769  Available
ZFS, Read/Write, Device: 17, ACLs=Y, No SUID, Exported, No Security
FSFULL(90,1)
File System owner: AQFT    Automove=E    Client=N
System List (Exclude): sysname1 sysname2 ... sysnameN
Quiesce Owner : AQTS    Quiesce Jobname : MEGA    Quiesce PID: 16777321
Filetag : T=on    codeset=ISO8859-1
Aggregate Name: POSIX.ZFS.ETC
```
Localization

df uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Exit values

0
Successful completion.

1
Failure due to any of the following reasons:
• Inability to access filename.
• Inability to access device.
• device is not a device.

2
Incorrect command-line option.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.

Related information
du, ls

diff - Compare two text files and show the differences

Format

diff [-BbefHhimNnrsw] [-C n] [-c[n]] [-Difname] [M mark] [-W option[option] ... path1 path2

Description

The diff command attempts to determine the minimal set of changes needed to convert a file whose name is specified by the path1 argument into the file specified by the path2 argument.

Input files must be text files. If either (but only one) file name is --, diff uses a copy of the standard input (stdin) for that file. If exactly one of path1 or path2 is a directory, diff uses a file in that directory with the same name as the other file name. If both are directories, diff compares files with the same file names under the two directories; however, it does not compare files in subdirectories unless you specify the -r option. When comparing two directories, diff does not compare character special files, or FIFO special files with any other files.

By default, output consists of descriptions of the changes in a style like that of the ed text editor. A line indicating the type of change is given. The three types are a (append), d (delete), and c (change). The output is symmetric: A delete in path1 is the counterpart of an append in path2. diff prefixes each operation with a line number (or range) in path1 and suffixes each with a line number (or range) in path2.
After the line giving the type of change, **diff** displays the deleted or added lines, prefixing lines from *path1* with `<` and lines from *path2* with `>.

**Options**

Options that control the output or style of file comparison are:

-B
Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-b
Ignores trailing blanks and tabs and considers adjacent groups of blanks and tabs elsewhere in input lines to be equivalent.

For example, if one file contained a string of three spaces and a tab at a given location while the other file contained a string of two spaces at the same location, **diff** would not report this as a difference.

-C n
Shows n lines of context before and after each change. **diff** marks lines removed from *path1* with –, lines added to *path2* with +, and lines changed in both files with !.

-c[n]
Is equivalent to –Cn, but n is optional. The default value for n is 3. **diff** marks lines removed from *path1* with –, lines added to *path2* with +, and lines changed in both files with !.

-D ifname
Displays output that is the appropriate input to the C preprocessor to generate the contents of *path2* when *ifname* is defined, and the contents of *path1* when *ifname* is not defined.

-e
Writes out a script of commands for the ed text editor, which converts *path1* to *path2*. **diff** sends the output to the standard output (stdout).

-f
Writes a script to stdout that shows modifications necessary to convert *path1* to *path2* in the reverse order of that produced by the -e option. However, the script is not in a form that is suitable for use with the ed editor. The commands produced is reversed from that produced by -e, and the line number ranges are separated by spaces, rather than commas. This option conflicts with the -m option.

-H
Uses the half-hearted (-h) algorithm only if the normal algorithm runs out of system resources.

-h
Uses a fast, half-hearted algorithm instead of the normal **diff** algorithm. This algorithm can handle arbitrarily large files; however, it is not good at finding a minimal set of differences in files with many differences.

-i
Ignores the case of letters when doing the comparison.

-m
Produces the contents of *path2* with extra formatter request lines interspersed to show which lines were added (those with vertical bars in the right margin) and deleted (indicated by a * in the right margin).

-M
Is an internal option and is not supported.

-N
Is an internal option and is not supported.

-N
Is an internal option and is not supported.
-r
Compares corresponding files under the directories, and recursively compares corresponding files
under corresponding subdirectories under the directories. You can use this option when you specify
two directory names on the command line.

-s
Compares two directories, file by file, and prints messages for identical files between the two
directories.

-w
Ignores white space during the comparison process.

-W option[,option]...
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

filecodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded
character set of the file is codeset. codeset can be a code set name known to the system or a
numeric coded character set identifier (CCSID). Note that the command iconv -1 lists existing
CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset
options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is
ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor
pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or
the _TEXT_CONV environment variable indicates text conversion. For more information about text
conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page
993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this
command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System
Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For
more information about supported code sets, see z/OS Unicode Services User's Guide and
Reference.

pgmcodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded
character set of the program (command) is codeset. codeset can be a code set name known to the
system or a numeric coded character set identifier (CCSID). Note that the command iconv -1 lists existing
CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is
IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur
unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text
conversion. For more information about text conversion, see Appendix L, “Controlling text
conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this
command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System
Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

Examples
1. To compare two text files containing UTF-8 characters and show the differences, assuming that:
   • The text files are untagged and you do not want to tag them or enable automatic conversion, and
• You cannot alter the tag (for example, you are comparing untagged public text files or read-only text files):

```bash
diff -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File01 myUtf8File02
```

2. To compare two text files containing EBCDIC characters and show the differences, assuming that automatic conversion has been enabled but the text files are incorrectly tagged as ASCII:

```bash
diff -B myMisTaggedFile01 myMisTaggedFile02
```

3. The following example illustrates the effect of the -c option on the output of the `diff` command. The following two files, price1 and price2, are compared with and without the use of the -c option.

The contents of `price1` are as follows:

```
Company X Price List:
$  0.39  -- Package of Groat Clusters  
$  5.00  -- Candy Apple Sampler Pack    
$12.00  -- Box of Crunchy Frog Chocolates  
$ 15.99  -- Instant Rain (Just Add Water)  
$ 20.00  -- Asparagus Firmness Meter  
$ 25.00  -- Package of Seeds for 35 Herbs  
$ 30.00  -- Child's Riding Hood (Red)  
$ 35.00  -- Genuine Placebos  
$ 45.00  -- Case of Simulated Soy Bean Oil  
$ 75.88  -- No-Name Contact Lenses  
$ 99.99  -- Kiddie Destructo-Bot  
$125.00  -- Emperor's New Clothes  
```

The contents of `price2` are as follows:

```
Company X Price List:
$  0.39  -- Package of Groat Clusters  
$  5.49  -- Candy Apple Sampler Pack  
$12.00  -- Box of Crunchy Frog Chocolates  
$ 15.99  -- Instant Rain (Just Add Water)  
$ 20.00  -- Asparagus Firmness Meter  
$ 25.00  -- Package of Seeds for 35 Herbs  
$ 30.00  -- Child's Riding Hood (Red)  
$ 35.00  -- Genuine Placebos  
$ 45.00  -- Case of Simulated Soy Bean Oil  
$ 75.88  -- No-Name Contact Lenses  
$ 99.99  -- Kiddie Destructo-Bot  
```

The command:

```
diff price1 price2
```

results in the following display:

```
3c3
< $  5.00  -- Candy Apple Sampler Pack  
<  $  5.49  -- Candy Apple Sampler Pack  
<  $ 17.00  -- Simulated Naugahyde cleaner  
<  $ 20.00  -- Asparagus Firmness Meter  
<  $ 25.00  -- Package of Seeds for 35 Herbs  
<  $ 30.00  -- Child's Riding Hood (Red)  
<  $ 35.00  -- Genuine Placebos  
<  $ 45.00  -- Case of Simulated Soy Bean Oil  
<  $ 75.88  -- No-Name Contact Lenses  
<  $ 99.99  -- Kiddie Destructo-Bot  
```

The addition of the -c option, as in:

```
diff -c price1 price2
```

results in the following display:

```
3c3
< $  5.00  -- Candy Apple Sampler Pack  
<  $  5.49  -- Candy Apple Sampler Pack  
<  $ 17.00  -- Simulated Naugahyde cleaner  
<  $ 20.00  -- Asparagus Firmness Meter  
<  $ 25.00  -- Package of Seeds for 35 Herbs  
<  $ 30.00  -- Child's Riding Hood (Red)  
<  $ 35.00  -- Genuine Placebos  
<  $ 45.00  -- Case of Simulated Soy Bean Oil  
<  $ 75.88  -- No-Name Contact Lenses  
<  $ 99.99  -- Kiddie Destructo-Bot  
```
diff -c marks lines removed from price1 with –, lines added to price1 with + and lines changed in both files with !. In the example, diff shows the default three lines of context around each changed line. One line was changed in both files (marked with !), one line was added to price1 (marked with +), and one line was removed from price1 (marked with –).

Note: If there are no marks to be shown in the corresponding lines of the file being compared, the lines are not displayed. Lines 11 to 13 of price2 are suppressed for this reason.

Localization
diff uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_TIME
- LC_SYNTAX
- NLSPATH


Environment variables
diff uses the following environment variable:

_TEXT_CONV
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.
Exit values

0
No differences between the files compared.

1
diff compared the files and found them to be different.

2
Failure due to any of the following reasons:
• The code set is not valid.
• Could not turn off automatic conversion.
• Could not perform requested text conversion.
• Incorrect command-line argument.
• Inability to find one of the input files.
• Out of memory.
• Read error on one of the input files.

4
At least one of the files is a binary file containing embedded NUL (\0) bytes or newlines that are more than LINE_MAX bytes apart.

Messages
Possible error messages include:

file filename: no such file or directory
The specified filename does not exist. filename was either typed explicitly, or generated by diff from the directory of one file argument and the basename of the other.

Files file1 and file2 are identical
The -s option was specified and the two named files are identical.

Common subdirectories: name and name
This message appears when diff is comparing the contents of directories, but you have not specified –r. When diff discovers two subdirectories with the same name, it reports that the directories exist, but it does not try to compare the contents of the two directories.

Insufficient memory (try diff –h)
 diff ran out of memory for generating the data structures used in the file differencing algorithm. (See “Limits” on page 262.) The -h option of diff can handle any size file without running out of memory.

Internal error—cannot create temporary file
 diff was unable to create a working file that it needed. Ensure that you either have a directory /tmp or that the environment contains the TMPDIR environment variable that names a directory where diff can store temporary files. Also, ensure that there is sufficient file space in this directory.

Missing ifdef symbol after -D
You did not specify a conditional label on the command line after the -D option.

Only one file may be –
Of the two input files typically found on the command line of diff, only one can be the standard input (stdin).

Too many lines in filename
A file of more than the maximum number of lines (see “Limits” on page 262) was given to diff.

Limits
The longest input line is 1024 bytes. Except under –h, files are limited to INT_MAX lines. INT_MAX is defined in limits.h.
Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The -B, -D, -H, -h, -i, -m, -s, -W, and -w options, and the n argument to the -c option, are extensions of the POSIX standard.

Related information
cmp, comm, patch

dircmp - Compare directories

Format
dircmp [-Bds] [-W option[option]] ... dir1 dir2

Guideline: The dircmp utility is fully supported for compatibility with older UNIX systems. However, use diff –r instead because it might provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

Description
dircmp examines dir1 and dir2 and generates listings about the contents of the directories. Listings of files that are unique to each directory are generated for all the options. If no option is entered, a list is output indicating whether the file names that are common to both directories have the same contents.

Options
- B
  Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.
- d
  Compares the contents of files with the same name in both directories and creates a list telling what must be changed in the two files to bring them into agreement. The list format is described in diff.
- s
  Suppress messages about identical files.
- W option[option]...
  Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

  filecodeset=codeset
  Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

  If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

  If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.
When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User's Guide and Reference.

pgmcodeset=codeset
Perform text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). The command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

Restriction: The only supported values for pgmcodeset are IBM-1047 and 1047.

Examples

1. To compare the contents of two directories, showing files that are identical, files that differ, and files or directories that are unique to a directory:

   `dircmp MyDir01 MyDir02`

2. To compare the contents of two directories and only show files that differ, along with a listing of those differences, and files or directories that are unique to a directory:

   `dircmp -ds MyDir01 MyDir02`

3. To compare the contents of two directories that consists of text files containing ASCII characters, showing files that are identical, files that differ, along with a listing of those differences, and files or directories that are unique to a directory, assuming that:

   • The text files are untagged and you do not want to tag them or enable automatic conversion, and
   • You cannot alter the tag (for example, you are comparing untagged public text files or read-only text files):

   `dircmp -d -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 MyAsciiDir01 MyAsciiDir02`

4. To compare the contents of two directories consisting of text files containing EBCDIC characters and only show files that differ, along with a listing of those differences, and files or directories that are unique to a directory, assuming that automatic conversion has been enabled but the text files are incorrectly tagged as UTF-8:

   `dircmp -Bds MyMisTaggedDir01 MyMisTaggedDir02`

Localization

dircmp uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH

Environment variables

dircmp uses the following environment variable:

_TEXT_CONV
  Contains text conversion information for the command. The text conversion information is not used
  when either the -B option or the filecodeset or pgmcodeset option (-w option) is specified. For more
  information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX
  shell commands,” on page 993.

Exit values

0
  Successful completion.

>0
  • The code set is not valid.
  • Could not turn off automatic conversion.
  • Could not perform requested text conversion.

Related information

cmp, diff

dirname - Return the directory components of a path name

Format

dirname pathname

Description

dirname deletes the trailing part of a file name. The result is the path name of the directory that contains
the file. This is useful in shell scripts. dirname does not try to validate the path name. For validation, use
pathchk.

dirname follows these rules:
1. If pathname is //, return it.
2. Otherwise, if it is all slashes, return one slash.
3. Otherwise, remove all trailing slashes.
4. If there are no slashes remaining in pathname, return period (.).
5. Otherwise, remove trailing nonslash characters.
6. If the remaining string is //, return it.
7. Otherwise, remove any trailing slashes.
8. If the resulting string is empty, return period (.).
9. Otherwise, return the resulting string.

Examples

The command:

dirname src/lib/printf.c

produces:

src/lib
Localization
dirname uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values

0  Successful completion.
1  Failed
2  Unknown command-line option.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

Related information

basename, pathchk

. (dot) - Run a shell file in the current environment

Format

. file [argument ...]

Description

. (dot) runs a shell script in the current environment and then returns. Normally, the shell runs a
command file in a child shell so that changes to the environment by such commands as cd, set, and
trap are local to the command file. The . (dot) command circumvents this feature.

If there are slashes in the file name, . (dot) looks for the named file. If there are no slashes . (dot)
searches for file in the directories specified in the PATH variable. This may surprise some people when
they use dot to run a file in the working directory, but their search rules are not set up to look at the
working directory. As a result, the shell does not find the shell file. If you have this problem, you can use:

. ./file

This indicates that the shell file you want to run is in the working directory. Also, the file need not be
executable, even if it is looked for on the PATH. If you specify an argument list argument ..., . (dot) sets
the positional parameters to the arguments while running the shell script, then restores the positional
parameters of the invokers. If no argument list is specified, the shell script has the same positional
parameters as the invoker. Any changes made to the positional parameters (for example, by the set
command) in the shell script remain in effect when the . (dot) command ends.

Usage notes

1. . (dot) is a special built-in shell command.
2. The file specified is treated as a shell script containing shell commands. Files that are not shell scripts (such as REXX execs, executable programs) should not be specified as file.

Localization

(dot) uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH


Exit values

1
The path search failed.

2
Failure because of an incorrect command-line option.

Otherwise, the exit status is the exit status of the last command run from the script.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

Related information

cd, set, sh, trap

dspcat - Display all or part of a message catalog

Format

dspcat [-gt] CatalogName [SetNumber [MessageNumber]]

Description

dspcat displays a particular message, all of the messages in a set, or all of the messages in a catalog. Messages are displayed as they are specified in the message catalog; no substitution of variables takes place.

It directs the messages to standard output (stdout).

It has the following parameters:

• The CatalogName parameter specifies a message catalog.
• The SetNumber parameter specifies a set in the catalog specified by the CatalogName parameter. If you specify a nonexistent SetNumber value, all messages in the catalog are displayed.
• The MessageNumber parameter specifies a particular message in the set specified by the SetNumber parameter.

If you include all three parameters, dspcat displays a particular message. If you do not include the MessageNumber parameter, or if the MessageNumber value is in error, all the messages in the set are displayed. If you specify only the CatalogName parameter, all the messages in the catalog are displayed. You must include the SetNumber parameter if you include the MessageNumber parameter.

Use the NLSPATH environment variable to find the specified message catalog if slash (/) characters are not used in the value of the CatalogName parameter.
Options

-g
  Formats the output so it can be used as input to the gencat command. The MessageNumber parameter is not valid when -g is specified.

-t
  Displays the timestamp of the message catalog.

Examples
To display message number 2 in set number 1 of test.cat, enter:

dspcat test.cat 1 2

dspmsg - Display selected messages from message catalogs

Format

dspmsg [-d][-s SetNumber] CatalogName MessageNumber['DefaultMessage'][Arguments]

Description

dspmsg displays either the text of a particular message from a message catalog that is generated with the gencat command or, if the message cannot be retrieved, a default message supplied as a parameter to the command. dspmsg directs the message to standard output. This command is intended for use in shell scripts as a replacement for the echo command.

The NLSPATH environment variable and the LANG category are used to find the specified message catalog if / (slash) characters are not used in the value of the CatalogName parameter. If the catalog named by the CatalogName parameter is not found or if the message named by the MessageNumber parameter (and optional SetNumber value) is not found, then the supplied DefaultMessage value is displayed. If a DefaultMessage value is not specified, a system-generated error message is displayed.

dspmsg allows up to ten string arguments to be substituted into the message if it contains the %s or %n $s, fprintf() conversion specification. Only string variables are allowed. If arguments are specified, then a DefaultMessage must also be specified.

Missing arguments for conversion specifications result in a dspmsg error message. Normal fprintf() subroutine control character escape codes (for example, –n) are recognized.

Options

-d
  If you are receiving the default message, use this option to request debugging information about why dspmsg cannot get the message from the message catalog.

-s SetNumber
  Specifies an optional set number. The default value for the SetNumber variable is 1.

Examples
To display set number 1, message number 2 of the test.cat catalog, enter:

dspmsg -s 1 test.cat 2 'message %s not found' 2

If the message is not found, message 2 not found is displayed.
du - Summarize usage of file space

Format

`du [-a|-s[-krtx] [pathname ...]`

Description

`du` reports the amount of file space that is used by the files indicated by the given path name. If the path name is a directory, `du` reports the total amount of file space that is used by all files in that directory and in each subdirectory in its hierarchy. If you do not specify a path name, `du` assumes the current directory. Files with multiple links are only counted once. On systems supporting symbolic links, only the disk space that is used by the symbolic link is counted.

`du` measures file space in 512-byte units.

Options

- **-a**
  Generates a report for all files in `pathname`.

- **-k**
  Displays file sizes in 1024-byte (1 KB) units.

- **-r**
  Reports files that cannot be opened and directories that cannot be read; this is the default.

- **-s**
  Does not display file size totals for subdirectories.

- **-t**
  Displays the total amount of space used by all path names examined.

- **-x**
  Displays file sizes for only those files that are contained on the same device as `pathname`.

Usage notes

`du` computes file space in units of 512 bytes. The actual disk space that is used by files and directories might be more, since some systems allocate space in units of some multiple of a sector. On UNIX System V, it is usually two sectors; on UNIX Version 7, it is one sector.

The allocation unit is file system specific.

Localization

`du` uses the following localization variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


Exit values

0
  Successful completion
1

Failure due to any of the following reasons:

• Incorrect command-line option.
• Cannot access a directory.
• Cannot read a directory.
• Cannot access file information.

**Portability**

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
The -t option is an extension to the POSIX standard.

**Related information**

df, find, ls

---

**echo - Write arguments to standard output**

**Format**

```
echo argument ...
tcsh shell: echo [-n] word ...
```

**Description**

`echo` writes its arguments, which are specified with the `argument` argument, to standard output. `echo` accepts these C-style escape sequences:

- \a Bell
- \b Backspace
- \c Removes any following characters, including \n and \r.
- \f Form feed
- \n Newline
- \r Carriage return
- \t Horizontal tab
- \v Vertical tab
- \0num The byte with the numeric value that is specified by the zero to three-digit octal `num`.
- \- Backslash

`echo` follows the final argument with a newline unless it finds \c in the arguments. Arguments are subject to standard argument manipulation.
In the tcsh shell, echo writes each word to the shell's standard output, which is separated by spaces and terminated with a newline.

tcsh echo accepts these C-style escape sequences:

\a
   Bell
\b
   Backspace
\e
   Escape
\f
   Form feed
\n
   Newline
\r
   Carriage return
\t
   Horizontal tab
\v
   Vertical tab
\nnnn
   The EBCDIC character corresponding to the octal number nnn

For more information, see “tcsh: Invoke a C shell” on page 671.

Examples

1. One important use of echo is to expand file names on the command line, as in:

```bash
echo *.[ch]
```

   This displays the names of all files with names ending in .c or .h, typically C source and include (header) files. echo displays the names on a single line. If there are no file names in the working directory that end in .c or .h, echo displays the string *.[ch].

2. echo is also convenient for passing small amounts of input to a filter or a file:

```bash
echo 'this is\nreal handy' > testfile
```

Usage notes

echo is a built-in shell command.

Localization

echo uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH

Exit value

echo always returns the following exit status value:

0
  Successful completion.

Portability

POSIX.2, UNIX System V.

The POSIX standard does not include escape sequences, so a strictly conforming application cannot use them. printf is suggested as a replacement.

Related information

sh, tcsh

ed - Use the ed line-oriented text editor

Format

ed [-Bbs] [-p prompt] [-W option[option] ... [file]

Description

ed is a line-oriented text editor that lets you manipulate text files interactively. It reads the text of a file into memory and stores it in an area called a buffer. You can edit the text in the buffer by using its subcommands. You can also write the contents of the buffer back out to the file, thereby overwriting the old contents of the file.

Options

-B
  Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-b
  Enables you to edit larger files by restricting the amount of memory dedicated to paging. Using a large amount of memory may make ed run more slowly.

-p prompt
  Displays the given prompt string prompting you to input a subcommand. By default, ed does not typically prompt for subcommand input. See “Subcommands” on page 274 for more information about subcommand prompting.

-s
  Puts ed into a quiet mode, in which e, E, r, and w, subcommands do not display file size counts; the q and e subcommands do not check buffer modification; and ! is not displayed after calling the shell to run a subcommand. This mode is particularly useful when you invoke ed from within a shell script.

-W option[,option]...
  Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

    filecodeset=odeset
      Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

      If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor
pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

pgmcodeset=codeset

Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command icnv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

If the optional file argument is present on the command line, ed reads the specified file into the editor by simulating an efile subcommand.

Addresses

You can prefix subcommands in ed with zero, one, or two addresses. These addresses let you refer to single lines or ranges of lines in the buffer. You do not need to specify addresses for certain subcommands that use default addresses. Consult the description for a particular subcommand. You can construct each address out of the following components:

.  The single dot character represents the current line in the buffer. Many subcommands set the current line; for example the e command sets it to the last line of the new file being edited.

$  The dollar sign refers to the last line in the buffer.

n  The number n refers to the nth line in the buffer.

/regexp/

This searches for a line containing a string that matches the regular expression, regexp. The search begins at the line immediately following the current line. It proceeds forward through the buffer; if ed reaches the end of the buffer without finding a match, it wraps around to the first line of the buffer and continues the search. If ed does not find a match, the search ends when it reaches the original current line. If it does find a match, the address /regexp/ refers to the first matching line. If you omit regexp, the last used regular expression becomes the object of the search. You can omit the trailing /. Within regexp, \ represents a literal slash and not the regexp delimiter.

?regexp?

This is similar to the previous address form, except that the search goes backward through the buffer. If the search reaches the first line in the buffer without finding a match, ed wraps around and continues searching backward from the last line in the buffer. If you omit regexp, the last used regular
expression becomes the object of the search. You can omit the trailing ?. Within regexp, |? represents a literal question mark and not the regexp delimiter.

`l`

The address is the line that is marked with the mark name 1. The name 1 must be a lowercase letter that is set by the k subcommand.

You can combine these basic addresses with numbers using the + and – operators, with the usual interpretation. Missing left operands default to . (dot); missing right operands default to 1. Missing right operands also have a cumulative effect; so an address of – – refers to the current line number less two.

You can specify address ranges in the following ways:

- **a1,a2**
  Specifies a range of addresses from address a1 to address a2, inclusive. If you omit a1 and a2 (that is, the comma alone is specified), this is equivalent to the range 1, $.

- **a1; a2**
  Is similar to the previous form except that ed resets the current line after calculating the first address, a1, so that the second address, a2, is relative to a1. If you omit a1 and a2 (that is, the semicolon alone is specified), this is equivalent to . ; $. If you specify only a1 and the command requires both a1 and a2, the command operates as though you specified a range of:

  ```
  a1; . command
  ```

- **>**
  Is equivalent to . , . +22 (that is, page forward), except that it never attempts to address any line beyond $.

- **<**
  Is equivalent to . –22 , . (that is, page backward), except that it never addresses any line before line 1.

**Subcommands**

An ed command has the form [address] command

All commands end with a newline; you must press <Enter>. Most commands allow only one command on a line, although you can modify commands by appending the 1, n, and p commands.

Subcommands generally take a maximum of zero, one, or two addresses, depending upon the particular subcommand. In the following descriptions, we show commands with their default addresses (that is the addresses used when you don’t specify any addresses) in a form that shows the maximum number of permitted addresses for the command. In any of the subcommands that take a file argument, file can be a path name or:

- `!command-line`
  If you use the ! form, ed runs the given command line, reading its standard output (stdout) or writing its standard input (stdin), depending on whether the ed command does reading or writing.

If a terminal disconnect is detected:

- If the buffer is not empty and has changed since the last write, the ed utility will attempt to write a copy of the buffer to a file named ed . hup in the current directory. If this write fails, ed will attempt to write a copy of the buffer to a file name ed . hup in the directory that is named by the HOME environment variable. If both these attempts fail, ed will exit without saving the buffer.

- The ed utility will not write the file to the currently remembered path name or return to command mode, and will terminate with the exit status of 1.

If an end-of-file is detected on standard input:

- If the ed utility is in input mode, ed will terminate input mode and return to command mode. Any partially entered lines (that is, input text without a terminating newline) will be saved.
• If the ed utility is in command mode, it will act as if a q command had been entered.

ed accepts the following subcommands:

.a
Appends text after the specified line. Valid addresses range from 0 (text is placed at the beginning of the buffer, before the first line) to $ (text is placed after the last line of the buffer). ed reads lines of text from the workstation until a line consisting solely of an unescaped . (dot) is entered. ed sets the current-line indicator to the last line appended.

.. .c
Changes the addressed range of lines by deleting the lines and then reading new text in the manner of the a or i subcommands. If the variable _UNIX03 is set to YES, address 0 is valid for this subcommand and it will be interpreted as if address 1 were specified.

.. .d
Deletes the addressed range of lines. The line after the last line deleted becomes the new current line. If you delete the last line of the buffer, ed sets the current line to the new last line. If no lines remain in the buffer, it sets the current line to 0.

E[file]
Is similar to the e command, but ed gives no warning if you have changed the buffer.

e [file]
Replaces the contents of the current buffer with the contents of file. The text conversion that is specified on the ed command (for example, the -B or -W option) is used. If you did not specify file, ed uses the remembered file name, if any. In all cases, the e subcommand sets the remembered file name to the file that it has just read into the buffer. ed displays a count of the bytes in the file unless it is in quiet mode. If you have changed the current buffer since the last time its contents were written, ed warns you if you try to run an e subcommand, and does not run the subcommand. If you enter the e subcommand a second time, ed goes ahead and runs the command.

f [file]
Changes the remembered file name to file. ed displays the new remembered file name. If you do not specify file, ed displays the current remembered file name.

1,$G/regexp/
Is similar to the g command except that when ed finds a line that matches regexp, it prints the line and waits for you to type in the subcommand to be run. You cannot use the a, c, i, g, G, v, and V subcommands. If you enter a, the G subcommand reruns the last subcommand you typed in. If you just press <Enter>, G does not run any subcommand for that line. Note that the subcommands input as part of the execution of the G subcommand can address and affect any lines in the buffer. If the variable _UNIX03 is set to YES, any line modified by the subcommand will be unmarked.

1,$g/regexp/command
Performs command on all lines that contain strings matching the regular expression regexp. This subcommand works in two passes. In the first pass, ed searches the given range of lines and marks all those that contain strings matching the regular expression regexp. The second pass performs command on those lines. If the variable _UNIX03 is set to YES, any line modified by the command will be unmarked. You cannot use !, g, G, V, or v as command. command consists of one or more ed subcommands, the first of which must appear on the same line as the g subcommand. All lines of a multiline command list, except the last, must end with a backslash (\). If command is empty, ed assumes it to be the p subcommand. If no lines match regexp, ed does not change the current line number; otherwise, the current line number is the one set by the last subcommand in command. Instead of the slash (/) to delimit regexp, you can use any character other than space or newline.

H
Tells ed to display more descriptive messages when errors occur. If ed is already printing descriptive messages, H returns to terse error messages. Normally, ed indicates error messages by displaying a ?. When you turn on descriptive error messages with this subcommand, ed also displays the descriptive message for the most recent ? message.

h
Provides a brief explanation of the last error that occurred. This does not change the current line number.
Works similarly to the a subcommand, except that ed places the text before the addressed line. Valid addresses range from line 1 to $ (the last line). ed sets the current line number to the last inserted line. If the variable _UNIX03 is set to YES, address 0 is valid for this subcommand and it will be interpreted as if address 1 were specified.

Joins a range of lines into one line. To be precise, the j command removes all newline characters from the addressed range of lines, except for the last one. ed sets the current line number to the resulting combined line.

Marks the addressed line with the mark name x, which is any single lowercase letter of the alphabet. This lets you refer to a marked line with the construct 'x. This is called an absolute address, because it always refers to the same line, regardless of changes to the buffer.

Displays the addressed range of lines, representing nonprintable (control) characters in a visible manner. The end of each line will be marked with a '$' character. The characters listed in the Base Definitions volume of IEEE Std 1003.1-2001, Table 5-1, Escape Sequences and Associated Actions (‘\", \a, \b, \f, \r, \t, \v’) shall be written as the corresponding escape sequence; the '\n' in that table is not applicable. If the variable _UNIX03 is set to YES, '$' characters within the text will be written with a preceding backslash. ed sets the current line to the last line so displayed. You can append this subcommand to most other commands, to check on the effect of those subcommands.

Moves the addressed lines to the point immediately following the line given by the address a. The address a must not be in the range of addressed lines. If address a is 0, ed moves the lines to the beginning of the buffer. The last line moved becomes the new current line.

Displays the addressed lines in a way similar to the p command, but ed puts the line number and a tab character at the beginning of each line. The last line displayed becomes the new current line. You can append n to any subcommand (except for E, e, f, Q, r, w, or !) so that you can check on the effect that the subcommands had.

Turns on subcommand prompting if it is not already on. If you specified the –p prompt option on the ed command line, ed displays the prompt string whenever it is ready for you to type in another subcommand. If you did not include the –p option, ed uses the * character as a prompt. If subcommand prompting is currently turned on, issuing the P subcommand turns it off.

Displays (prints) the addressed lines. The last line displayed becomes the new current line. You can append p to most subcommands, so that you can check on the effect that the subcommands had.

You can append p to any subcommand (except for E, e, f, Q, r, w, or !) so that you can check on the effect that the subcommands had.

Quits unconditionally, without checking for buffer changes.

Causes the editor to exit. If you have made changes to the buffer since the last save and you try to quit, ed issues a warning. Entering the q subcommand again lets you quit, regardless of unsaved changes.

Reads the contents of the file into the buffer after the addressed line. The text conversion that is specified on the ed command (for example, the -B or -W option) is used. If the address is 0, ed places the text before the first line in the buffer. If you do not specify file, ed uses the remembered file name; if no remembered file name exists, file becomes the new remembered name. If file contains bytes that are not valid in the current character set, they are replaced by the rubout character.
The `r` subcommand displays the number of bytes read from file unless you specified the –s option. The last line read from the file becomes the new current line. If file is replaced by `, the rest of the line is considered a shell command line, the output of which is to be read.

```
.,.s/regexp/new/[flags]
```

Searches the specified range of lines for strings matching the regular expression `regexp`. Normally the `s` subcommand replaces the first such matching string in each line with the string `new`. The `s` subcommand sets the current line to the last line on which a substitution occurred. If `ed` makes no such replacements, `ed` considers it an error.

`flags` can be one of the following:

- **n**
  - Replaces the `n`th matching string in the line instead of the first one.

- **g**
  - Replaces every matching string in each line, not just the first one.

- **l**
  - Displays the new current line in the format of the `l` subcommand.

- **n**
  - Displays the new current line in the format of the `n` subcommand.

- **p**
  - Displays the new current line in the format of the `p` subcommand.

You can use any single printable character other than space or newline instead of `/` to separate parts of the subcommand provided that you use the same character to delimit all parts of the subcommand. You can omit the trailing delimiter.

You can include a newline in the `new` string by putting a `\` immediately in front of the newline. This is a good way to split a line into two lines. If `new` consists only of the `%` character, `s` uses the `new` string from the previous `s` command. If the variable `_UNIX03=YES` is set and there was no previous `s` command, the use of `%` in this manner is an error. If `&` appears anywhere in `new`, `ed` replaces it with the text matching the `regexp`. If you want `new` to contain a literal ampersand, or percent sign, put a backslash (`\`) in front of the `&` or `%` character.

```
.,.ta
```

Copies the addressed lines to the point after the line given by the address `a`. The address `a` must not fall in the range of addressed lines. If address `a` is 0, `ed` copies the lines to the beginning of the buffer. This sets the current line to the last line copied.

```
u
```

Rolls back the effect of the last subcommand that changed the buffer. For the purposes of `u`, subcommands that change the buffer are: `a`, `c`, `d`, `G`, `g`, `i`, `j`, `m`, `r`, `s`, `t`, `v`, `V`, and (of course) `u`. This means that typing `u` repeatedly switches the most recent change back and forth. This subcommand sets the current line number to the value it had immediately before the subcommand being undone started.

```
1,$V/regexp/
```

Is similar to the `G` subcommand, except that this subcommand gives you the chance to edit only those lines that do not match the given regular expression.

```
1,$v/regexp/commands
```

Is similar to the `g` (global) command, except that `ed` applies the given `commands` only to lines that do not match the given regular expression.

```
1,$w [file]
```

Is similar to the `w` subcommand, except that this command appends data to the given `file` if the file already exists.

```
1,$w [file]
```

Writes the addressed lines of the buffer to the named `file`. The text conversion that is specified on the `ed` command (for example, the `-B` or `-W` option) is used. This does not change the current line. If you do not provide `file`, `ed` uses the remembered file name; if there is no remembered file name, `file`
becomes the remembered name. If the output file does not exist, ed creates it. ed displays the number of characters written unless you had specified the –s option.

**X**
Prompts you to enter an encryption key. All subsequent e, i, and w subcommands use this key to decrypt or encrypt text read from or written to files. To turn encryption off, issue an X subcommand and press <Return> in response to the prompt for an encryption key.

!**command**
Runs **command** as if you typed it to your chosen command interpreter. If **command** contains the % character, ed replaces it with the current remembered file name. If you want a subcommand to contain a literal %, put a backslash (\) in front of the character. As a special case, typing !!! reruns the previous **command**.

**$**=
Displays the line number of the addressed line. This does not change the current line.

**.**=
Displays the current line number.

**.+1,**.+1
If you supply zero, one, or two addresses without an explicit subcommand, ed displays the addressed lines in the mode of the last print subcommand: p, l, or n. This sets the current line number to the last line displayed.

**Examples**
1. To edit or browse a file using the ed line-oriented text editor:
   
ed myFile

2. To edit or browse a file containing UTF-8 characters using the ed line-oriented text editor, assuming that
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are browsing an untagged public text file or a read-only text file)

   then issue:
   
ed -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File

3. To edit or browse a file containing EBCDIC characters using the ed line-oriented text editor, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as ASCII:
   
ed -B myMisTaggedFile

**Environment variables**
ed uses the following environment variables:

**COLUMNS**
Contains the terminal width in columns. ed folds lines at that point. If it is not set, ed uses the appropriate value from the terminfo database or if that is not available, it uses a default of 80.

**HOME**
Contains the path name of your home directory.

**SHELL**
Contains the full path name of the current shell.

**TMPDIR**
The path name of the directory being used for temporary files. If it is not set, ed uses /tmp.
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

For more information about the effect of _UNIX03 on this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Files
ed uses the following files:

/tmp/e*
This is the paging file. It holds a copy of the file being edited. You can change the directory for temporary files using the environment variable TMPDIR.

ed.hup
When ed receives a hang up signal (or detects a terminal disconnect) and the current buffer has changed since the last write, ed will attempt to write the current buffer to ed.hup in the current directory. If this write fails, ed will attempt to write the current buffer to ed.hup in the $HOME directory.

Localization
ed uses the following localization environment variables:

• LANG
• LC_ALL
• LC_COLLATE
• LC_CTYPE
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH

Exit values
0
Successful completion
1
Failure due to any of the following reasons:

• Addressed line out of range
• Only one file name is allowed
• No space for the line table
• Temporary file error
• Badly constructed regular expression
• No remembered regular expression
• File read error
• Out of memory
• Unknown command
• Command suffix not permitted
• No match found for regular expression
• Wrong number of addresses for the subcommand
• Not enough space after the subcommand
• The name is too long
• Badly formed name
• Subcommand redirection is not permitted
• Restricted shell
• No remembered file name
• The mark name must be lowercase
• The mark name is not defined
• m and t subcommands require a destination address
• The destination cannot straddle source in m and t
• A subcommand not allowed inside g, v, G, or V
• The x subcommand has become X (uppercase)
• The global command is too long
• Write error (no disk space)
• The code set is not valid
• Could not turn off automatic conversion
• Could not perform requested text conversion

2
Usage error

Messages
Some error messages are issued only if h or H subcommands are used after ed displays ?. Possible error messages include:

Destination cannot straddle source in m and t
The range of lines being moved or copied by m or t cannot include the destination address.

Global command too long
There is a limit on the length of a global instruction (g or v). See “Limits” on page 281 for this limit.

m and t require destination address
You must follow the m or t subcommands with an address indicating where you want to move or copy text. You omitted this address.

No remembered file name
You tried to run a subcommand that used a remembered file name (for example, you used w to write without specifying an output file name). However, there is no remembered file name at present. Run the subcommand again, but specify a file name this time.

Restricted shell
The command line invoked the restricted form of ed, but you tried an action that was not allowed in the restricted editor (the ! subcommand).

Temporary file error
You ran out of space on disk or encountered other errors involving the page file stored in the temporary file.

Warning: file not saved
You entered a subcommand to quit editing the current file, for example, q or e to edit a new file; however, you have changed the file since the last time you saved it. ed is suggesting that you save the file before you exit it; otherwise, your recent changes will be lost. To save the file, use the w command. If you really do not want to save the recent changes, use q to quit or e to edit a new file.
An error occurred during an attempt to open or create file. This is applicable to the e, r, and w subcommands.

? An unspecified error occurred. Use the h or H subcommand for more information. If the input to ed comes from a script rather than from a workstation, ed exits when any error occurs.

Limits
ed allows a limit of 1024 bytes per line and 28 000 lines per file. It does not allow the NUL (\0') character. The maximum length of a global command is 256 characters, including newlines.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The addresses < and >, the −B, −b, and −W options, and the W and X subcommands are extensions of the POSIX standard.

Related information
awk, diff, env, ex, grep, sed, vi
See Appendix C, “Regular expressions (regexp),” on page 945 for more information about regexp.

edcmtext: Display errnojr reason code text

Format
edcmtext errnojr_value

Description
edcmtext displays the description and action text for C/C++ run-time library errnojr (errno2) values. No other values are supported by this command. This command is intended as an aid for problem determination.

errnojr_value is specified as 8 hexadecimal characters.
You can specify one of the following in place of a errnojr value to view a help dialog: -h, help, ?.
To display the output in uppercase, specify the -U option.

Usage notes
errnojr_values are also accepted in mixed case and with hexadecimal digits prefixed with the "0x".
Message returns
If the user specifies a -h, help or ? in place of the errnojr_value, the following message is displayed:

Usage: edcmtext errnojr_value

If no text is available for the errnojr_value the following message is displayed:

ernojr_value: No information is currently available for this errnojr_value.

If the errnojr_value is not comprised of 1-8 hexadecimal digits the following message is displayed:

Usage: edcmtext errnojr_value

If the errnojr_value is not in the C/C++ run-time library range, the following message is displayed:

Notice: The errnojr_value is not in the C/C++ run-time library range.

If the environment that edcmtext is being run in is not TSO/E or z/OS UNIX, the following message is displayed:

Error: The environment is not TSO/E or z/OS UNIX.

ernojr (errno2) values will be found in z/OS Language Environment Runtime Messages.

Examples
The command:

edcmtext C00B0021

produces data displayed in the following format:

JrEdc1opsEinval01: The mode argument passed to fopen() or freopen() did not begin with r, w, or a.
Action: Correct the mode argument. The first keyword of the mode argument must be the open mode. Ensure the open mode is specified first and begins with r, w, or a.
Source: edc1opst.c

Exit values
0
Successful completion
2
Failure due to an argument that is not 1–8 hexadecimal digits
8
Bad input due to an errnojr_value out of the C/C++ run-time range.
14
Environment not TSO/E or z/OS UNIX
>20
Internal error. Contact IBM.
egrep - Search a file for a specified pattern

Format

[-W option[option] ... [-e pattern] file ... [-f patternfile] file ...]

The egrep utility is fully supported for compatibility with older UNIX systems. However, use grep -E instead because it might provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

Description

See grep.

eov - Display or set environment variables for a process

Format

env [-i] [variable=value ...] [command argument ...]
env [-] [variable=value ...] [command argument ...]

Description

If you enter env with no arguments, it displays the environment variable that it received from its parent (presumably the shell).

Arguments of the form variable=value let you add new environment variables or change the value of existing environment variables.

If you specify command, env calls command with the arguments that are specified with the argument argument that appear on the command line, passing the accumulated environment variable to this command. The command is run directly as a program found in the search path, and is not interpreted by a shell.

In a double-byte locale, environment variable values can contain double-byte characters. The equal sign (=) must be single byte.

Options

env supports the following two options, both of which have the same effect.

- i
  Specifies that the environment variable inherited by env not be used.

- 
  Specifies that the environment variable inherited by env not be used.

Examples

1. Compare the output of the following two examples:

   env foo=bar env
   env -i foo=bar env

2. Compare the output of the following example:

   env - echo $PATH
   .:/usr/lpp/Printsrv/bin:/bin:/usr/sbin
The variable $PATH appears to still be valid but is resolved first by the shell before the initial `env` command is run to clear the environment variables with the - or -i option. For more information, see the section on using substitutions in commands in z/OS UNIX System Services User’s Guide.

**Localization**

`env` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Exit values**

0  
Successful completion.

1  
Failure due to any of the following reasons:
- Not enough memory.
- Name is too long.

2  
Incorrect command-line argument.

126  
`env` found command but could not invoke it.

127  
`env` could not find command.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX System V.

`printenv` on Berkeley UNIX systems works like `env`.

**Related information**

`sh`

---

**eval - Construct a command by concatenating arguments**

**Format**

eval [argument ...]

tcsh shell: eval argument ...

**Description**

The shell evaluates each argument as it would for any command. `eval` then concatenates the resulting strings, separated by spaces, and evaluates and executes this string in the current shell environment.

In the tcsh shell, `eval` treats the arguments as input to the shell and executes the resulting commands in the context of the current shell. This action is typically used to execute commands generated as the result
of command or variable substitution, since parsing occurs before these substitutions. See “tcsh: Invoke a C shell” on page 671.

**Examples**

The command:

```bash
for a in 1 2 3
do    eval x$a=fred
done
```

sets variables x1, x2, and x3 to fred. Then:

```bash
echo $x1 $x2 $x3
```

produces:

```
fred fred fred
```

**Usage notes**

eval is a special built-in shell command.

**Localization**

eval uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Exit values**

The only possible exit status value is:

0

No arguments were specified, or the specified arguments were empty strings. Otherwise, the exit status of eval is the exit status of the command that eval runs.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

**Related information**

eexec, sh, tcsh

---

**ex - Use the ex text editor**

**Format**

```
ex [-BelRsrv] [-c command] [-t tag] [-w size] [-W option[option] ... [file ... file ...]
```

**Description**

`ex` is the line-editor mode of the `vi` text editor.
Options

The **ex** internal commands are described in **vi**. It supports the following options:

**+command**

Begins the editing session by running the specified editor **command**. To specify multiple commands, separate them with a vertical bar (|).

**-B**

Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

**-c command**

Begins editing by executing the specified editor command. To specify multiple commands, separate them with an “or” bar (|). **command** can be any **ex** command except those that enter input mode, such as insert or append.

**-e**

Invokes **ex**. This option is intended for use with **vi**.

**-l**

Sets LISP mode. The ( and ) commands use blocks of LISP code as their context rather than sentences.

**-r**

Recovers named files after an editor or system fails. If you do not specify a file argument, **ex** lists all recoverable files and then exits.

When using **ex -r** to recover a file that was being edited with automatic conversion, the file must also be recovered with automatic conversion enabled when writing the data back to the original tagged text file. Likewise, if explicit conversion was being used when editing the file (by using the -W filecodeset or -W pgmcodeset options), the same options must be specified when writing the recovered data back to the original file. Failure to do either of these might result in incorrectly coded character data being written to the file when you save the recovered version.

**-R**

Sets read-only mode.

**-s**

Suppresses all interactive feedback (quiet mode). This option is for batch mode operation; **ex** assumes that the terminal cannot display text and ignores the value of TERM. **ex** also ignores all startup files and ignores the value of EXINIT.

**-t tag**

Edits the file containing the specified **tag** and sets the virtual position in the edit buffer to point of definition for the tag.

**-v**

Invokes **vi**.

**-w size**

Sets the option variable **window** equal to **size**.

**-W option[,option]...**

Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

**filecodeset=codeset**

Performs text conversion from one code set to another when reading from the file. The coded character set of the file is **codeset**. **codeset** can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command **iconv -l** lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text
conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

**pgmcodeset=** *codeset*

Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is *codeset*. *codeset* can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

**Examples**

1. To edit or browse a file containing UTF-8 characters using the **ex** editor, assuming that:
   - The file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are browsing an untagged public file or a read-only file)
   issue:

   ```
   ex -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File
   ```

2. To edit or browse a file containing EBCDIC characters using the **ex** editor, assuming that automatic conversion has been enabled but the file is incorrectly tagged as ASCII, issue:

   ```
   ex -B myMisTaggedFile
   ```

**Localization**

**ex** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Environment variables**

**ex** uses the following environment variable:
EXEC

Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, "Controlling text conversion for z/OS UNIX shell commands," on page 993.

Exit values

0

Successful completion.

1

Failure due to any of the following reasons:

• The code set is not valid.
• Could not turn off automatic conversion.
• Could not perform requested text conversion.

2

Failure due to any of the following reasons:

• Unknown command-line option.
• Missing or incorrect num in an –n option.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
The -B, -e, -W, and -x options are extensions of the POSIX standard.

Related information

ed, vi

eexec - Run a command and open, close, or copy the file descriptors

Format

eexec [-a name] [command_line]
tcsh shell: eexec command

Description

The command_line argument for eexec specifies a command line for another command. eexec runs this command without creating a new process. Some people picture this action as overlaying the command on top of the currently running shell. Thus, when the command exits, control returns to the parent of the shell.

Input and output redirections are valid in command_line. You can change the input and output descriptors of the shell by giving only input and output redirections in the command. For example:

```
eexec 2>errors
```

redirects the standard error stream to errors in all subsequent commands run by the shell.

If you do not specify command_line, eexec returns a successful exit status.

In the tcsh shell, eexec executes the specified command in place of the current shell. See “tcsh: Invoke a C shell” on page 671.
Options
-\texttt{a name}

The shell passes name as the zeroth argument (argv[0]) to \texttt{command_line}. -\texttt{a name} can be used to replace the current shell with a new login shell, by specifying name as a shell with a prefix of a dash (-).

Examples
To replace the current shell process with a new login shell (which will run the login profiles), specify:

\begin{verbatim}
exec -a -sh /bin/sh
\end{verbatim}

Usage notes
\texttt{exec} is a special built-in shell command.

Localization
\texttt{exec} uses the following localization environment variables:

- \texttt{LANG}
- \texttt{LC_ALL}
- \texttt{LC_MESSAGES}
- \texttt{NLSPATH}

See \textit{Appendix F, “Localization,” on page 965} for more information.

Exit values
If you specify \texttt{command_line}, \texttt{exec} does not return to the shell. Instead, the shell exits with the exit status of \texttt{command_line} or one of the following exit status values:

\begin{itemize}
  \item \texttt{1-125}  
  \hspace{1em} A redirection error occurred.
  \item \texttt{126}  
  \hspace{1em} The command in \texttt{command_line} was found, but it was not an executable utility.
  \item \texttt{127}  
  \hspace{1em} The given \texttt{command_line} could not be run because the command could not be found in the current PATH environment.
\end{itemize}

If you did not specify \texttt{command_line}, \texttt{exec} returns with an exit value of zero.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Related information
\texttt{sh, tcsh}

\textbf{exit - Return to the shell's parent process or to TSO/E}

\textbf{Format}

\begin{verbatim}
exit [expression]
tcsh shell: exit [expr]
\end{verbatim}
**Description**

exit ends the shell. If there is an *expression*, the value of the *expression* is the exit status of the shell.

The value of expression should be between 0 and 255. For values outside this range, the exit status will be the least significant 8 bits of the value of the expression. The EXIT trap is raised by the exit command, unless exit is being called from inside an EXIT trap.

If you have a shell background job running, you cannot exit from the shell until it completes. However, you can switch to subcommand mode and exit.

In the tcsh shell, the shell exits either with the value of the specified expression or, without expression, with the value of the status variable. The value of expression should be between 0 and 255. See “tcsh: Invoke a C shell” on page 671.

**Usage notes**

exit is a special built-in shell command.

**Localization**

exit uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Exit values**

exit returns the value of the arithmetic expression specified by the *expression* argument to the parent process as the exit status of the shell. If you omit *expression*, exit returns the exit status of the last command run.

**Related information**

`return`, `sh`, `tcsh`

The `exit()` ANSI C function, the `_exit` callable service, and the `_exit()` POSIX C function are unrelated to the exit shell command.

---

**expand - Expand tabs to spaces**

**Format**

```
expand [-B] [-t tablist] [-W option[option] ... [file ... file ...]
```

**Description**

`expand` reads text input from the files that are specified on the command line, converts tabs into spaces, and writes the result to the standard output (stdout). If you do not specify any files on the command line, `expand` reads from the standard input (stdin).

`expand` preserves backspace characters. By default, tab stops are set every eight columns. A tab after the last tabstop is replaced by a space.
Options

The first syntax of **expand** supports the following options:

- **-B**
  Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

- **-t** *tablist*
  Sets tab stops at positions that are indicated by *tablist*. Numbers in *tablist* must be in ascending order (origin 0) and separated by commas or blanks; however, the list must be one argument so you need shell quoting if you are using blanks. The list can consist of a single number, in which case tabs are set every *tablist* positions apart.

- **-W option[,option]...**
  Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

  - **filecodeset=** *codeset*
    Performs text conversion from one code set to another when reading from the file. The coded character set of the file is *codeset*. *codeset* can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

    If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

    If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

    The only supported values for filecodeset are ISO8859-1 and 819.

  - **pgmcodeset=** *codeset*
    Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is *codeset*. *codeset* can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

    If pgmcodeset is specified but filecodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

    If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

    The only supported values for pgmcodeset are IBM-1047 and 1047.

The second syntax of **expand** (which the POSIX standard considers obsolete) supports the following options:

- **-tabstop**
  Sets tab stops every *tabstop* columns.

- **--tab1,tab2,...,tabn**
  Sets tab stops at each column *tab1,tab2* and so on (origin 0).
Examples

1. To convert tabs in a text file to spaces that are 10 positions apart:

   ```
   expand -t 10 myTextFile
   ```

2. To convert tabs in a text file containing ASCII characters to spaces, assuming that
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file)

   then issue:

   ```
   expand -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 myAsciiFile
   ```

3. To convert tabs in a text file containing EBCDIC characters to spaces, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as UTF-8:

   ```
   expand -B myTextFile
   ```

Localization

`expand` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


Environment variables

`expand` uses the following environment variable:

`_TEXT_CONV`

Contains text conversion information for the command. The text conversion information is not used when either the `-B` option or the `filecodeset` or `pgmcodeset` option (`-W` option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

Exit values

0

Successful completion.

1

Failure due to any of the following reasons:
- Cannot open the input file.
- Insufficient memory.
- Incorrect tab stop specification.
- The code set is not valid.
- Could not turn off automatic conversion.
- Could not perform requested text conversion.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, 4.2BSD and later.
The -B and -W options are extensions of the POSIX standard.

**Related information**
pr, unexpand

---

**export - Set a variable for export**

**Format**

```
export  [ name [=value] ...]
export -p
```

**Description**

`export` marks each variable `name` so that the current shell makes it automatically available to the environment of all commands run from that shell. Exported variables are thus available in the environment to all subsequent commands. Several commands (for example, `cd`, `date` and `vi`) look at environment variables for configuration or option information.

Variable assignments of the form `name=value` assign `value` to `name` as well as marking `name` for export. The `name` can contain only the underscore and alphanumeric characters from the portable character set.

Calling `export` without arguments lists, with appropriate quoting, the names and values of all variables in the format `Variable=“value”`. If you reinput this format to another shell, variables are assigned appropriately but not exported. The `-p` option lists variables in a format suitable for reinput to the shell (see the description of the `-p` option).

**Options**

- `-p`
  Lists variables in a form that is suitable for reinput to the shell:
  ```
  export name="value"
  ```

**Usage notes**

`export` is a special built-in shell command.

**Localization**

`export` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

0  
  Successful completion.

1  
  Failure due to incorrect command-line argument.
Failure, usually due to incorrect an incorrect command-line argument, that results in a usage message.

**Portability**

POSIX.2, X/Open Portability Guide.

Assigning a value to name, and the behavior given for calling `export` with arguments are extensions of the POSIX standard.

**Related information**
cd, date, set, sh, typeset, vi

---

**expr - Evaluate arguments as an expression**

**Format**

`expr -W expression`

**Description**

The set of arguments that are passed to `expr` constitutes an expression to be evaluated. Each command argument is a separate token of the expression. `expr` writes the result of the expression on the standard output. This command is primarily intended for arithmetic and string manipulation on shell variables.

`expr` supports the following operators. Operators explained together have equal precedence; otherwise, they are in increasing order of precedence. `expr` stores an expression as a string and converts it to a number during the operation. If the context requires a Boolean value, a numeric value of 0 (zero) or a null string ("") is false, and any other value is true. Numbers have an optional leading sign. If the -W option is not specified, numbers are decimal. If the -W option is specified, expressions may contain octal, hexadecimal, or decimal numbers. `expr` determines the base of the number as follows:

- Any number that starts with 0x is hexadecimal.
- Any number that starts with 0 is octal.
- Any number that does not start with 0x or 0 is decimal.

Numbers are manipulated as long integers.

- `expr1 | expr2`
  Results in the value `expr1` if `expr1` is true; otherwise, it results in the value of `expr2`.

- `expr1 & expr2`
  Results in the value of `expr1` if both expressions are true; otherwise, it results in 0.

- `expr1 <= expr2 | expr1 < expr2 | expr1 = expr2 | expr1 != expr2 | expr1 >= expr2 | expr1 > expr2`
  If both `expr1` and `expr2` are numeric, `expr` compares them as numbers; otherwise, it compares them as strings. If the comparison is true, the expression results in 1; otherwise, it results in 0.

- `expr1 + expr2 | expr1 - expr2`
  Performs addition or subtraction on the two expressions. If either expression is not a number, `expr` exits with an error.

- `expr1 * expr2 | expr1 / expr2 | expr1 % expr2`
  Performs multiplication, division, or modulus on the two expressions. If either expression is not a number, `expr` exits with an error.

- `expr1 : re | match expr1 re`
  matches the regular expression `re` against `expr1` treated as a string. The regular expression is the same as that accepted by ed, except that the match is always anchored—that is, there is an implied leading ^. Therefore, `expr` does not consider ^ to be a metacharacter. If the regular expression...
contains \(\ldots\), \) and it matches at least part of expr1, expr results in only that part; if there is no match, expr results in 0. If the regular expression doesn't contain this construct, the result is the number of characters matched. The function match performs the same operation as the colon operator.

\textbf{substr expr1 expr2 expr3}

Results in the substring of expr1 starting at position expr2 (origin 1) for the length of expr3.

\textbf{index expr1 expr2}

Searches for any of the characters in expr2 in expr1 and results in the offset of any such character (origin 1), or 0 if no such characters are found.

\textbf{length expr1}

Results in the length of expr1.

\textbf{(expr)}

Groups expressions.

\textbf{Options}

\textbf{–W}

Allows the expression to use hexadecimal and octal numbers.

\textbf{Usage notes}

The parser stack depth is limited to 150 levels. Attempting to process extremely complicated expressions might result in an overflow of this stack, causing an error.

\textbf{Examples}

1. The example

\begin{verbatim}
fname=src/fn_abs.c
exr $fname : '.*\(.*\)\..c'
\end{verbatim}

returns abs.

2. The example

\begin{verbatim}
a='expr $a + 1'
\end{verbatim}

adds 1 to the value of the shell variable a.

\textbf{Localization}

expr uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


\textbf{Exit values}

0

The result of expression is true.
The result of expression is false.

Failure due to any of following reasons:
• Not enough memory.
• Command-line syntax error.
• Too few arguments on the command line.
• Incorrect regular expression.
• Regular expression is too complicated.
• Nonnumeric value found where a number was expected.

Messages
Possible error messages include:

internal tree error
Syntax errors or unusual expression complexity make it impossible for expr to evaluate an expression. If an expression has syntax errors, correct them; if not, simplify the expression (perhaps by breaking it into parts).

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
In the shell, let largely supersedes this command.
match, substr, length, and index are not documented on all UNIX systems, though they do appear to exist. They are extensions of the POSIX standard.

Related information
ed, let, sh, test
See Appendix C, “Regular expressions (regexp),” on page 945 for more information about regexp.

exrecover daemon - Retrieve vi and ex files

Format
exrecover [-s] [name_file ...]
exrecover [-v]

Description
The exrecover daemon recovers text files from working files created by vi and ex. (These working files are in one or more temporary directories.) It is normally invoked from a system startup file before these working files are purged.

Options
-s
Suppresses error messages.
-v
Displays the version number of exrecover.
Environment variables

`exrecover` uses the following environment variables:

**TMP_VI**
Contains a directory path name that can be specified by an administrator as a location for vi temporary files. This is useful if the current default directory for these files (usually `/tmp`) is implemented as a TFS. In this case, all vi temporary files that the `exrecover` daemon uses for recovery would be gone after a system crash.

IBM recommends that this environment variable be set by a system administrator as opposed to a user setting it for their environment. If the latter occurs and the user sets the TMP_VI directory to something different than what `exrecover` recognizes as TMP_VI, the user will need to run the `exrecover` daemon manually to allow the temporary files to be converted to the recoverable files used by vi (located in `/etc/recover/$LOGNAME`).

The system administrator should not do the following:
- Set TMP_VI to `/etc/recover/$LOGNAME`.
- Set TMP_VI to any directory where a path name component is an environment variable with a user's value different than the initialization process's value (for example, `$HOME`). vi temporary files are converted into a form recoverable by vi when `exrecover` is run during IPL. Because `exrecover` is issued during IPL, it is owned by the initialization process and will therefore contain different values for certain environment variables, if those environment variables are set. Throughout the file system, there may exist some temporary files that can only be converted by `exrecover`. This conversion can be done manually by a system administrator (to recover files owned by all users) or by a single user (to recover only their own files).

**TMPDIR**
The default directory. When this environment variable is set, `exrecover` looks in this directory for the ex and vi working files.

**TMP**
If TMPDIR is not set, TMP specifies the directory to be searched when looking for the ex and vi working files.

If both TMPDIR and TMP are not set, `exrecover` uses the directory that the XL C/C++ runtime library function `tempnam()` would use.

**Localization**

`exrecover` uses the following localization environment variables:
- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Files**

`exrecover` uses the following files:

- `/tmp/VII*` Line table files.
- `/tmp/VIN*` Name files.
- `/tmp/VIIt*` Paged text files.
The directory containing subdirectories of user names whose files have been recovered. Only users with the appropriate privileges, such as the system administrator, can create the /etc/recover directory.

/etc/recover/$LOGNAME/VIn*
Contains the name of the file that was being edited.

/etc/recover/$LOGNAME/VIt*
Contains the recovered text of the file that corresponds to the VIn* file

Using a TFS for vi temporary files will make it impossible to recover vi files after a system crash. vi writes temporary files to TMP_VI or TMPDIR (or /tmp by default), and if the system crashes, those files can be recovered by the exrecover command, which automatically runs from /etc/rc. If the files are written to a TFS, then they will be wiped out when the system is IPLed. See the TMP_VI description under Environment Variables section of this command.

Usage notes

1. To recover all the files in the temporary directory, this command must be run with appropriate permissions (for example, superuser privileges) so the recovered files can be stored in the /etc/recover directory with the appropriate ownerships and permissions.

   For example, the following is a shell script to recover the files from TMPDIR, where TMPDIR is the default directory:

   ```bash
   export TMPDIR=/tmp
   exrecover
   ```

2. If it is invoked by a nonprivileged user (for example, a user who is not a root user), then only those files that are owned by that user are recovered. Because vi and ex create their working files in directories that are specified by the TMPDIR or TMP environment variables, one of these environment variables must be set before exrecover can be issued.

   For example, the following is a shell script that recovers files from $HOME/tmp:

   ```bash
   export TMPDIR=$HOME/tmp
   exrecover
   ```

3. exrecover is also invoked by vi or ex when you issue the ex preserve command or when exrecover receives a SIGHUP signal. The working files created by vi and ex are found in a default temporary directory (such as /tmp) or in the directory specified by the TMPDIR or in the directory specified by the TMP_VI, TMPDIR, or TMP environment variable. Three working files are created:

   name_file
   Contains the actual name of the vi file. The names of all name_files begin with VIn.

   line_table_file
   Contains a dummy page followed by data that gives, in line number order, the offset for each line of text in the corresponding paged_text_file. The page size is typically 1K, but may vary on some systems. The names of all line table files begin with VIl.

   paged_text_file
   Contains lines of text that are at most LINE_MAX bytes in length. Lines shorter than LINE_MAX byte are ended by a newline. The names of all paged text files begin with VIt.

4. You can also run the program by specifying name_file on the command line. For example:

   ```bash
   exrecover /tmp/VInaaaa.111 /tmp/VInbbbb.222 ...
   ```

   exrecover searches for a name_file and tries to open the associated line table and paged text files. If all these files are found, exrecover builds, from the line table and paged text files, a text file and stores it in the directory /etc/recover/$LOGNAME.
It also stores a corresponding name_file to identify the file that was recovered and sends mail, using the mailx utility, to the owner of the file indicating the date, time, and name of the file recovered. You can retrieve recovered files in one of the following ways:

```
vi -r file [issued from a shell command line]
ex -r file [issued from a shell command line]
:recover file [issued from within a vi session]
```

Each command loads the most recent occurrence of the file recovered from a system failure or the ex preserve command. If vi successfully loads the file, it removes the preserved file.

**Exit values**

0
   Successful completion.

1
   Failure due to any of the following reasons:
   • Memory allocation error.
   • No working files were found.
   • No /etc/recover directory errors that affect the overall operation of the exrecover command.
   • An incomplete set of working files were found.

2
   Usage error.

3
   An error occurred while recovering a specific file. Some, but not all, files were recovered.

**Related information**

ex, vi

**extattr - Set, reset, and display extended attributes for files**

**Format**

```
extattr [+alps][-alps][-Fformat] file ...
```

**Note:** l is a lowercase L, not an uppercase i.

**Description**

The extattr command sets, resets, and displays extended attributes for files.

**Extended attributes**

The following extended attributes are defined:

a
   When this attribute is set (+a) on an executable program file (load module), it behaves as if loaded from an APF-authorized library. For example, if this program is exec()ed at the job step level and the program is linked with the AC=1 attribute, the program will be executed as APF-authorized.

   To be able to use the extattr command for the +a option, you must have at least read access to the BPX.FILEATTR.APF resource in the FACILITY class profile. For more information about BPX.FILEATTR.APF, see z/OS UNIX System Services Planning.

1
   When this attribute is set (+1) on an executable program file (load module), it will be loaded from the shared library region.
To be able to use the `extattr` command for the +l option, you must have at least read access to the BPX.FILEATTR.SHARELIB resource in the FACILITY class. For more information about BPX.FILEATTR.SHARELIB, see z/OS UNIX System Services Planning.

**Note:** 1 is a lowercase L, not an uppercase I.

p

When this attribute is set (+p) on an executable program file (load module), it causes the program to behave as if an RDEFINE had been done for the load module to the PROGRAM class. When this program is brought into storage, it does not cause the environment to be marked dirty.

To be able to use the `extattr` command for the +p option, you must have at least read access to the BPX.FILEATTR.PROGCTL resource in the FACILITY class. For more information about BPX.FILEATTR.PROGCTL, see z/OS UNIX System Services Planning.

s

When this attribute is not set (-s), the _BPX_SHAREAS=YES and _BPX_SHAREAS=REUSE environment variable settings are ignored when the file is spawn()ed. Use of the _BPX_SHAREAS=MUST setting and the -s option will result in a spawn() failure. By default, this attribute is set (+s) for all executable files.

**Note:** In addition to the access permissions to the BPX.FILEATTR.XXXX (XXXX is APF, SHARELIB or PROGCTRL) profiles in the FACILITY class described in the command description, the user must have write access to the file or have SUPERUSER authority. The SUPERUSER.FILESYS UNIXPRIV class profile does not grant authority to update any extended attributes.

**Options**

- **F format file ...**

  `extattr` command will accept the -F option flag with values consistent with the cp command to indicate the format of the file. The command will set the file format accordingly.

  Setting the file format flag on a file does not modify the data in the file. Use the `ls -H` to display the file format.

  For `format`, you can specify:

  - **BIN**
    - Binary data
  - **CR**
    - Carriage return
  - **CRLF**
    - Carriage return followed by line feed
  - **CRNL**
    - Carriage return followed by a newline character
  - **LF**
    - Line feed
  - **LFCR**
    - Line feed followed by carriage return
  - **NA**
    - Not specified
  - **NL**
    - Newline character
  - **REC**
    - File data consists of records with prefixes. The record prefix contains the length of the record that follows.

  The format option can be specified in lowercase, uppercase or in mixed cases. The format option can also be specified with a space or no space after the file format flag (-F). For example: `extattr -FLFCR file`
The file format flag (-F) can be used with other `extattr` flags (+alps/-alps), but it must be separated by a space or tab. For example:

```
  extattr +aps -F BIN file is a valid entry.
  extattr -apsF NA file is not a valid entry.
```

**Usage notes**

The APF-authorized (a), shared library (l) and program-control (p) attributes are reset by the system if the file is opened for write, an external link to the file is created, or the file is renamed.

**Examples**

Following are valid examples of the use of `extattr`:

```
  extattr +ap -F BIN -sl <filename>
  extattr -F NA -aps +l <filename>
  extattr -FCRnl <filename>
```

To have the `c89` and `tso` utilities not run in an address space shared with other processes, issue:

```
  extattr -s /bin/c89 /bin/tso
```

**Related information**

`chtag`, `ls`, ISHELL

---

### `false` - Return a nonzero exit code

**Format**

```
false [argument ...]
```

**Description**

`false` returns an exit status value of 1 (failure). It ignores any arguments given on the command line. This option can be useful in shell scripts.

**Usage notes**

`false` is a built-in shell command.

**Localization**

`false` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Exit values**

`false` always returns an exit status value of 1.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.
fc - Process a command history list

Format

fc [-r] [-e editor] [first][last]]
f -l [-nr] [first][last]]
f -s [old=new] [specifier]

Description

fc displays, edits, and reenters commands that have been input to an interactive shell. fc stands for “fix commands.” If the variable HISTSIZE is not defined, 128 commands are accessible. The number of commands that are accessible is determined by the HISTSIZE variable.

The shell stores these commands in a history file. When the HISTFILE environment variable is defined as the name of a writable file, the shell uses this as the history file. Otherwise, the history file is $HOME/.sh_history, if HOME is defined and the file is writable. If the HOME variable is not defined, or the file is not writable, the shell attempts to create a temporary file for the history. If a temporary file cannot be created, the shell does not keep a history file.

Note: A shell shares history (commands) with all shells that have the same history file. A login shell truncates the history file if it is more than HISTSIZE lines long.

Normally, the shell does not keep a history of commands run from a profile file or the ENV file. By default, however, it begins recording commands in the history file when it encounters a function definition in either of these setup files. This means that the HISTSIZE and HISTFILE variables must be set up appropriately before the first function definition. If you do not want the history file to begin at this time, use:

```
set -o nolog
```

For more information, see sh and set. Any variable assignment or redirection that appears on the fc command line affects both the fc command itself and the commands that fc produces.

The first form of fc in “Format” on page 302 puts you into an editor with a range of commands to edit. When you leave the editor, fc inputs the edited commands to the shell.

The first and last command in the range are specified with first and last. There are three ways to specify a command.

- If the command specifier is an unsigned or positive number, fc edits the command with that number.
- If the command specifier is a negative number –n, fc edits the command that came n commands before the current command.
- If the command specifier is a string, fc edits the most recent command beginning with that string.

The default value of last is first. If you specify neither first nor last, the default command range is the previous command entered to the shell.

Options

- **-e editor**
  
  Invokes editor to edit the commands. If you do not specify the -e option, fc assumes that the environment variable FCEDIT, if defined, contains the name of the editor for fc to use. If FCEDIT is not defined, fc invokes ed to edit the commands.
-l
Displays the command list. This option does not edit or reenter the commands. If you omit last with this option, fc displays all commands from the one indicated by first through to the previous command entered. If you omit both first and last with this option, the default command range is the 16 most recently entered commands.

-n
Suppresses command numbers when displaying commands.

-r
Reverses the order of the commands in the command range.

-s
Reenters exactly one command without going through an editor. If a command specifier is given, fc selects the command to reenter as described earlier; otherwise, fc uses the last command entered. To perform a simple substitution on the command before reentry, use a parameter of the form old=new. The string new replaces the first occurrence of string old. fc displays the (possibly modified) command before reentering it.

Environment variables

FCEDIT
Contains the default editor to be used if none is specified with the –e option.

HISTFILE
Contains the path name of the history file.

HISTSIZE
Gives the maximum number of previous commands that are accessible.

Files

/tmp
Used to store temporary files. You can use the TMPDIR environment variable to dictate a different directory to store temporary files.

$HOME/.sh_history
This default history file is created.

Localization

fc uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

Usage notes

1. fc is a built-in shell command.
2. r is a built-in alias for fc –s. history is a built-in alias for fc –l.

Exit values

0
If you specified -l, this indicates successful completion.

1
Failure due to any of the following reasons:
• Missing history file.
• Inability to find the desired line in the history file.
• Inability to create temporary file.

2

An incorrect command-line option or argument.

If fc runs one or more commands, the exit status of fc is the exit status of the last run command.

Messages

Possible error messages include:

Cannot create temporary file

fc must create a temporary file to do some operations, such as editing. It prints this message when it cannot create its temporary file—for example, because the disk is full.

No command matches string

You asked to edit a command beginning with a particular string, but there was no such command in the history file.

Portability

POSIX.2.

Related information

alias, ed, print, read, sh, vi

fg - Bring a job into the foreground

Format

fg [%job-identifier]
tcsh shell: fg [%job ...]

Description

fg restarts a suspended job or moves a job from the background to the foreground. To identify the job, you give a job-identifier (preceded by %) as given by the jobs command.

If you do not specify job-identifier, fg uses the most recent job to be suspended (with the kill command) or placed in the background (with the bg command). fg is available only if you have enabled job control. See the -m option of set for more information.

In the tcsh shell, fg brings the specified jobs (or, without arguments, the current job) into the foreground, continuing each if it is stopped. job can be ", %, +, -, a number, or a string. See also the run-fg-editor editor command that is described in “tcsh: Invoke a C shell” on page 671.

Localization

fg uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH

Exit values
0
   Successful completion.
>0
   No current job.

Messages
Possible error messages include:
Not a stopped job
   Job was not stopped.

Portability
POSIX.2 User Portability Extension.

Related information
bg, jobs, kill, ps, tcsh

fgrep - Search a file for a specified pattern

Format
fgrep [Bncilnqsvx][-A | -P | -C num][-S string]
[-W option[option] ... [-e pattern] ... [-f patternfile] ... [pattern] [file ... file ...]

The fgrep utility is fully supported for compatibility with older UNIX systems. However, use grep -F instead because it might provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

Description
See grep.

file - Determine file type

Format
file [-BcdEh] [-f filelist] [-M magic] [-m magic] [-W option[option] ... file ... file ...]
file -i [-h] [-f filelist] file ... file ...

Description
file determines the format of each file by inspecting the attributes and (for a regular file) reading the contents of the file. If the file is an executable, its addressing mode is determined for output. If file is not an executable, file compares each file to entries found in one or more magic files to determine their file type. If you specify - as a file name, file reads from the standard input (stdin).

file then divides files that do not match a template in the magic file into text files and binary data. Then, by reading the text files and making an informed guess based on the contents, file further divides text files into various types such as C programs, assembler programs, files of commands to the shell, and yacc or lex programs.
The `file` utility uses three types of tests to determine the file type: the file attribute tests, the position-sensitive tests and the context-sensitive tests.

- The file attribute tests determine file types such as directory, character special, FIFO, socket, symbolic link, and external link.
- The position-sensitive tests determine file types by looking for certain string or binary values at specific offsets in the file being examined. The default position-sensitive tests are defined by:
  - The `/etc/magic` file
  - The AMODE test built into the `file` utility

If a magic file test succeeds, the message field of the line will be printed and no further tests will be applied, except for tests on immediately following lines beginning with a single `>` character.

- The default context-sensitive tests are built into the `file` utility. These tests look for language constructs in text files trying to identify shell scripts, C, FORTRAN, and other computer language source files, and even plain text files. The default context-sensitive tests will never be applied before any position-sensitive tests even if the `-d` option is specified before an `-m magic` option or `-M magic` option.

Options

-B
   Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgm codeset options (-W option) are specified.

-c
   Only checks the file `magic` (specified by `-m` or `-M`) and `/etc/magic`. If the semantics imply it, see the usage notes for the validity of the format.

-d
   Apply any default position-sensitive tests and default context-sensitive tests to the file. This option is the default if no `-M` or `-m` option is specified. See the usage notes for more information.

-E
   Uses the magic file and bypasses the checking of regular files for executables.

- f `filelist`
   Examines the files listed in the file `filelist`.

-h
   When a symbolic link is encountered, identify `file` as a symbolic link instead of following the link.
   If `-h` is not specified and
   - `file` is an external link or `file` is a symbolic link referring to a nonexistent file:
     - **If the variable `_UNIX03=YES` is set**
       The type will be reported as if `-h` was specified.
     - **If the variable `_UNIX03` is unset or is not set to YES**
       The type will be reported as if `-h` was not specified.

-i
   If `file` is a regular file, does not attempt to classify the type of the file further. This option can only be used with `-h` and `-f` options. See the usage notes for the file types that `file` command does not attempt to classify.
-M magic
Uses the file magic to classify the file type. No default position-sensitive tests, default context-sensitive tests, nor AMODE tests shall be applied, unless the -d option is also specified. See the usage notes for more information.

-m magic
Alters the classification of regular files when examining the file content.

If the variable _UNIX03=YES is set
Then file attempts to classify the file type using the following tests, in order:
1. Using the file magic.
2. Using the default position-sensitive tests (/etc/magic).
3. Using the default context-sensitive tests built into the file command.

If the variable _UNIX03 is unset or is not set to YES
Then file attempts to classify the file type using the following tests, in order:
1. Using the file magic rather than /etc/magic.
2. Using the default context-sensitive built into the file command.

-W option[,...]
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

filecodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

pgmcodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

Restriction: The only supported values for pgmcodeset are IBM-1047 and 1047.
Examples

1. To display the type of a file:

   file myFile

2. To display the type of a file, without attempting to classify regular files:

   file -i myFile

3. To display the type of a text file containing UTF-8 characters, assuming that:
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are checking an untagged public text file or a read-only text file)

   file -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File

4. To display the type of a text file containing EBCDIC characters, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as ASCII:

   file -B myMisTaggedFile

Localization

file uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


Environment variables

file uses the following environment variables:

_UNIX03

See Appendix N, “Shell commands changed for UNIX03,” on page 1005 for more information about the effect of the _UNIX03 environment variable on this command.

_TEXT_CONV

Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

Files

file uses the following file:

/etc/magic

Default system magic file.

For file to work, you need to copy the magic file from the /samples directory to the /etc directory.

For more information about enabling file, see the section on copying configuration files in z/OS UNIX System Services Planning. Additional information about the magic file can be found in “magic - Format of the /etc/magic file” on page 970.
Usage notes

1. Files in code pages other than IBM-1047 might be considered as binary data unless the file is tagged and autoconversion is turned on.

2. LC_SYNTAX only affects the interpretation of the input file that did not match any magic file template. It does not affect the interpretation of the magic file.

   Because of this, an input file that contains characters from a code page other than IBM-1047 cannot match the magic file, which contains IBM-1047 characters. If you need to match character in different code pages, you can use the -m or -M option to specify a magic file created with the desired code page.

3. The tests applying to a file when running the command follow the rules outlined in Table 13 on page 309.

<table>
<thead>
<tr>
<th>If the following options are specified</th>
<th>Then the position-sensitive tests are applied in the following sequence</th>
<th>Default context-sensitive tests applied?</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>AMODE &gt; /etc/magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-d</td>
<td>AMODE &gt; /etc/magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-M MAGIC</td>
<td>MAGIC</td>
<td>No</td>
</tr>
<tr>
<td>-m magic</td>
<td>• If the variable _UNIX03= YES is set,</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>magic &gt; AMODE &gt; /etc/magic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If the variable _UNIX03 is unset or is not set to YES,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AMODE &gt; magic</td>
<td></td>
</tr>
<tr>
<td>-d -M MAGIC</td>
<td>AMODE &gt; /etc/magic &gt; MAGIC</td>
<td>Yes</td>
</tr>
<tr>
<td>-M MAGIC -d</td>
<td>MAGIC &gt; AMODE &gt; /etc/magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-d -m magic</td>
<td>AMODE &gt; /etc/magic &gt; magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-m magic -d</td>
<td>magic &gt; AMODE &gt; /etc/magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-M MAGIC -m magic</td>
<td>MAGIC &gt; magic</td>
<td>No</td>
</tr>
<tr>
<td>-m magic -M MAGIC</td>
<td>magic &gt; MAGIC</td>
<td>No</td>
</tr>
<tr>
<td>-d -M MAGIC -m magic</td>
<td>AMODE &gt; /etc/magic &gt; MAGIC &gt; magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-d -m magic -M MAGIC</td>
<td>AMODE &gt; /etc/magic &gt; magic &gt; MAGIC</td>
<td>Yes</td>
</tr>
<tr>
<td>-M MAGIC -d -m magic</td>
<td>MAGIC &gt; AMODE &gt; /etc/magic &gt; magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-M MAGIC -m magic -d</td>
<td>MAGIC &gt; magic &gt; AMODE &gt; /etc/magic</td>
<td>Yes</td>
</tr>
<tr>
<td>-m magic -d -M MAGIC</td>
<td>magic &gt; AMODE &gt; /etc/magic &gt; MAGIC</td>
<td>Yes</td>
</tr>
<tr>
<td>-m magic -M MAGIC -d</td>
<td>magic &gt; MAGIC &gt; AMODE &gt; /etc/magic</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note:

a. The first column specifies the appearance of the -d, -M and -m options in the command line.

b. The second column gives what position-sensitive tests are applied and in what sequence, given the options specified in the first column:
   - AMODE is a default position-sensitive system test which is only used on an executable file to determine the addressing mode.
   - /etc/magic means the default position-sensitive tests in /etc/magic.
   - MAGIC means the position-sensitive tests in the magic file specified by -M.
   - magic means the position-sensitive tests in the magic file specified by -m.
• Tests not appearing in the cell are not applied.

c. The third column gives whether the default context-sensitive tests (built into the file command) are applied, given the options specified in the first column.

4. If -d option is specified together with -E option, the AMODE tests will not be applied. If -M magic option is specified alone, the AMODE tests will not be applied.

5. The standard output messages of the file utility will contain the specified strings, but not limited to, listed in Table 14 on page 310.

<table>
<thead>
<tr>
<th>If file is:</th>
<th>Will contain the string:</th>
<th>See note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonexistent</td>
<td>Cannot open</td>
<td>None</td>
</tr>
<tr>
<td>Block special</td>
<td>Block special</td>
<td>a</td>
</tr>
<tr>
<td>Character special</td>
<td>Character special</td>
<td>a</td>
</tr>
<tr>
<td>Directory</td>
<td>Directory</td>
<td>a</td>
</tr>
<tr>
<td>FIFO</td>
<td>Fifo</td>
<td>a</td>
</tr>
<tr>
<td>Socket</td>
<td>Socket</td>
<td>a</td>
</tr>
<tr>
<td>Symbolic link</td>
<td>Symbolic link to</td>
<td>a</td>
</tr>
<tr>
<td>External symbolic link</td>
<td>External link to</td>
<td>a</td>
</tr>
<tr>
<td>Regular file</td>
<td>Regular file</td>
<td>a,b</td>
</tr>
<tr>
<td>Empty regular file</td>
<td>Empty</td>
<td>c</td>
</tr>
<tr>
<td>Regular file that cannot be read</td>
<td>Cannot open</td>
<td>c</td>
</tr>
<tr>
<td>Executable binary</td>
<td>Executable</td>
<td>d,f</td>
</tr>
<tr>
<td>ar archive library (see ar)</td>
<td>Archive</td>
<td>d,f</td>
</tr>
<tr>
<td>Extended cpio format (see pax)</td>
<td>cpio archive</td>
<td>d,f</td>
</tr>
<tr>
<td>Extended tar format (ustar in pax)</td>
<td>tar archive</td>
<td>d,f</td>
</tr>
<tr>
<td>Shell script</td>
<td>Commands text</td>
<td>e,f</td>
</tr>
<tr>
<td>C-language source</td>
<td>C program text</td>
<td>e,f</td>
</tr>
<tr>
<td>FORTRAN source</td>
<td>Fortran program text</td>
<td>e,f</td>
</tr>
<tr>
<td>Regular file whose type cannot be determined</td>
<td>Data</td>
<td>None</td>
</tr>
</tbody>
</table>

Note:

a. This is a file attribute test.
b. This test is applied only if the -i option is specified.
c. This test is applied only if the -i option is not specified.
d. This is a default position-sensitive test.
e. This is a default context-sensitive test.
f. Default position-sensitive tests and default context-sensitive tests are not applied if the -M magic option is specified unless the -d option is also specified.
Exit values

0
  Successful completion

1
  Failure due to any of the following reasons:
  • A missing filelist after -f.
  • More than one -f option on the command line.
  • Cannot find the magic file.
  • Incorrect command-line option.
  • Too few command-line arguments.
  • Cannot access a specified file.
  • Cannot open filelist.
  • Cannot open the magic file.
  • A format error in the magic file.
  • Out of memory for reading or magic entries.
  • A bad number in the magic file.
  • A misplaced > in the magic file.
  • The code set is not valid.
  • Could not turn off automatic conversion.
  • Could not perform requested text conversion.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
All options are extensions of the POSIX standard.
The -B and -W options are extensions of the POSIX standard.

Related information

“magic - Format of the /etc/magic file” on page 970 for more information about the magic file format.

find - Find a file that meets specified criteria

Format

find path ... expression

Description

find searches a given file hierarchy that is specified by path, finding files that match the criteria given by expression. Each directory, file, and special file is passed through expression. If you use the -exec, -ok, or -cpio primary, expression runs a specified command on each file found. A nonexistent expression or an expression with commands to run automatically uses the -print primary to display the name of any file that matches the criteria of expression.

find builds expression from a set of primaries and operators; juxtaposition of two primaries implies a logical AND operator.

Operators and primaries

find supports the following operators:
find

-a
Used between primaries for a logical AND. You can omit this operator to get the same result, because logical AND is assumed when no operator is used between two primaries.

-o
Used between primaries for a logical OR.

!
Precedes an expression in order to negate it.

When using the `find` command, follow these rules:

- When you are using parentheses to group primaries and operators, you must escape the parentheses with the `\` (backslash) character if the command is being executed in the shell environment.
- You must delimit all primaries, operators, numbers, arguments, and parentheses with white space.

Each `number` in the primary list is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, `find` tests for equality; a plus sign implies "greater than" or "older than" and a minus sign implies "less than" or "newer than".

Options

The `find` command supports the following primaries.

**Tip:** If you use the ACL primaries, with the exception of -acl, performance might be affected.

- `a审计 auditmask`
  The -a auditmask primary is used to match the auditor audit bits. See -audit auditmask.

- `acl c`
  Matches if the type of ACL is the same as the type given by the character c. Possible values of the character are:
  - a
    Access ACL (matches only if there are extended ACL entries)
  - d
    Directory default ACL
  - f
    File default ACL

  If acl c is not defined, then `find` matches any of these ACLs when other ACL primaries are used.

- `acl_count number`
  Matches if the numbers of extended ACL entries for any of the types of ACLs for the object is `number`.

  `number` is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, `find` tests for equality; a plus sign implies “greater than” or “older than,” and a minus sign implies “less than” or “newer than”.

- `acl_entry acl_text`
  Matches if the ACL on the file contains an entry equivalent to `acl_text` where `acl_text` is a single extended ACL entry. This primary uses user and group names rather than UID and GID numbers to make the matches. If aliases exist for a name, then it is possible a match might not occur. This behavior is different than the -acl_user and -acl_group primaries, which match based on UID and GID. Extended ACL entries have the following format:

  ```
  [d[efault]: | f[efault]:]u[ser]:uid:[][+|^]perm
  [d[efault]: | f[efault]:]g[roup]:gid:[][+|^]perm
  ```

  where:

  - `d[efault]`
    If specified, extended ACL refers to directory default ACL

  - `f[efault]`
    If specified, extended ACL refers to file default ACL
Extended ACL refers to a particular numeric user ID (UID) or user name

Extended ACL refers to a particular numeric group ID (GID) or group name

User name or numeric user ID (UID)

Group name, or numeric group ID (GID)

Permissions specified either in absolute form (string rwx with - as a placeholder or octal form), or in relative format (by using the + or ^ modifiers).

For relative permission settings, specifying +perm means that you want the ACL entry to have that permission turned on. Specifying ^perm means that you want the ACL entry to have that permission off. For example, specifying the following will find files with an extended access ACL entry for user Billy in which the permissions are either -w- or rw-:

```
user:Billy:+w^x
```

If the permission field of acl_text is omitted, then the ACL entries are searched to match only the ACL type, and user or group portions of the user-supplied entry.

If you want to find any of the base ACL entries (user, group, or other), you can use the -perm primary.

The first field of an ACL entry can specify the type of ACL (access, directory default, or file default) that will be processed. If the type is not specified, the operation applies only to the access ACL. If you are updating the ACL entries, you can specify the base ACL entries; however, specifying the base ACL entries might cause the file or directory's permission bits to change if what is specified is different than the current settings.

```
-acl_group groupid
```

Matches if the object has an extended group ACL entry for groupid. groupid can also be a group ID number.

If your security product supports ACLs, the group base ACL entry can be matched using this primary. If a numeric group exists as a group name in the group data base, the group ID number that is associated with that group is used.

```
-acl_nogroup
```

Matches if a group ACL entry (for any type of ACL) exists in which a group is not defined. The GID for at least one extended ACL entry for the file does not have a group name that is associated with it.

```
-acl_nouser
```

Matches if a user ACL entry (for any type of ACL) exists in which a user is not defined. The UID for at least one extended ACL entry for the file does not have a user name that is associated with it.

```
-acl_user userid
```

Matches if the ACL of the object has an extended user ACL entry for userid. userid can also be a user ID number.

If a numeric owner exists as a user name in the user database, the user ID number that is associated with that user name is used. If your security product supports ACLs, the user base ACL entry can be matched, using this primary.

```
-atime number
```

Matches if someone has accessed the file exactly number days ago.

number is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, find tests for equality; a plus sign implies “greater than” or “older than,” and a minus sign implies “less than” or “newer than”.

Shell command descriptions 313
-audit auditmask

The -audit primary is used to match the user audit bits. auditmask can be in octal or in symbolic form. The mask can be preceded by a - character (as in the perm primary), but it is ignored.

Symbolic form is an operation=condition list, separated by commas:

\[ [\text{rwx}]=\text{[sf]} \]

where:

=sf
Success or failure on any of rwx

r=s
Success on read

r=s, x=sf
Success on read or exec, failure on exec

r, w=s
Incorrect syntax

x
Incorrect syntax

Octal form is specified by using the chaudit bit constant definitions in the /usr/include/sys/stat.h header file. For example, in stat.h, the flag for failing read accesses is AUDTREADFAIL. It is defined to be 0x02000000, which has an octal value of 20000000. This octal value can be used as the auditmask to find failure on read.

-cpio cpio-file

 Writes the file found to the target file cpio-file in cpio format. This is equivalent to:

\[
\text{find ... | cpio -o >cpio-file}
\]

This primary matches if the command succeeds.

-ctime number

Matches if someone changed the attributes of the file exactly number days ago.

number is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, find tests for equality; a plus sign implies “greater than” or “older than,” and a minus sign implies “less than” or “newer than”.

-depth

Processes directories after their contents. If present, this primary always matches.

-exec command ;

Takes all arguments between -exec and the semicolon as a command line, replacing any argument that is exactly {} (that is, the two brace characters) with the current file name. It then executes the resulting command line, treating a return status of zero from this command as a successful match, nonzero as failure. You must delimit the terminal semicolon with white space.

Rule: The semicolon is a shell metacharacter. To use it in expression, you must escape it, either by enclosing it in single quotation marks or by preceding it with a backslash (\).

-ext c

Matches when the regular file has the extended attribute that is specified by character c. See “extattr - Set, reset, and display extended attributes for files” on page 299 for details on extended attributes.

Possible values of the character are:

a
Program runs APF-authorized if linked AC = 1

l
Program is loaded from the shared library region
Program is considered program-controlled

Program is allowed to run in a shared address space

**-filetag c**
Matches if the file tag is the same as the one given by character c. Possible values of the character are:

- **b**
  Matches if the file is tagged as binary (txtflag = OFF and ccsid = 0xFFFF)

- **n**
  Matches if the file has txtflag = OFF

- **t**
  Matches if the file is tagged as text (txtflag = ON)

- **u**
  Matches if the file is untagged (ccsid = 0)

**-filetag_codeset codeset**
Matches if the file is tagged with the given code set. *codeset* can be a code set name that is known to the system or the numeric coded character set identifier (CCSID). If a code set name exists, the numeric CCSID associated with that name is used. Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names.

**Note:** Because code sets that are aliases of each other exist, tests might fail if the file inquiry operator returns an alias of the code set that you are testing.

**-follow**
Follows symbolic links. If present, this primary always matches.

**-group name**
Matches if the group owner is *name*. If *name* is not a valid group name, it is treated as a group ID.

**-inum number**
Matches if the file has inode number *number*.

*number* is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, **find** tests for equality; a plus sign implies “greater than” or “older than,” and a minus sign implies “less than” or “newer than”.

**-level number**
Does not descend below *number* levels.

*number* is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, **find** tests for equality; a plus sign implies greater than or older than, and a minus sign implies less than or newer than.

**-links number**
Matches if there are *number* links to the file.

*number* is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, **find** tests for equality; a plus sign implies greater than or older than, and a minus sign implies less than or newer than.

**-mtime number**
Matches if someone has modified the file exactly *number* days ago.

*number* is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, **find** tests for equality; a plus sign implies greater than or older than, and a minus sign implies less than or newer than.

**-name pattern**
Compares the current file name with *pattern*. If there is no match, the expression fails. The pattern uses the same syntax as file name generation (see **sh**). It matches as many trailing path name components as specified in *pattern*. Slashes in the pattern are matched explicitly.
-ncpio cpio-file
Writes the file found to the target file cpio-file in cpio -c format. This is equivalent to:

```
find ... | cpio -oc >cpio-file
```

This primary matches if the command succeeds.

-newer file
Comparis the modification date of the found file with that of the file given. This matches if someone has modified the found file more recently than file.

-nogroup
Matches if no defined group owns the file.

-none
Indicates that some action was taken; thus find does not invoke the default -print action. If present, this primary always matches.

-nouser
Matches if no defined user owns the file.

-okcommand;
Is similar to -exec, but before find executes the command, it displays the command to confirm that you want to go ahead. find executes the command line only if your input matches the expression for “yes” (yes and no expressions are defined in LC_MESSAGES). If you type the expression for “no”, the primary does not match. You must delimit the terminal semicolon with white space.

The semicolon is a shell metacharacter. To use it in expression, you must quote it.

-perm[-] mask
By default, matches if the permissions on file are identical to the ones given in mask. You can specify mask in octal or in symbolic mode (see chmod). If you use symbolic mode, find assumes that you begin with no bits set in mask, and that the symbolic mode is a recipe for turning the bits you want on and off. A leading minus sign (-) is special. It means that a file matches if at least all the bits in mask are set. As a result, with symbolic mode, you cannot use a mask value that begins with a minus sign (-).

If you use octal mode, find uses only the bottom 12 bits of mask. With an initial minus sign (-), find again matches only if at least all the limits in mask are set in the file permissions lists.

-print
Displays the current file name. This primary always matches.

-prune
Stops searching deeper into the tree at this point. If present, this primary always matches. -prune has no effect if -depth is also specified.

-seclabel pattern
Compares the file’s security label with pattern. If there is no match, the expression fails. The pattern uses the same syntax as file name generation (see “File name generation” on page 604).

-size number[c]
Matches if the size of the file is number blocks long, where a block is 512 bytes. If you include the suffix c, the file size is number bytes.

number is a decimal number, optionally preceded by a plus or minus sign. If a number is given without a sign, find tests for equality. A plus sign implies "greater than" or "older than" while a minus sign implies "less than" or "newer than".

-type c
Matches if the type of the file is the same as the type given by the character c. Possible values of the character are:

b
Block special file (not supported for z/OS UNIX System Services)
Character special file
Directory
Regular file
Symbolic link
Network file
FIFO (named pipe)
Socket

```
-user name
Matches if the owner of the file is name. name can also be a user ID number.
-xdev
Does not cross device boundaries from the root of the tree search. If present, this primary always matches.
```

**Examples**

1. To find all files with a suffix of .c that have the audit mode set to rwx (read, write, execute), issue:

   ```
   find / -name "*.c" -audit rwx=sf
   ```

   The quotation marks are required around the "*.c" if you do not want the shell to expand this value to all files with a suffix of .c from within the current directory.

2. To find all files with a suffix of .c and audit mode bits set to 777 (rwx), issue:

   ```
   find / -name "*.c" -audit 777
   ```

3. To find all files with the extensions .c and .h, starting at the current point in the directory hierarchy:

   ```
   find . -name "*.c*.h"
   ```

4. To find all files that have the extension .z and that have not been accessed in the last three days:

   ```
   find . -name "*.Z" -mtime +3
   ```

5. To find all files that have a security label of OS390:

   ```
   find . -seclabel OS390
   ```

6. To find all files that have a security label starting with OS390:

   ```
   find . -seclabel "OS390*"
   ```

7. To find all files that have no security labels:

   ```
   find . ! -seclabel "*
   ```

8. To find all files and directories starting at the current directory point, with an extended ACL user entry for user Billy for any ACL (access, file default, or directory default), issue:

   ```
   find . -acl_user Billy
   ```

   or
9. To find all files and directories (starting from the current directory) that have more than 10 extended ACL entries for any of the ACL types, issue:

    find . -acl_count +10

10. To find all files and directories containing access ACLs that have an extended ACL entry for user Averi, starting from the current user's home directory:

    find ~ -acl_entry user:Averi

11. To find all directories whose file default ACLs have a group entry for Lakers, starting at the current point in the directory hierarchy:

    find . -acl_entry fdefault:group:Lakers

12. To find all files for user Marc (in other words, all the files that Marc owns), starting from Marc's home directory:

    find /u/marc -user marc

13. To find all directories (starting from current directory) which have file default ACLs:

    find . -acl f

14. To find all directories whose file default or directory default ACLs have a group entry for Lakers, starting at the current point in the directory hierarchy:

    find . -acl_entry fdefault:group:Lakers -o -acl_entry default:group:Lakers

**Localization**

*find* uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


**Exit values**

0  
  Successful completion

1  
  Failure due to any of the following reasons:
  - Not enough memory.
  - Incorrect character was specified after -type.
  - Inability to get information about a file for -newer.
  - Incorrect permissions for -perm.
  - Inability to open a file for the -cpio option.
  - Unknown user or group name.
• Unable to access the PATH variable.
• Cannot run a command that is specified for -exec or -ok.
• Syntax error.
• Stack overflow caused by an expression that is too complex.

2
Failure due to one of the following:
• Incorrect command-line option.
• Not enough arguments on the command line.
• Missing option.
• Argument list that is not properly ended.

Messages
Possible error messages include:

**Bad number specification in string**
You specified an option that takes a numeric value (for example, -atime, -ctime) but did not specify a valid number after the option.

**Cannot stat file name for -newer**
You used a -newer option to compare one file with another; however, find could not obtain a modification time for the specified file. Typically, the file does not exist or you do not have appropriate permissions to obtain this information.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Most UNIX systems do not have a default action of -print. Therefore, they do not need the -none option. The -a operator is not documented on many UNIX systems.

The following primaries are extensions of the POSIX standard: -aaudit, -acl, -acl_count, -acl_entry, -acl_group, -acl_nogroup, -acl_nouser, -acl_user, -audit, -audit, -cpio, -follow, -level, -ncpio, -none

The aaudit and audit options are unique to the z/OS shell.

Related information
chaudit, chmod, cpio, sh

**filecache - Manage file caches**
The filecache command manages the kernel file cache for files that are only read.

While it continues to be shipped in /usr/sbin/filecache, the filecache command is not operational and will not return any cached data.

**fold - Break lines into shorter lines**

Format
fold [-bs] [-w width] [-width] [file...]

Shell command descriptions 319
**Description**

`fold` reads the standard input (stdin) or each file, if you specify any. Each input line is broken into lines no longer than `width` characters. If you do not specify `width` on the command line, the default line length is 80. The output is sent to the standard output (stdout).

**Options**

- **-b**
  Specifies `width` in bytes rather than in column positions; that is, `fold` does not interpret tab, backspace, and carriage return characters. If the last byte specified by `width` is part of a double-byte character, `fold` does not break the character. Instead, the line is broken before the double-byte character.

- **-s**
  Breaks each line at the last blank within `width` column positions. If there is no blank that meets the requirement, `fold` breaks the line normally.

- **-w width**
  Specifies a maximum line length of `width` characters.

- **--width**
  Is identical in effect to `–w width`.

**Localization**

`fold` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

0
  Successful completion.

1
  Failure because the input file could not be opened.

2
  Invalid command-line option or a missing `width` argument.

**Portability**

POSIX.2, 4.2BSD.

The `–width` option is an extension of the POSIX standard.

**Related information**

`pr`

---

**functions - Display or assign attributes to functions**

**Format**

`functions [⋆tux][name ...]`
Note: * indicates a + or - character.

Description
functions is an alias for typeset -f.
See “typeset - Assign attributes and values to variables” on page 761 for more information.

Related information
typeset, sh

fuser - List process IDs of processes with open files

Format
fuser [-cfku] file ...

Description
The fuser command writes to standard output the process IDs of all processes running on the local system that have one or more named files open. file is the path name of the file for which information is to be reported, or, if the –c options is used, the path name of a file on the file system for which information is to be reported. It also writes additional information to standard error, such as the user name of the process and a character indicating how the process is using the file.
fuser only reports on local processes, not remote ones. It only reports information for processes that the user has appropriate privileges to access. Only a superuser can access information for processes that belong to another user.

Options
-c
   Reports on all open files within the file system that the specified file is a member of.
-f
   Reports on only the named files. This is the default.
-k
   Sends the SIGKILL signal to each local process (with the exception of the fuser process and parent processes of fuser). Only a superuser can terminate a process that belongs to another user. This option is a z/OS extension.
-u
   Writes to standard error the user name associated with each process ID written to standard output.

Usage notes
fuser writes the process ID for each process to standard output. fuser also writes the following to standard error:
• The path name of each file specified on the command line.
• An indicator of how the process is using this file (written after the process ID):
  -c
     The process is using the file as its current directory.
  -r
     The process is using the file as its root directory.
If no character follows the PID, this means that the process has the file open.
When the -u option is specified, `fuser` writes the user name corresponding to the real user ID of the process.

**Examples**

1. To list the process numbers of local processes using the `/etc/magic` file, enter:
   ```bash
   fuser /etc/magic
   ```
   which will give you the following output:
   ```
   /etc/magic:  67109274 144
   ```

2. To display the user names associated with the processes accessing the file `/etc/magic`:
   ```bash
   fuser -u /etc/magic
   ```
   Your output would be:
   ```
   /etc/magic:  67109274(Steve) 144(Fred)
   ```

3. To terminate all of the processes using a given file system, enter either the mount point name or the name of a file in that file system:
   ```bash
   fuser -ku /u/home
   ```
   or
   ```bash
   fuser -kuc /u/home/code
   ```
   Your output would look like:
   ```
   /u/home/code:  111111c(Steve) 222222r(Erin) 333333(Garth)
   444444c(Steve) 555555r(Renata) 66666c(Angie)
   ```
   This command lists the process number and user name, and then sends a kill signal to each process that is using the `/u/home` file system. Only a superuser can terminate processes that belong to another user. You might want to use this command if you are trying to unmount the `/u/home` file system and a process that is accessing the file system prevents this.

**Exit values**

- **0**
  Successful completion.

- **1**
  An error occurred.

**Related information**

`kill`, `ps`

---

**gendat - Create or modify message catalogs**

**Format**

```bash
gendat CatalogFile MessageFile ...
```

**Description**

`gendat` merges the message text source files `MessageFile` (typically `.msg`) into a formatted message catalog `CatalogFile` (typically `.cat`). The file `CatalogFile` is created if it does not already exist. If
CatalogFile does exist, its messages are modified according to the directives in the MessageFiles. If set and message numbers are the same, the new message text defined in MessageFile replaces the message text defined in CatalogFile.

You can specify any number of MessageFiles. gencat processes the MessageFiles one after another, in the sequence specified. Each successive MessageFile modifies the CatalogFile.

If - is specified for CatalogFile, standard output (stdout) is used. If – is specified for MessageFile, standard input (stdin) is used.

gencat does not accept symbolic message identifiers. You must use mkcatdefs if you want to use symbolic message identifiers.

Extended description

The format of a message text source file is defined as follows. All characters must be encoded as single-byte characters except where noted. The fields of a message text source line are separated by a single blank character. Any other blank characters are considered as being a part of the subsequent field.

$set n comment
  Specifies the set identifier of the following messages until the next $set or end of file appears. The n denotes the set identifier, which is defined as a number in the range 1–NL_SETMAX. Set identifiers must be in ascending order within a single source file, but need not be contiguous. Any string following the set identifier is treated as a comment. If no $set directive is specified in a message text source file, all messages are located in default message set NL_SETD.

$delset n comment
  Deletes message set n from an existing message catalog. The n denotes the set number, 1–NL_SETMAX. Any string following the set number is treated as a comment.

$, comment
  A line beginning with $ followed by a blank character is treated as a comment.

m message-text
  The m denotes the message identifier, which is defined as a number in the range 1–NL_MSGMAX. Message identifiers must be in ascending order within a single set, but need not be contiguous. The length of message-text must be in the range –NL_TEXTMAX. The message text is stored in the message catalog with the set identifier that is specified by the last $set directive, and with message identifier m. If the message text is empty, and a blank character field separator is present, an empty string is stored in the message catalog. If a message source line has a message number, but not a field separator or message text, the existing message with that number (if any) is deleted from the catalog. The message text can contain double-byte characters.

$quote c
  Specifies an optional quotation mark character c, which can be used to surround message-text so trailing spaces or null (empty) messages are visible in a message source line. By default, or if an empty $quote directive is supplied, no quoting of message-text is recognized. The quotation mark character can be a double-byte character.

$timestamp
  Specifies a time stamp that can be used to identity the subsequent .cat file as having come from this file. The timestamp can be up to 20 characters long and can be any format you want. Typically it follows this format:

    $timestamp 1994 137 19:09 UTC

The mkcatdefs command automatically generates a time stamp in the file it creates for input to gencat.

Empty lines in a message text source file are ignored. Lines starting with any character other than those defined are considered invalid.

Text strings can contain the special characters and escape sequences that are defined in the following table:
### Description

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>Backspace</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage-return</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double quotation mark</td>
</tr>
<tr>
<td>\f</td>
<td>Form-feed</td>
</tr>
<tr>
<td>\t</td>
<td>Horizontal tab</td>
</tr>
<tr>
<td>\n</td>
<td>Newline</td>
</tr>
<tr>
<td>\ddd</td>
<td>Octal bit pattern</td>
</tr>
<tr>
<td>\v</td>
<td>Vertical tab</td>
</tr>
</tbody>
</table>

These sequences must be encoded as single-byte characters.

The escape sequence \ddd consists of backslash followed by one, two, or three octal digits, which are taken to specify the value of the desired character. If the character following a backslash is not one of those specified, the backslash is ignored.

A backslash (\) followed by a newline character is also used to continue a string on the following line. Thus the following two lines describe a single message string:

```plaintext
1 This line continues \ to the next line
```

which is equivalent to:

```plaintext
1 This line continues to the next line
```

### Portability of message catalogs

**gencat** works with the z/OS XL C/C++ runtime library function `catgets()` to correctly display message text in the locale that the C program using `catgets()` is running in. **gencat** adds information to the `CatalogFile` about the code set that was in effect when **gencat** processed the `CatalogFile`. **gencat** should be run with the same locale that the recipients of the messages will be using. This should be the same locale that was used to create the message text in `MessageFile`.

Message catalogs produced by **gencat** are binary-encoded, meaning that their portability cannot be guaranteed between systems. Thus, just as C programs need to be recompiled, so message catalogs must be recreated via **gencat** when moved to another system.

### Examples

To generate a test.cat catalog from the source file test.msg, enter:

```plaintext
gencat test.cat test.msg
```

### Localization

**gencat** uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_SYNTAX`
• NLSPATH


Exit values

0
Successful completion.

1
An error occurred.

Portability

X/Open Portability Guide.

getconf - Get configuration values

Format

getc onf [-a]
getc onf system_var
getc onf [-a] path_var pathname

Description

getc onf writes the value of a configuration variable to the standard output (stdout). You can specify the configuration variable using one of forms that are listed in the Format section. If you use the first form, getc onf writes the value of the variable system_var. If you use the second form, getc onf writes the value of the variable path_var for the path name given by pathname. The -a option prompts getc onf to display all current configuration variables, and their values, to stdout.

getc onf writes numeric values in decimal format and nonnumeric values as simple strings. If the value is not defined, getc onf writes the string undefined to stdout.

Options

-a
Writes out all the configuration variables for the current system, and their values, to stdout. Path variables are written based on a path name of dot (.).

Configuration variables

You can use the second form of getc onf to find the value of the following POSIX.1-1990 standard configuration variables for the specified path name:

LINK_MAX
Specifies the maximum number of links that this file can have.

MAX_CANON
Specifies the maximum number of bytes in the workstation's canonical input queue (before line editing).

MAX_INPUT
Specifies the space available in the workstation's input queue.

NAME_MAX
Specifies the largest filename size.

PATH
Specifies the standard PATH setting.
Specifies the standard PATH setting.

**PATH_MAX**
Specifies the maximum number of bytes in a path name.

**PIPE_BUF**
Specifies the largest atomic write to a pipe.

**_POSIX_CHOWN_RESTRICTED**
Specifies the restrictions that apply to file ownership changes.

**_POSIX_NO_TRUNC**
If set, it is an error for any path name component to be longer than NAME_MAX bytes.

**_POSIX_VDISABLE**
Specifies that processes are allowed to disable ending special characters.

You can use the first form of `genconf` to find the value of the following POSIX.1-1990 standard configuration variables:

**ARG_MAX**
Specifies the maximum length of arguments for running a program, including environment data.

**CHILD_MAX**
Specifies the maximum number of simultaneous processes allowed per real user.

**CLK_TCK**
Specifies the number of intervals per second in the machine clock.

**NGROUPS_MAX**
Specifies the number of simultaneous group IDs per process.

**OPEN_MAX**
Specifies the maximum number of open files at any time per process.

**STREAM_MAX**
Specifies the number of streams that one process can have open at one time.

**TZNAME_MAX**
Specifies the maximum number of bytes supported for the name of a time zone (not of the **TZ** variable).

**_POSIX_ARG_MAX**
Specifies the minimum conforming value for **ARG_MAX**.

**_POSIX_CHILD_MAX**
Specifies the minimum conforming value for **CHILD_MAX**.

**_POSIX_JOB_CONTROL**
Specifies the POSIX job control supported.

**_POSIX_LINK_MAX**
Specifies the minimum conforming value for **LINK_MAX**.

**_POSIX_MAX_CANON**
Specifies the minimum conforming value for **MAX_CANON**.

**_POSIX_MAX_INPUT**
Specifies the minimum conforming value for **MAX_INPUT**.

**_POSIX_NAME_MAX**
Specifies the minimum conforming value for **NAME_MAX**.

**_POSIX_NGROUPS_MAX**
Specifies the minimum conforming value for **NGROUPS_MAX**.

**_POSIX_OPEN_MAX**
Specifies the minimum conforming value for **OPEN_MAX**.

**_POSIX_PATH_MAX**
Specifies the minimum conforming value for **PATH_MAX**.
_POSIX_PIPE_BUF
   Specifies the minimum conforming value for PIPE_BUF.

_POSIX_SAVED_IDS
   Specifies that processes have saved set-user-ID and saved set-group-ID bits set.

_POSIX_SSIZE_MAX
   Specifies the value that can be stored in an object of type ssize_t.

_POSIX_STREAM_MAX
   Specifies the minimum conforming value for STREAM_MAX.

_POSIX_TZNAME_MAX
   Specifies the minimum conforming value for TZNAME_MAX.

_POSIX_VERSION
   Specifies the version of POSIX adhered to in this release.

You can use the first form of genconf to find the value of the POSIX.2 standard configuration variables:

BC_BASE_MAX
   Specifies the maximum ibase and obase values for the bc command.

BC_DIM_MAX
   Specifies the maximum number of elements permitted in a bc array.

BC_SCALE_MAX
   Specifies the maximum scale size allowed in bc.

BC_STRING_MAX
   Specifies the maximum number of characters in a string in bc.

COLL_WEIGHTS_MAX
   Specifies the maximum number of weights assignable to an entry of the LC_COLLATE order keyword.

EXPR_NEST_MAX
   Specifies the maximum number of expressions that you can nest inside parentheses in an expression evaluated by expr.

LINE_MAX
   Specifies the maximum number of bytes that a utility can accept as an input line (either from the standard input or a text file) when the utility takes text files as input. This number includes the trailing <newline>.

RE_DUP_MAX
   Specifies the maximum number of repeated occurrences of a regular expression when using the interval notation \{m,n\}.

   See Appendix C, “Regular expressions (regexp),” on page 945 for more information.

POSIX2_C_BIND
   Indicates if the system supports the C Language Bindings Option.

POSIX2_C_DEV
   Indicates if the system supports the C Language Development Utilities Option.

POSIX2_FORT_DEV
   Indicates if the system supports the FORTRAN Development Utilities Option.

POSIX2_FORT_RUN
   Indicates if the system supports the FORTRAN Runtime Utilities Option.

POSIX2_LOCALEDEF
   Indicates if the system supports the creation of locales.

POSIX2_SW_DEV
   Indicates if the system supports the Software Development Utilities Option.

POSIX2_CHAR_TERM
   Indicates if the system supports at least one terminal type capable of all operations necessary for the User Portability Utilities Option. This parameter name is correct only on if POSIX2_UPE is on.
POSIX2_UPE
   Indicates if the system supports the User Portability Utilities Option.

POSIX2_VERSION
   Specifies the version of POSIX.2 adhered to in this release.

POSIX2_BC_BASE_MAX
   Specifies the minimum conforming value for BC_BASE_MAX.

POSIX2_BC_DIM_MAX
   Specifies the minimum conforming value for BC_DIM_MAX.

POSIX2_BC_SCALE_MAX
   Specifies the minimum conforming value for BC_SCALE_MAX.

POSIX2_BC_STRING_MAX
   Specifies the minimum conforming value for BC_STRING_MAX.

POSIX2_COLL_WEIGHTS_MAX
   Specifies the minimum conforming value for EQUIV_CLASS_MAX.

POSIX2_EXPR_NEST_MAX
   Specifies the minimum conforming value for EXPR_NEST_MAX.

POSIX2_LINE_MAX
   Specifies the minimum conforming value for LINE_MAX.

POSIX2_RE_DUP_MAX
   Specifies the minimum conforming value for RE_DUP_MAX.

You can use the third form of `genconf` to find the value of the POSIX.2 standard configuration variables:

ACL
   Specifies that access control lists (ACLs) are supported by the security product and file system.

_PC_ACL_ENTRIES_MAX
   Specifies the maximum number of extended ACL entries that can be placed in an access control list for the specified file.

This implementation of `genconf` also supports the following non-POSIX-conforming name:

CS_SHELL
   Specifies the default shell (command interpreter).

PC_ACL
   Security product supports access control lists (ACLs).

_PC_ACL_ENTRIES_MAX
   Maximum number of entries that can be placed in an access control list for a specified file.

Examples
This example uses `genconf` to find the minimum conforming value for PATH_MAX, which is contained in the variable _POSIX_PATH_MAX. If you issue

```
getconf _POSIX_PATH_MAX
```

`genconf` displays

```
255
```

Localization
`genconf` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
Exit values

0
The specified parameter_name was valid and **genconf** displayed its value successfully.

>0
An error occurred.

Portability

POSIX.2, X/Open Portability Guide.

_CS_SHELL is an extension of the POSIX standard. Some symbols are supported only on systems that support POSIX.2.

Related information

bc, expr, sh

See Appendix C, “Regular expressions (regexp),” on page 945 for more information about regexp.

---

**getfacl - Display owner, group, and access control list (ACL) entries**

Format

```
getfacl [-acdfhmoqs] [-e user] file ...
```

Description

**getfacl** displays the comment header, base ACL (access control list) entries, and extended ACL entries, if there are any, for each file that is specified. It also resolves symbolic links. You can specify whether to display access, file default, or directory default. You can also change the default display format. The output can be used as input to **setfacl**.

A description of access control list entries can be found in **z/OS UNIX System Services Planning**.

Options

- **-a**
  Displays the access ACL entries. This is the default if -a, -d, or -f is not specified.

- **-c**
  Displays each ACL entry, using commas to separate the ACL entries instead of newlines, which is the default. Does not display the header.

- **-d**
  Displays the directory default ACL entries. If the file is not a directory, a warning is issued.

- **-e user**
  Displays only the ACL entries for the specified types of access control lists (-a, -d, -f) which affects the specified user's access. If users look at the output, they may be able to determine why the access is granted or denied. The user can be an UID or user name. The output includes the user's entry, if it exists, as well as entries for any group to which the user is connected.

- **-f**
  Displays the file default ACL entries. If the file is not a directory, a warning is issued.

- **-h**
  Does not resolve the symbolic link. (ACLs are not allowed on symbolic links, so the file will not have anything displayed.)
getfacl

-m
  Specifies that the comment header (the first three lines of each file's output) is not to be displayed.

-o
  Displays only the extended ACL entries. Does not display the base ACL entries.

-q
  Quiet mode. Suppresses the warning messages and gives a successful return code if there are no other errors.

-s
  Skips files that only have the base ACL entries (such as owner, group, other). Only files that have the extended ACL entries are displayed.

Examples

1. To display access ACL information for file file, issue:

   getfacl file

   Where the following is a sample of the output:

   #file: file
   #owner: WELLIE
   #group: SYS
   user::rwx  <=== The owner's permission bit setting
   group::rwx  <=== The group's permission bit setting
   other::rw-  <=== Permission bit setting if neither user nor group
   user: WELLIE2: rw-
   group:SYS1:rwx

2. To display access, file default, and directory default ACL information for directory directory, issue:

   getfacl -a -f -d directory

   Where the following is a sample of the output:

   #file: file
   #owner: WELLIE
   #group: SYS
   user::rwx
   group::rwx
   other::rw-
   user: WELLIE2: rw-
   group:SYS1:rwx
   fdefault:user: WELLIE2: rw-
   fdefault:group:SYS1:rwx
   default:user:WELLIE4:---

3. To copy the ACL entries from file foo such that the file bar will have the same ACL entries:

   getfacl foo | setfacl -S - bar

Localization

getfacl uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_SYNTAX
• NLSPATH

Exit values

0
  Successful completion

>0
  Failure

getfacl displays the ACL entries in the following order: access, file default, and directory default. Errors will occur in the following situations:

- If a file is not a directory and the -d or -f option was used, you will get a warning and getfacl will continue to the next file.
- If the user does not have access to a file, you will get a warning and getfacl will continue to the next file.

Portability

An approved POSIX standard does not exist for getfacl.

Related information

find, ls, setfacl

getopts - Parse utility options

Format

getopts  opstring name [arg ...]

Description

getopts obtains options and their arguments from a list of parameters that follows the standard POSIX.2POSIX.2 option syntax (that is, single letters preceded by a hyphen (-) and possibly followed by an argument value). Typically, shell scripts use getopts to parse arguments passed to them. When you specify arguments with the arg argument on the getopts command line, getopts parses those arguments instead of the script command-line arguments (see set).

Options

opstring
  Gives all the option letters that the script recognizes. For example, if the script recognizes -a, -f, and s, opstring is afs. If you want an option letter to be followed by an argument value or group of values, put a colon after the letter, as in a:fs. This indicates that getopts expects the -a option to have the form -a value. Normally one or more blanks separate value from the option letter; however, getopts also handles values that follow the letter immediately, as in -a value. opstring cannot contain a question mark (?) character.

name
  Specifies the name of a shell variable. Each time you invoke getopts, it obtains the next option from the positional parameters and places the option letter in the shell variable name.

  getopts places a question mark (?) in name if it finds an option that does not appear in opstring, or if an option value is missing.

arg ...
  Each option on the script command line has a numeric index. The first option found has an index of 1, the second has an index of 2, and so on. When getopts obtains an option from the script command line, it stores the index of the script in the shell variable OPTIND.
When an option letter has a following argument (indicated with \( : \) in \opstring\), \getopts\ stores the argument as a string in the shell variable \OPTARG\. If an option doesn't take an argument, or if \getopts\ expects an argument but doesn't find one, \getopts\ unsets \OPTARG\.

When \getopts\ reaches the end of the options, it exits with a status value of 1. It also sets \name\ to the character \?\ and sets \OPTIND\ to the index of the first argument after the options. \getopts\ recognizes the end of the options by any of the following situations:

- Finding an argument that doesn't start with \( - \)
- Finding the special argument \( -- \), marking the end of options
- Encountering an error (for example, an unrecognized option letter)

\OPTIND\ and \OPTARG\ are local to the shell script. If you want to export them, you must do so explicitly. If the script invoking \getopts\ sets \OPTIND\ to 1, it can call \getopts\ again with a new set of parameters, either the current positional parameters or new \arg\ values.

By default, \getopts\ issues an error message if it finds an unrecognized option or some other error. If you do not want such messages printed, specify a colon as the first character in \opstring\.

**Examples**

Following is an example of using \getopts\ in a shell script:

```bash
# Example illustrating use of getopts builtin. This
# shell script would implement the paste command,
# using getopts to process options, if the underlying
# functionality was embedded in hypothetical utilities
# hpaste and vpaste, which perform horizontal and
# vertical pasting respectively.
# paste=vpaste    # default is vertical pasting
# seplist=" "    # default separator is tab

while getopts d:s o
do      case "$o" in
  d)      seplist="$OPTARG";;
  s)      paste=hpaste;;
  [?]     print >&2 "Usage: $0 [-s] [-d seplist] file ...
        exit 1;;
  esac
done

# perform actual paste command
$paste -d "$seplist" "$@"
```

**Environment variables**

\getopts\ uses the following environment variables:

- **\OPTARG\**: Stores the value of the option argument found by \getopts\.
- **\OPTIND\**: Contains the index of the next argument to be processed.

**Localization**

\getopts\ uses the following localization environment variables:

- **\LANG\**
- **\LC_ALL\**
- **\LC_CTYPE\**
- **\LC_MESSAGES\**

Usage notes

`getopts` is a built-in shell command.

Exit values

0  
`getopts` found a script command line with the form of an option. This happens whether or not it recognizes the option.
1  
`getopts` reached the end of the options, or an error occurred.
2  
Failure because of an incorrect command-line option

Portability

POSIX.2, X/Open Portability Guide.
On UNIX systems, `getopts` is built in both the KornShell and Bourne shell.

Related information

sh

grep - Search a file for a specified pattern

Format

```
grep [-E|-F] [] [-A|-P|-C num] [-S string]
    [-W option[option] ... [-e pattern] ... [-f patternfile] ... [pattern] [file ... file ...]

egrep [-BNbcilnsvx] [-A|-P|-C num] [-Sstring]
    [-W option[option] ... [-e pattern] ... [-f patternfile] ... [pattern] [file ... file ...]

fgrep [-BNbcilnsvx] [-A|-P|-C num] [-Sstring]
    [-W option[option] ... [-e pattern] ... [-f patternfile] ... [pattern] [file ... file ...]
```

Description

`fgrep` searches files for one or more `pattern` arguments. It does not use regular expressions; instead, it uses fixed string comparisons to find matching lines of text in the input.

`egrep` works in a similar way, but uses extended regular expression matching. (For information about regular expression matching, see Appendix C, “Regular expressions (regexp),” on page 945.) If you include special characters in patterns typed on the command line, escape them by enclosing them in apostrophes to prevent inadvertent misinterpretation by the shell or command interpreter. To match a character that is special to `egrep`, put a backslash (\) in front of the character. It is usually simpler to use `fgrep` when you do not need special pattern matching.

`grep` is a combination of `fgrep` and `egrep`. If you do not specify either `-E` or `-F`, `grep` behaves like `egrep`, but matches basic regular expressions instead of extended ones. You can specify a pattern to search with either the `-e` or `-f` option. If you do not specify either option, `grep` (or `egrep` or `fgrep`) takes the first non-option argument as the pattern for which to search. If `grep` finds a line that matches a pattern, it displays the entire line. If you specify multiple input files, the name of the current file precedes each output line.

If you do not specify any files, the command reads from the standard input (stdin).

Options

`grep` accepts all the following options while `egrep` and `fgrep` accept all but the `-E` and `-F` options.
-A num
   Displays num lines of trailing context after the lines are matched.

-B
   Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-b
   Precedes each matched line with its file block number.

-C num
   Displays num lines of leading and trailing output context.

-c
   Displays only a count of the number of matched lines and not the lines themselves.

-E
   Matches using extended regular expressions (causes grep to behave like egrep).

-e pattern
   Specifies one or more patterns that are separated by newlines for which grep is to search.
   You can indicate each pattern with a separate -e option character, or with newlines within pattern. For example, the following two commands are equivalent:

   ```bash
   grep -e pattern_one -e pattern_two file
   grep -e 'pattern_one' pattern_two' file
   ```

-F
   Matches using fixed strings (causes grep to behave like fgrep).

-f patternfile
   Reads one or more patterns from patternfile. Patterns in patternfile are separated by newlines.

-i
   Ignores the case of the strings that are being matched.

-l
   Lists only the file names that contain the matching lines.

-N
   When -A, -C or -P have been specified, does not print a separator between groups of lines.

-n
   Precedes each matched line with its fileline number.

-P num
   Displays num lines of leading context before matching lines.

-q
   Suppresses output and returns the appropriate return code.

-S string
   When -A, -C or -P have been specified, displays string instead of "--" between groups of lines.

-s
   Suppresses the display of any error messages for nonexistent or unreadable files.

-v
   Complements the sense of the match. That is, all lines that do not match a pattern are displayed.

-W option[,...]
   Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

   filecodeset=codeset
       Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command icnv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.
If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

If you are specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

\textbf{pgmcodeset=\textit{codeset}}

Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is \textit{codeset}. \textit{codeset} can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command \texttt{iconv -l} lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is turned off for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

\textbf{Restriction:} The only supported values for pgmcodeset are IBM-1047 and 1047.

\textbf{-x}

Requires a string to match an entire line.

\textbf{Examples}

1. To display every line that mentions an astrological element:
   \begin{verbatim}
   egrep "earth|air|fire|water" astro.log
   \end{verbatim}

2. To display every line in a text file that contains the string ".com":
   \begin{verbatim}
   fgrep ".com" myTextFile
   \end{verbatim}

3. To display every line in a text file that contains the string ".com" or ".org":
   \begin{verbatim}
   egrep \"\.[a-zA-Z0-9]nc\" myTextFile
   \end{verbatim}

   In this example, the period (\.) is escaped by the \ character to ensure that the period is not treated as a special character. Otherwise, any string with the letters "com" or "org" preceded by any character would match (for example, lines that contain the strings "become" or "forge").

4. To display every line in a text file that starts with "Hello" or "hello":
   \begin{verbatim}
   grep '^\[Hh\]ello' myTextFile
   \end{verbatim}

5. To display every line in a text file that contains ASCII characters that do not start with the # character, assuming that:
   \begin{itemize}
   \item The text file is untagged and you do not want to tag it or enable automatic conversion, and
   \end{itemize}
You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file):

```
grep -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 -v '^#' myAsciiFile
```

6. To display every line in a text file that contains EBCDIC characters having the string "path", regardless of case and assuming that automatic conversion was enabled but the text file is incorrectly tagged as UTF-8:

```
grep -Bi 'path' myMisTaggedFile
```

In this example, lines that contain strings such as "path", "Path", "PATH", and "pAtH" would match.

7. To display the lines either before or after the matching lines:

   - To print 10 lines after the matching lines:
     
     ```
     grep -A 10 'hello' myTextFile
     ```

   - To print 10 lines before the matching lines:
     
     ```
     grep -P 10 'hello' myTextFile
     ```

   - To print 10 lines before and after the matching lines, using group separator "**" instead of "--":
     
     ```
     grep -C 10 -S "**" 'hello' myTextFile
     ```

Localization

`grep` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_COLLATE`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_SYNTAX`
- `NLSPATH`


Environment variables

`grep` uses the following environment variable:

_/TEXT_CONV_

Contains text conversion information for the command. The text conversion information is not used when either the -B option or the `filecodeset` or `pgmcodeset` option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

Exit values

0

At least one match for `pattern` was found.

1

No matches for `pattern` were found.

2

Failure due to any of the following conditions:

- The -e option was missing `pattern`.
• The -f option was missing *patternfile*.
• Out of memory for input or to hold a pattern.
• *patternfile* could not be opened.
• Incorrect regular expression.
• Incorrect command-line option.
• The command line had too few arguments.
• The input file could not be opened.
• The code set is not valid.
• Could not turn off automatic conversion.
• Could not perform requested text conversion.

If the program fails to open one input file, it tries to go on to look at any remaining input files, but it returns 2 even if it succeeds in finding matches in other input files.

**Messages**

Possible error messages include:

**input lines truncated—result questionable**
One or more input lines were longer than *grep* could handle; the line has been truncated or split into two lines. Shorten the line or lines, if possible. This message does not affect the exit status.

**out of space for pattern string**
*grep* did not have enough memory available to store the code needed to work with the given pattern (regular expression). The pattern might be too complex. Make the pattern simpler, or try to release memory so that *grep* has more space to work with.

**Limits**

The longest input record (line) is restricted by the system variable LINE_MAX. It is always at least 2048 bytes. *fgrep* might be able to handle lines longer than LINE_MAX. Longer lines are treated as two or more records.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

Only the *grep* command is part of the POSIX and X/Open standards. The *egrep* and *fgrep* commands are extensions. The -B, -b, and -W options are extensions of the POSIX standard.

**Related information**

*ed, find*

See Appendix C, “Regular expressions (regexp),” on page 945 for more information about regexp.

**hash - Create a tracked alias**

**Format**

```
hash [name ...]
hash -r
```

**Description**

*hash* creates one or more tracked aliases. Each *name* on the command line becomes an alias that is resolved to its full path name; thus the shell avoids searching the PATH directories for the command.
whenever you issue it. A tracked alias is assigned its full path name the first time that the alias is used. It is reassigned a path name the first time that it is used after the variable PATH is changed or the shell command cd is used.

**hash** is a built-in alias that is defined with

```
alias hash='alias -t'
```

If you specify hash without any arguments on the command line, **hash** displays the current list of tracked aliases.

**Options**

- `-r`
  Removes all current tracked aliases.

**Usage notes**

**hash** is a built-in shell command.

**Localization**

**hash** uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

- `0`
  Successful completion.
- `1`
  Failure because of an incorrect command-line option.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

**Related information**

alias, sh

---

**head - Display the first part of a file**

**Format**

```
head [-B] [-W option[option] ... [-b|-c|-k|-l|-m|-n num] [file ... file ...]
head [-B] [-W option[option] ... [-num] [file ... file ...]
```

**Description**

By default, **head** displays the first 10 lines of each file given on the command line. If you do not specify `file`, **head** reads standard input (stdin).
Options
-B
  Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-b num
  Displays the first num blocks (a block is 512 bytes) of each file.

-c num
  Displays the first num bytes of each file.

-k num
  Displays the first num kilobytes (1024 bytes) of each file.

-l num
  Displays the first num lines of each file.

-m num
  Displays the first num megabytes of each file.

-n num
  Displays the first num lines of each file.

-# num
  Displays the first num lines of each file.

-W option[,...]
  Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

  filecodeset=codeset
  Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any tag. If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

  If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

  When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User's Guide and Reference.

  pgmcodeset=codeset
  Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

  If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.
If `filecodeset` or `pgmcodeset` is specified, then automatic conversion is disabled for this command invocation and the `-B` option is ignored if it is also specified. See *z/OS UNIX System Services Planning* for more information about automatic conversion.

The only supported values for `pgmcodeset` are IBM-1047 and 1047.

**Examples**

1. To display the first 10 lines of a text file to the standard output (stdout):
   ```
   head myTextFile
   ```

2. To display the first 400 bytes of a text file to the standard output (stdout):
   ```
   head -c 400 myTextFile
   ```

3. To display the first 50 lines of a text file containing UTF-8 characters to the standard output (stdout), assuming that:
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file):
   ```
   head -n 50 -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File
   ```

4. To display the first 10 lines of two text files containing EBCDIC characters to the standard output (stdout), assuming that automatic conversion has been enabled but the text files are incorrectly tagged as ASCII:
   ```
   head -B myMisTaggedFile01 myMisTaggedFile02
   ```

**Localization**

*head* uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Environment variables**

*head* uses the following environment variable:

- `_TEXT_CONV`

  Contains text conversion information for the command. The text conversion information is not used when either the `-B` option or the `filecodeset` or `pgmcodeset` option `-W` option is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

**Exit values**

0  
  Successful completion.

1  
  Failure due to any of the following reasons:
  - Inability to open an input file.
• Read error on standard input (stein).
• Write error on standard output (stdout).
• The code set is not valid.
• Could not turn off automatic conversion.
• Could not perform requested text conversion.

Failure due to any of the following:
• Unknown command-line option.
• Missing or incorrect num in an -n option.

Messages
Possible error messages include:

**Badly formed line or character count num**
The value num, following a -b, -c, -k, -l, m, or -n option, was not a valid number.

Portability
POSIX.2, X/Open Portability Guide.
This program originated with Berkeley Software Distribution (BSD) and is a frequent add-on to UNIX systems.
The POSIX standard included only the -n num and -num options, though it considers the latter obsolete.

Related information
cat, sed, tail

**history - Display a command history list**

**Format**

history [first[last]]
tcsh shell:
history [-hTr][n]
history -S|-L|-M [filename]
history -c

**Description**

history is an alias for fc -l. Like fc -l, history displays the list of commands that have been input to an interactive shell. This command does not edit or reenter the commands. If you omit last, history displays all commands from the one indicated by first through to the previous command entered. If you omit both first and last with this command, the default command range is the 16 most recently entered commands.

See “fc - Process a command history list” on page 302 for more information.

For the tcsh shell, history, when used alone, prints the history event list. If n is given only the n most recent events are printed or saved.

**Note:** See “tcsh: Invoke a C shell” on page 671 for descriptions of the tcsh shell variables and commands.

The tcsh shell history built-in command uses the following options:
With -h, the history list is printed without leading numbers.

With -T, timestamps are printed also in comment form. (This can be used to produce files suitable for loading with history -L or source -h.)

With -r, the order of printing is most recent first rather than oldest first.

With -S, history saves the history list to filename. If the first word of the savehist shell variable is set to a number, at most that many lines are saved. If the second word of savehist is set to merge, the history list is merged with the existing history file instead of replacing it (if there is one) and sorted by time stamp. Merging is intended for an environment like the X Window System with several shells in simultaneous use. Currently it only succeeds when the shells quit one after another.

With -L, the shell appends filename, which is presumably a history list saved by the -S option or the savehist mechanism, to the history list. -M is like -L, but the contents of filename are merged into the history list and sorted by timestamp. In either case, histfile is used if filename is not given and ~/.history is used if histfile is unset. history -L is exactly like source -h except that it does not require a filename.

With -c, clears the history list.

tcsh login shells do the equivalent of history -L on startup and, if savehist is set, history -S before exiting. Because only ~/.tcshrc is normally sourced before ~/.history, histfile should be set in ~/.tcshrc rather than ~/.login.

If histlit is set, the first form (history [-hTr] [n]) and second form (history -S | -L | -M [filename]) print and save the literal (unexpanded) form of the history list.

Related information
fc, sh, tcsh

iconv - Convert characters from one code set to another

Format

```bash
iconv [-cs] [-B] [-M|-T] -f oldset -t newset [file ... file ...]
iconv [-cs] [-B] [-M|-T] -F [-f oldset] -t newset [file ... file ...]
iconv -l [-v]
```

Description

iconv converts characters in file (or from standard input if no file is specified) from one code page set to another. The converted text is written to standard output (stdout). See z/OS XL C/C++ Programming Guide for more information about the code sets supported for this command.

If the input contains a character that is not valid in the source code set, iconv replaces it with the byte 0xff and continues, unless the -c option is specified.

If the input contains a character that is not valid in the destination code set, behavior depends on the system's iconv() function. See z/OS XL C/C++ Runtime Library Reference for more information about the character that is used for converting incorrect characters.

Also, z/OS XL C/C++ Programming Guide has a list of code pages that are supported by the z/OS shell.

You can use iconv to convert single-byte data or double-byte data.

Options

-B
The byte order mark (BOM) at the beginning of the Unicode data is removed during the conversion.

-c
Characters containing conversion errors are not written to the output. By default, characters not in the source character set are converted to the value 0xff and written to the output.
-f oldset
Specifies the source code set of the input. oldset can be either the code set name that is known to the system, the numeric coded character set identifier (CCSID), or a path name to a file that contains an external code set.

-F
Uses the input file's coded character set (as defined in the file tag) as the source code set. If -f is also specified, and the oldset matches the file tag or if no file tag code is set, then oldset is used as the source code set. If -F and -f are specified and oldset does not match the file tag code set, then iconv fails with an error.

-I
Lists supported code sets and CCSIDs. (This option was accepted in releases before V1R3, but was not supported.)

-M
Tags a new output file as mixed. The text flag (txtflag) will be off and the value for code set will be the same as the one that is specified on the -t option.

-s
Suppresses all error messages about faulty encodings.

-t newset
Specifies the destination code set for the output. newset can be either the code set name that is known to the system, the numeric coded character set identifier (CCSID), or a path name to a file containing an external code set.

-T
Tag a new output file as text. The txtflag will be on and the value for code set will be the same as the one that is specified on the -t option.

-v
Specifies verbose output.

For information about file tagging and code set specifications, see z/OS UNIX System Services Planning.

Examples
1. To convert the file words.txt from the IBM-1047 standard code set to the ISO 8859-1 standard code set and store it in converted:

   iconv -f IBM-1047 -t ISO8859-1 words.txt > converted

   Also, for the exact conversion table names, refer to z/OS XL C/C++ Programming Guide.

2. To convert the file mbcsdata, which is in code page IBM-932 (double-byte ASCII), to code page IBM-939 and put the output in a file called dbcsdata:

   iconv -f IBM-932 -t IBM-939 mbcsdata > dbcsdata

Localization
iconv uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

Exit values

0
  Successful completion.

1
  Failure due to any of the following reasons:
  • Insufficient memory.
  • Inability to open the input file.
  • Incorrect or unknown option.

2
  Input contained a character sequence that is not permitted in the source code set

Portability

X/Open Portability Guide.

-v is an extension to the POSIX.2 standard. The -c, -l, and -s options are extensions to the XPG standard.

id - Return the user identity

Format

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id [user]</td>
<td>id -G [-n] [user]</td>
</tr>
</tbody>
</table>

Description

id displays the user name and group affiliations of the user who issued the command. Specifying a user argument on the command line displays the same information for the given user instead of the person invoking id. In this case, you require appropriate permissions.

The output has the format:

uid=runum(username) gid=rgnum(groupname)

where runum is the user's real user ID (UID) number, username is the user's real user name, rgnum is the user's real group ID (GID) number, and groupname is the user's real group name.

A user's real and effective IDs may differ. In this case, there might be separate entries for effective user ID (UID) with the format:

euid=eunum(euname)

where eunum is the effective user ID number and euname is the effective user name. An entry for effective group ID has the format:

egid=egnum(egname)

where egnum is the effective group ID number and egname is the effective group name.

If a user is a member of other supplemental groups, these are listed at the end of the output, with this format:

groups=gnum(groupname)
where gnum is the user's supplemental group ID number and groupname is the user's supplemental group name.

id may also display the multilevel security label for the user's current address space. See z/OS Planning for Multilevel Security and the Common Criteria for more information about multilevel security.

Options
- `G`
  Displays all different group IDs (effective, real, and supplementary) as numbers separated by spaces.
- `g`
  Displays only the effective group ID number.
- `M`
  Displays the multilevel security label for the user's current address space. See z/OS Planning for Multilevel Security and the Common Criteria for more information about multilevel security.
- `n`
  With -G, -g, or –u, displays the name rather than the number.
- `r`
  With -g or –u, displays the real ID rather than the effective one.
- `u`
  Displays only the effective user ID number.

Localization
id uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_NUMERIC
- NLSPATH


Examples

```
> id -M
SYSHIGH
```

Usage notes
See z/OS Planning for Multilevel Security and the Common Criteria for more information about multilevel security and seclabels.

Exit values

- 0
  Successful completion
- 1
  You specified an incorrect user with the -u option
- 2
  Failure due to an incorrect command-line argument
inetd daemon - Provide service management for networks

Format
inetd [-d] [configuration file]

Description
The inetd daemon provides service management for a network. For example, it starts the rlogind program whenever there is a remote login request from a workstation.

The rlogind program is the server for the remote login command rlogin commonly found on UNIX systems. It validates the remote login request and verifies the password or password phrase of the target user. It starts a z/OS shell for the user and handles translation between ASCII and EBCDIC code pages as data flows between the workstation and the shell.

When inetd is running and receives a request for a connection, it processes that request for the program associated with that socket. For example, if a user tries to log in from a remote system into the z/OS shell while inetd is running, inetd processes the request for connection and then issues a fork() and execel() to the rlogin program to process the rlogin request. It then goes back to monitoring for further requests for those applications that can be found as defined in the /etc/inetd.conf file.

Options
-d
  Specifies that the inetd daemon be started in debug mode. All debug messages are written to standard error.

configuration file
  Specifies that the inetd daemon be started with a configuration file other than the default /etc/inetd.conf file.

Signals
inetd recognizes the following signals:
SIGTERM
  Terminates inetd in an ordinary fashion and deletes /etc/inetd.pid. You can restart inetd, if you want.
SIGINT
  Same as SIGTERM.
SIGHUP
  Rereads the inetd configuration file. This can be used to start new services, or to restart services with a different port.

Other signals that normally end a process (such as SIGQUIT or SIGKILL) should not normally be sent to inetd because the program will not have a chance to remove /etc/init.pid.

Usage notes
  1. Buffer sizes should only be specified if the documentation for the daemon being specified in the inetd.conf statement calls for something other than the default.
2. The configuration file is field-sensitive, but not column-sensitive. Fields must be arranged in the order shown in Table 15 on page 347. Continuation lines for an entry must begin with a space or tab. Each entry must contain all fields. The inetd daemon uses the configuration file entry to properly set up the environment expected by the server. Specifying an incorrect value for one or more of the parameters is likely to cause the server to fail.

Table 15: Fields in the configuration file (inetd daemon)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ip_address:]service_name</td>
<td>ip_address is a local IP, followed by a colon. If specified, the address is used instead of INADDR_ANY or the current default. To specifically request INADDR_ANY, use &quot;<em>:</em>&quot;. If ip_address (or a colon) is specified, without any other entries on the line, it becomes the default for subsequent lines until a new default is specified. service_name is a well-known service name such as login or shell. The name and protocol specified must match one of the server names defined in /etc/services. For more information about /etc/services, see z/OS Communications Server: IP Configuration Reference and z/OS Communications Server: New Function Summary.</td>
</tr>
<tr>
<td>socket_type</td>
<td>Stream or dgram</td>
</tr>
<tr>
<td>protocol [,sndbuf=n],[rcvbuf=n]</td>
<td>protocol can be tcp or udp, or (for IPv6) tcp6 or udp6. tcp4 and udp4 can also be specified to explicitly request IPv4. The protocol is used to further qualify the service name. Both the service name and the protocol must match an entry in /etc/services, except that, the 4 or the 6 should not be included in the /etc/services entry. For more information about /etc/services, see z/OS Communications Server: IP Configuration Reference and z/OS Communications Server: New Function Summary. If tcp6 or udp6 is specified, the socket supports IPv6 (that is, AF_INET6 is used.) sndbuf and rcvbuf specify the size of the send and receive buffers. The size can be in bytes, or a k or m can be added to indicate kilobytes or megabytes respectively. sndbug and rcvbuf can be used in either order.</td>
</tr>
<tr>
<td>wait_flag [.max]</td>
<td>Wait or nowait. Wait indicates that the daemon is single-threaded and another request will not be serviced until the first one completes. If nowait is specified, the inetd daemon issues an accept when a connect request is received on a stream socket. If wait is specified, the inetd daemon does not issue the accept. It is the responsibility of the server to issue the accept if this is a stream socket. max is the maximum number of requests that are allowed in a 60-second interval. The default is 40. If the maximum number is exceeded, the service's port is shut down.</td>
</tr>
</tbody>
</table>
Table 15: Fields in the configuration file (inetd daemon) (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>login_name</td>
<td>User ID and group that the forked daemon is to execute under. inetd can run a program with a UID that is not 0. However, if the program that inetd runs needs to change the identity of the process to that of the user, then the login_name must have been defined to RACF via ADDUSER as a superuser with a UID of 0 (UID 0) and the login_name must have been defined to RACF. This will allow inetd to use special functions like setgid() and setuid(). If the program that will be invoked by inetd requires the use of special functions like setuid() and seteuid(), then it must be permitted to the BPX.DAEMON class as in the following example for login, which is a typical ADDUSER command. ADDUSER rlogind omvs(uid(0) home(/) A typical permit command is: permit bpx.daemon class(facility) id(rlogind) access(read) How you set up security for daemons is the final determining factor. For more information, see the topic on Establishing the correct level of security for daemons in z/OS UNIX System Services Planning.</td>
</tr>
<tr>
<td>server_program</td>
<td>Full path name of the service. For example: /usr/sbin/rlogind is the full path name for the rlogind command.</td>
</tr>
<tr>
<td>server_arguments</td>
<td>Maximum of 20 arguments. The first argument is the server name and must be provided. Additional arguments are optional.</td>
</tr>
</tbody>
</table>

Related information
The inetd daemon creates a temporary file, /etc/inetd.pid, that contains the PID of the currently executing inetd daemon. This PID value is used to identify syslog records that originated from the inetd daemon process, and also to provide the PID value for commands such as kill that require you to specify a PID, and to provide a lock to prevent more than one inetd from being active at one time.

For more information about setting up the inetd configuration file and configuring daemons in general, see the topic on daemons in z/OS UNIX System Services Planning or z/OS Communications Server: IP Configuration Reference.

infocmp - Compare or print the terminal description

Format
infocmp [-ducn] [-ILC] [-1Vv] [-s d||l|c] [-A directory] [-B directory] [term_names...]

Description
infocmp compares terminfo database entries, or prints a terminfo database entry. Output is written to standard output.

The Curses application uses the terminfo database, which contains a list of terminal descriptions. This enables you to manipulate a terminal’s display regardless of the terminal type. To create the terminfo
database, use tic. For information about customizing the terminfo database, see z/OS UNIX System Services Planning.

For more information about curses, see z/OS C Curses.

**Options**

- **d**
  Prints the two terminal definitions showing the differences between the capabilities.

- **u**
  Prints the differences between the two terminal definitions.

- **c**
  Prints entries that are common to the two terminfo databases.

- **n**
  Does not print entries in either terminfo database.

- **I**
  Prints the current terminal description using capname. (*capname* is the short name for a capability specified in the terminfo source file.)

- **C**
  Prints the current terminal description using termcap.

- **L**
  Prints the current terminal description using variables (names that the curses functions can use when working with the terminfo database)

- **1**
  Single-column output.

- **V**
  Prints the program version.

- **v**
  Prints debugging information (verbose) to standard error

- **s**
  Changes sort order of the fields printed.

- **d**
  Sorts by database

- **i**
  Sorts by terminfo

- **c**
  Sorts by termcap

- **l**
  Sorts by the variables (names that the curses function can use when working with the terminfo database)

- **A**
  First terminfo database.

- **B**
  Other terminfo database.

**term_names**

Names of entries to be processed.

**Usage notes**

When displaying terminal database information for entries that are to be processed, infocmp operates as follows:
1. If you omit term_names, infocmp locates the terminal database information specified by the TERM environment variable and displays that as the entry’s terminal database information.

2. If you specify a single term_name, infocmp displays terminal database information for that named entry.

3. If you specify more than one term_name, infocmp displays the results of a terminal database comparison between all of the specified term_names.

Examples

1. To print out the current terminal description using capname, issue:

```
infocmp
```

You will see:

```
infocmp ibm3101
#    Reconstructed via infocmp from file:/usr/share/lib/terminfo/i
ibm3101|IBM 3101-10:
ibm3101|IBM 3101-10,
am,
cols#80, lines#24,
bel=¬?, clear=¬.\322, cr=\r, cub1=\b, cud1=\n,
cuf1=¬.\303,
```

2. To print out the current terminal description using the curses capability names, issue:

```
infocmp -L
```

You will get:

```
infocmp -L ibm3101
#    Reconstructed via infocmp from file:/usr/share/lib/terminfo/i
ibm3101|IBM 3101-10:
Terminal type ibm3101
ibm3101|IBM 3101-10
flags
 auto_right_margin,
numbers
columns = 80, lines = 24,
strings
 bell = '¬?', carriage_return = '', clear_all_tabs = '¬.\310',
clear_screen = '¬.\322', clr_eol = '¬.\311', clr_eos = '¬.\321',
```

3. To print out the current terminal description using capname, issue:

```
infocmp -I
```

You will get:

```
infocmp -I ibm3101
#    Reconstructed via infocmp from file:/usr/share/lib/terminfo/i
ibm3101|IBM 3101-10:
ibm3101|IBM 3101-10,
am,
cols#80, lines#24,
bel=¬?, clear=¬.\322, cr=\r, cub1=\b, cud1=\n,
cuf1=¬.\303,
```

4. To print out the current terminal description using termcap, issue:

```
infocmp -C
```
You will get:

```
infocmp -C ibm3101
# Reconstructed via infocmp from file:/usr/share/lib/terminfo/i
ibm3101|IBM 3101-10:|
   :am:bs:
   :co#80:li#24:kn#2:
   :cd=\EJ.:ce=\EI.:cl=\EK:|
   :c:ce=\EY%p1%' '%+%c%p2%'
   '%+%c:ct=\EH:.ho=\EH:|
   :nd=\EC:.st=\E0:.up=\EA:
```

5. To print entries in single-column format, issue:

```
infocmp -1
```

You will get:

```
infocmp -C -1 ibm3101
# Reconstructed via infocmp from file:/usr/share/lib/terminfo/i
ibm3101|IBM 3101-10:|
   :am:bs:
   :co#80:li#24:kn#2:
   :cd=\EJ:|
   :ce=\EI:|
   :cl=\EK:
```

6. To print the two terminal definitions showing the difference between the capabilities (F indicates False, entry not present; T indicates True, entry present):

```
infocmp -d ibm3101 hft--c
```

You will get:

```
infocmp -d ibm3101 hft--c
# Reconstructed via infocmp from file:/usr/share/lib/terminfo/i
ibm3101|IBM 3101-10:|
   )bw:F:T.
   )msgr:F:T.
   )xon:F:T.
   )comparing booleans.
   )it:-1:8.
   )lines:24:25.
   )comparing numbers.
   )batt1:'NULL','\206\361'.
   )batt2:'NULL','\206\361\224\204'.
```

To print the capabilities that are different between the two terminal definitions. The values for the first terminal definitions are shown.

```
infocmp -u ibm3101 hft--c
```

You will get:
7. To print the capabilities that are the same in both terminal definitions, issue:

```
infocmp -c ibm3101 hft-c
```

You will get:

```
comparing ibm3101 to hft-c.
comparing booleans.
am= T.
comparing numbers.
cols= 80.
comparing strings.
bel='¬?',
clr='\r',
cub1='\b',
cud1='\n'.
```

8. To print the capabilities that are not found in either terminal definition, issue:

```
infocmp -u ibm3101 hft-c
```

You will get:

```
comparing ibm3101 to hft-c.
comparing booleans.
bce.
!ccc.
!chts.
!cpix.
!crxm.
!da.
!daisy.
```

Environment variables
infocmp uses the following environment variables:

**TERMINFO**
Contains the path name of the terminfo database.

**TERM**
Contains the name of your terminal, that is, the current terminal definition.

Related information
captoinfo, tic
integer - Mark each variable with an integer value

Format
integer [number]

Description
integer is an alias for typeset -i. Like typeset -i, integer marks each variable as having an integer value, thus making arithmetic faster. If number is given and is nonzero, the output base of each variable is number. The default is decimal.

See “typeset - Assign attributes and values to variables” on page 761 for more information.

Related information
typeset, sh

ipcrm - Remove message queues, semaphore sets, or shared memory IDs

Format
ipcrm [-m SharedMemoryID] [-M SharedMemoryKey] [-q QMessageID] [-Q MessageKey] [-s SemaphoreID] [-S SemaphoreKey]

Description
ipcrm removes one or more message queues, semaphores set, or shared memory identifiers.

Options
-m SharedMemoryID
Removes the shared memory identifier SharedMemoryID. The shared memory segment and data structure associated with SharedMemoryID are also removed after the last detach operation.

-M SharedMemoryKey
Removes the shared memory identifier, created with the key SharedMemoryKey. The shared memory segment and data structure associated with it are also removed after the last detach operation.

-q QMessageID
Removes the message queue identifier MessageID and the message queue and data structure associated with it.

-Q MessageKey
Removes the message queue identifier, created with the key MessageKey, and the message queue and data structure associated with it.

-s SemaphoreID
Removes the semaphore identifier SemaphoreID and the set of semaphores and data structure associated with it.

-S SemaphoreKey
Removes the semaphore identifier, created with the key SemaphoreKey, and the set of semaphores and data structure associated with it.

The msgctl, shmctl, and semctl subroutines provide details of the remove operations. You can use the ipcrm command to find the identifiers and keys.
Examples

1. To remove the shared memory segment associated with SharedMemoryID 18602, enter:

   ```
ipcrm -m 18602
   ```

2. To remove the message queue that was created with a key of 0xC1C2C3C3, enter:

   ```
ipcrm -Q 0xC1C2C3C4
   ```

Exit values

0  
Successful completion.

1  
Incorrect command-line option.

Related information

ipcs

**ipcs - Report status of the interprocess communication facility**

Format

```
ipcs [-mqS [-abcoptxyMB]]
```

Description

**ipcs** writes to the standard output information for active interprocess communication facilities. If you do not specify any flags, **ipcs** writes information in a short form about currently active message queues, shared memory segments, and semaphores.

The column headings and the meaning of the columns in an **ipcs** command listing are listed in Table 16 on page 354. The letters in parentheses indicate the command flags that cause the corresponding heading to appear. (all) means that the heading is always displayed. These flags determine what information is provided for each facility. They do not determine which facilities are listed.

<table>
<thead>
<tr>
<th>Table 16: Explanation of the ipcs command listing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Column heading</strong></td>
</tr>
<tr>
<td>T (all except y)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ID (all except x,w,y,S,B)</td>
</tr>
<tr>
<td>KEY (all except y,S,B)</td>
</tr>
</tbody>
</table>
Table 16: Explanation of the ipcs command listing (continued)

<table>
<thead>
<tr>
<th>Column heading</th>
<th>Meaning of the column</th>
</tr>
</thead>
</table>
| MODE (all except x,w,y,S,B) | The facility access modes and flags. The mode consists of 11 characters that are interpreted as follows:  
The first two characters can be the following:  
• R: If a process is waiting in a msgrcv() system call.  
• S: If a process is waiting in a msgsnd() system call.  
• D: If the associated shared memory segment was removed. It disappears when the last process that is attached to the segment detaches it.  
The next nine characters are interpreted as three sets of three characters each. The first set refers to the owner's permissions; the next to permissions of others in the user group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is unused.  
The permissions are indicated as follows:  
• r: If read permission is granted.  
• w: If write permission is granted.  
• a: If alter permission is granted.  
• -: If the indicated permission is not granted. |
| OWNER (all, except S,B)  | The login name or user ID of the owner of the facility entry.                         |
| GROUP (all)              | The name or group ID of the group that owns the facility entry.                      |
| CREATOR (a,c)            | The login name or user ID of the creator of the facility entry.                     |
| CGROUP (a,c)             | The group name or group ID of the creator of the facility entry.                   |
| CBYTES (a,x,o)           | The number of bytes in messages currently outstanding on the associated message queue. |
| INFO (x)                 | Provides additional extended state information. Under this field is returned one or more of the following codes (codes are not mutually exclusive unless noted):  
For shared memory output:  
• M: Megaroo.  
For semaphore output:  
• P: PLO in use. Mutually exclusive with L.  
• L: Latch in use. Mutually exclusive with P.  
• B: Binary semaphore.  
For message queue output:  
• P: PLO in use. Mutually exclusive with L.  
• L: Latch in use. Mutually exclusive with P.  
• R: RCV type PID.  
• S: Send type PID.  
• 1: PLO1 flag on--IPC_PLO1 set on msgget().  
• 2: PLO2 flag on--IPC_PLO2 set on msgget(). |
<p>| QNUM (a,o)               | The number of messages currently outstanding on the associated message queue.       |
| QBYTES (a,b)             | The maximum number of bytes allowed in messages outstanding on the associated message queue. |</p>
<table>
<thead>
<tr>
<th>Column heading</th>
<th>Meaning of the column</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LSPID (p)</strong></td>
<td>The ID of the last process that sent a message to the associated queue.</td>
</tr>
<tr>
<td><strong>LRPID (p)</strong></td>
<td>The ID of the last process that received a message from the associated queue.</td>
</tr>
<tr>
<td><strong>STIME (a,t)</strong></td>
<td>The time when the last message was sent to the associated queue.</td>
</tr>
<tr>
<td><strong>RTIME (a,t)</strong></td>
<td>The time when the last message was received from the associated queue.</td>
</tr>
<tr>
<td><strong>CTIME (a,t)</strong></td>
<td>The time when the associated entry was created or changed.</td>
</tr>
<tr>
<td><strong>NATTCH (a,o)</strong></td>
<td>The number of processes that are attached to the associated shared memory segment.</td>
</tr>
<tr>
<td><strong>SEGSZPG (a,b,x)</strong></td>
<td>The size in <em>pages</em> of the associated shared memory segment.</td>
</tr>
<tr>
<td><strong>PGSZ (a,b,x)</strong></td>
<td>The page size of the associated shared memory segment.</td>
</tr>
<tr>
<td><strong>SEGSZ (a,b,x)</strong></td>
<td>The size in <em>bytes</em> of the associated shared memory segment.</td>
</tr>
<tr>
<td><strong>CPID (p)</strong></td>
<td>The process ID of the creator of the shared memory entry.</td>
</tr>
<tr>
<td><strong>LPID (p)</strong></td>
<td>The process ID of the last process to attach or detach the shared memory segment.</td>
</tr>
<tr>
<td><strong>ATIME (a,t)</strong></td>
<td>The time when the last attach was completed to the associated shared memory segment.</td>
</tr>
<tr>
<td><strong>DTIME (a,t)</strong></td>
<td>The time the last detach was completed on the associated shared memory segment.</td>
</tr>
<tr>
<td><strong>NSEMS (a,b)</strong></td>
<td>The number of semaphores in the set associated with the semaphore entry.</td>
</tr>
<tr>
<td><strong>OTIME (a,t)</strong></td>
<td>The time the last semaphore operation was completed on the set associated with the semaphore entry.</td>
</tr>
<tr>
<td><strong>RCVWAIT (x)</strong></td>
<td>A count of <em>msgrcv()</em> waiters.</td>
</tr>
<tr>
<td><strong>SNDWAIT (x)</strong></td>
<td>A count of <em>msgsnd()</em> waiters.</td>
</tr>
<tr>
<td><strong>MSGQPID (w)</strong></td>
<td>For the message Q report, up to 10 lines of data is shown under this heading.</td>
</tr>
<tr>
<td><strong>MSGQTYPE (w)</strong></td>
<td>For the message Q report, up to 10 lines of data is shown under this heading.</td>
</tr>
<tr>
<td><strong>RCVPID (w)</strong></td>
<td>The process ID of a <em>msgrcv()</em> waiter. A maximum of 10 process IDs can be written.</td>
</tr>
<tr>
<td><strong>RCVTYP (w)</strong></td>
<td>The message type of a <em>msgrcv()</em> waiter that is associated with <strong>RCVPID</strong>. A maximum of 10 message type is written. If the caller does not have read access, this field is not displayed.</td>
</tr>
<tr>
<td><strong>SNDPID (w)</strong></td>
<td>The process ID of a <em>msgsnd()</em> waiter. A maximum of 10 process IDs can be written.</td>
</tr>
<tr>
<td><strong>SNDLEN (w)</strong></td>
<td>The message send length of a <em>msgsnd()</em> waiter that is associated with <strong>SNDPID</strong>. A maximum of 10 message send lengths can be written.</td>
</tr>
<tr>
<td><strong>TERMA (x)</strong></td>
<td>The number of times sem_val was changed during termination for semaphore adjustments.</td>
</tr>
<tr>
<td><strong>CNADJ (x)</strong></td>
<td>The current number of processes with semaphore adjustments.</td>
</tr>
<tr>
<td><strong>SNCNT (x)</strong></td>
<td>The number of waiters waiting for a sem_val greater than zero.</td>
</tr>
<tr>
<td><strong>SZCNT (x)</strong></td>
<td>The number of waiters waiting for a sem_val equal to zero.</td>
</tr>
<tr>
<td><strong>WTRPID (w)</strong></td>
<td>The process IDs of a <em>semop</em> waiter. A maximum of 10 semop waiters are written.</td>
</tr>
</tbody>
</table>
Table 16: Explanation of the ipcs command listing (continued)

<table>
<thead>
<tr>
<th>Column heading</th>
<th>Meaning of the column</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTRNM (w)</td>
<td>The semaphore number associated with WTRPID. A maximum of 10 semaphore numbers are written.</td>
</tr>
<tr>
<td>WTROP (w)</td>
<td>The semaphore operation value that is associated with WTRNM and WTRPID. A maximum of 10 semaphore operation values are written.</td>
</tr>
<tr>
<td>AJPID (w)</td>
<td>The process ID of a process with semaphore adjustments. A maximum of 10 process IDs are written.</td>
</tr>
<tr>
<td>AJNUM (w)</td>
<td>The semaphore number of the semaphore adjustment associated with AJPID. A maximum of 10 semaphore numbers are written.</td>
</tr>
<tr>
<td>AJPID (w)</td>
<td>The process ID of a process with semaphore adjustments. A maximum of 10 process IDs are written.</td>
</tr>
<tr>
<td>AJNUM (w)</td>
<td>The semaphore number of the semaphore adjustments associated with AJPID. A maximum of 10 semaphore numbers are written.</td>
</tr>
<tr>
<td>AJVAL (w)</td>
<td>The semaphore adjustment value that is associated with AJNUM and AJPID. A maximum of 10 semaphore adjustment values are written.</td>
</tr>
<tr>
<td>ATPID (x)</td>
<td>The process ID of a process that is attached to this shared memory segment. A maximum of 10 process IDs are written.</td>
</tr>
<tr>
<td>ATADDR (x)</td>
<td>The shared memory address where the process ATPID is attached to this segment. A maximum of 10 addresses are written.</td>
</tr>
<tr>
<td>MNIDS (y)</td>
<td>The system limit for maximum number of message queues, semaphores, or shared memory IDs.</td>
</tr>
<tr>
<td>HWIDS (y)</td>
<td>The most message queues, semaphores, or shared memory IDs created.</td>
</tr>
<tr>
<td>CIDSA (y)</td>
<td>The current number of message queues, semaphores, or shared memory IDs available.</td>
</tr>
<tr>
<td>CPRIV (y)</td>
<td>The current number of message queues, semaphores, or shared memory IDs created with IPC_PRIVATE.</td>
</tr>
<tr>
<td>CKEY (y)</td>
<td>The current number of message queues, semaphores, or shared memory IDs created with a key.</td>
</tr>
<tr>
<td>GETEX (y)</td>
<td>The number of times msgget, semget, or shmget exceeded the maximum number of IDs MNID.</td>
</tr>
<tr>
<td>MAXQB (y)</td>
<td>The system limit for maximum number of bytes on a message queue.</td>
</tr>
<tr>
<td>QMNUM (y)</td>
<td>The system limit for maximum number messages on a message queue.</td>
</tr>
<tr>
<td>ENOMEM (y)</td>
<td>The number of times msgsnd() calls returned ENOMEM.</td>
</tr>
<tr>
<td>MNSEMS (y)</td>
<td>The system limit for maximum number of semaphores per set.</td>
</tr>
<tr>
<td>MNOPS (y)</td>
<td>The system limit for maximum number of operations per semop.</td>
</tr>
<tr>
<td>CSBYTES (y)</td>
<td>The current number of bytes used by the system for semaphores.</td>
</tr>
<tr>
<td>TPAGES (y)</td>
<td>The system limit for number of system-wide shared memory pages.</td>
</tr>
<tr>
<td>SPAGES (y)</td>
<td>The system limit for number of pages per shared memory segment.</td>
</tr>
<tr>
<td>SEGPR (y)</td>
<td>The system limit for number of segments per process.</td>
</tr>
<tr>
<td>CPAGES (y)</td>
<td>The current number of system-wide shared memory pages.</td>
</tr>
<tr>
<td>MAXSEG (y)</td>
<td>The largest number of shared memory pages that are allocated to a single shared memory segment.</td>
</tr>
<tr>
<td>CREATEPID (S,B)</td>
<td>The creator PID of the map memory area. It is a unique identifier of the map area.</td>
</tr>
</tbody>
</table>
Table 16: Explanation of the ipcs command listing (continued)

<table>
<thead>
<tr>
<th>Column heading</th>
<th>Meaning of the column</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERPID (S,B)</td>
<td>The user PID of the map memory area, which is using it.</td>
</tr>
<tr>
<td>USER (S,B)</td>
<td>The user name of the user of the map memory segments entry.</td>
</tr>
<tr>
<td>SHUTDOWN (S,B)</td>
<td>This field indicates that for this particular map memory object, shutdown flag has been marked for freeing of this area. While blocks can be freed in this area, the map memory object is not freed until the last process using it terminates.</td>
</tr>
<tr>
<td>BLKSIZE (B)</td>
<td>The block size of the map area object in megabytes.</td>
</tr>
<tr>
<td>BLKSINUSE (B)</td>
<td>The number of blocks is in use in the map memory area object.</td>
</tr>
<tr>
<td>BLKSINMAP (B)</td>
<td>The number of blocks in this map area object.</td>
</tr>
<tr>
<td>BLKSMAPPED (B)</td>
<td>The number of blocks that are mapped by this process.</td>
</tr>
<tr>
<td>SECLABEL (M)</td>
<td>The multilevel security label that is associated with Message queues, Semaphores, and Shared Memory.</td>
</tr>
</tbody>
</table>

Options

-q
 Writes information about active message queues.

-m
 Writes information about active shared memory segments.

-s
 Writes information about active semaphore set.

-S
 Write information about active __map memory segments.

If -q, -m, -s, or -S are specified, only information about those facilities is written. If none of these four are specified, information about message queues, shared memory segments, and semaphores are written subject to the following options. __Map memory information will not written unless the -S is specified.

-a
 Uses the -b, -c, -o, -p, and -t flags.

-b
 Writes the maximum number of bytes in messages on queue for message queues, the size of segments for shared memory, and the number of semaphores in each semaphores set. This option is ignored for __map memory option [-S].

-c
 Writes the login name and group name of the user that made the facility. This option is ignored for __map memory option [-S].

-m
 Writes information about active shared memory segments.

-o
 Writes the following usage information:
   • Number of messages on queue.
   • Total number of bytes in messages in queue for message queues.
   • Number of processes that are attached to shared memory segments.

-p
 Writes the following:
   • Process number of the last process to receive a message on message queues.
   • Process number of the creating process.
• Process number of last process to attach or detach on shared memory segments.

-q
Writes information about active message queues.

-s
Writes information about active semaphore set.

-t
• Time of the last control operation that changed the access permissions for all facilities.
• Time of the last sendmsg() and recvmsg() on message queues.
• Time of the last shmat and shmct on shared memory.
• Time of the last semop on semaphore sets.

-w
Writes message queue wait status and semaphore adjustment status in these fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Field</th>
<th>Field</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJNUM</td>
<td>KEY</td>
<td>RCVPID</td>
<td>T</td>
</tr>
<tr>
<td>AJPID</td>
<td>MSGQPID</td>
<td>RCVTYP</td>
<td>WTRNM</td>
</tr>
<tr>
<td>AJVAL</td>
<td>MSGQTPY</td>
<td>SNDLEN</td>
<td>WTROP</td>
</tr>
<tr>
<td>GROUP</td>
<td>OWNER</td>
<td>SNDPID</td>
<td>WTRPID</td>
</tr>
</tbody>
</table>

This option ignores all other print options except -x and -y.

-x
Writes extended status in these fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Field</th>
<th>Field</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATADDR</td>
<td>KEY</td>
<td>SEGSZ</td>
<td>T</td>
</tr>
<tr>
<td>ATPID</td>
<td>OWNER</td>
<td>SEGSZPG</td>
<td>TERMA</td>
</tr>
<tr>
<td>CNADJ</td>
<td>PGSZ</td>
<td>SNCNT</td>
<td></td>
</tr>
<tr>
<td>GROUP</td>
<td>QCBYTES</td>
<td>SNDWAIT</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>RCVWAIT</td>
<td>SZCNT</td>
<td></td>
</tr>
</tbody>
</table>

This option will be ignored for __map memory option [-S]. This option ignores all other print options except the -y option.

-y
Writes summary and system limit status in these fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Field</th>
<th>Field</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIDSA</td>
<td>ENOMEM</td>
<td>MAXSEG</td>
<td>SPAGES</td>
</tr>
<tr>
<td>CKEY</td>
<td>GETEX</td>
<td>MNIDS</td>
<td>TPAGES</td>
</tr>
<tr>
<td>CPAGES</td>
<td>HWIDS</td>
<td>MNOPS</td>
<td></td>
</tr>
<tr>
<td>CPRIV</td>
<td>QMNUM</td>
<td>MNSEMS</td>
<td></td>
</tr>
<tr>
<td>CSBYTES</td>
<td>MAXQB</td>
<td>SEGPR</td>
<td></td>
</tr>
</tbody>
</table>

This option ignores all other print options. This option is a summary and system limit status for message queues, semaphores, and shared memory. It will not include the __map memory segments [-S] summary and system limit status.

-M
Writes the multilevel security label that is associated with the resources except the __map memory facility. See z/OS Planning for Multilevel Security and the Common Criteria for more information about multilevel security.
-B
  Writes extended information about __map memory segments in these fields: BLKSIZE, BLKSINUSE, BLKSINMAP, and BLKSMAPPED. This option only applies to __map memory segments [-S].

Examples

Following is a sample output from entering ipcs without flags:

```plaintext
IPC status as of Wed Apr  6 14:56:22 EDT 1994
Message Queues:
  T   ID    KEY       MODE       OWNER    GROUP
  q 1234567890 0x4107001c -Rw-rw----     root   printq
Shared Memory:
  T   ID    KEY       MODE       OWNER    GROUP
  m    0 0x00000000 --rw-------     root   system
  m    1 0x00000001 --rw-rw-rw-     root   system
Semaphores:
  T   ID    KEY       MODE       OWNER    GROUP
  s   4096 0x01b8c86e --ra-------     root   system
  s    1 0x62b8c8e6 --ra-r--r--     root   system
  s    2 0x4d0b00a6 --ra-ra----     root   system
  s 24579 0x00000058 --ra-ra-ra-     xlin   vendor
  s 176132 0x00000059 --ra-ra-ra-     xlin   vendor
```

Following is a sample output from entering ipcs -S:

```plaintext
IPC status as of Wed Apr  6 14:56:22 EDT 1994
Map Memory Service:
  T CREATEPID USERPID   USER       GROUP    SHUTDOWN
  S     1056      114  Andrew    vendor           N
  S      165 1159   root      system         Y
  S       61  114  John      system         Y
  S   1052      111  Andrew    vendor           N
  S       96  678   xlin      vendor           Y
```

Exit values

0
  Successful completion.

1
  Failure due to incorrect command-line option.

Related information

ipcrm

**jobs - Return the status of jobs in the current session**

Format

```plaintext
jobs [-l|-p] [job-identifier...]
tcsh shell: jobs [-l]
```

Description

**jobs** produces a list of the processes in the current session. Each such process is numbered for easy identification by `fg` or `kill`, and is described by a line of information:

```
[job-identifier]  default state  shell_command
```

**job-identifier**

Is a decimal number that identifies the process for such commands as `fg` and `kill` (preface `job-identifier` with `%` when used with these commands).
default
Identifies the process that would be the default for the `fg` and `bg` commands (that is, the most recently suspended process). If `default` is a +, this process is the default job. If `default` is a -, this job becomes the default when the current default job exits. There is at most one + job and one – job.

state
Shows a job as:

- Running
  If it is not suspended and has not exited.
- Done
  If it exited successfully.
- Done(exit status)
  If it exited with a nonzero exit status.
- Stopped (signal)
  If it is suspended; `signal` is the signal that suspended the job.

shell_command
Is the associated shell command that created the process.

In the tcsh shell, `jobs` lists the active jobs. With `-l`, lists process IDs in addition to the normal information. See "tcsh: Invoke a C shell" on page 671.

Options
- `-l`
  Displays the process group ID of a job (before state).
- `-p`
  Displays the process IDs of all processes.

The `-l` and `-p` options are mutually exclusive.

Localization
`jobs` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Usage notes
`jobs` is a built-in shell command.

Exit values
0
  Successful completion.
2
  Failure due to an incorrect command-line argument.

Portability
POSIX.2 User Portability Extension.
join - Join two sorted textual relational databases

**Format**

```
join [-a n] [-e s] [-o list] [-t c] [-v n] [-1 n] [-2 n] file1 file2
join [-a n] [-e s] [-j[n] m] [-o list] [-t c] file1 file2
```

**Description**

`join` joins two databases. It assumes that both `file1` and `file2` contain textual databases in which each input line is a record and that the input records are sorted in ascending order on a particular join key field (by default the first field in each file). If you specify `-` in place of `file1` or `file2`, `join` uses the standard input (stdin) for that file. If you specify `- -` in place of both `file1` and `file2`, the output is undefined.

Conceptually, `join` computes the Cartesian product of records from both files. By default, spaces or tabs separate input fields and `join` discards any leading or trailing white space. (There can be no white-space-delimited empty input fields.) It then generates output for those combined records in which the `join` key field (the first field by default) matches in each file. The default output for `join` is the common `join` key field, followed by all the other fields in `file1`, and then all the other fields in `file2`. The other fields from each file appear in the same order they appeared in the original file. The default output field separator is a space character.

**Restriction:** Line lengths in files that are used with the `join` command are limited to a length of 2048.

**Options**

- `-a n`
  Produces an output line for lines that do not match in addition to one for a pair of records that does match. If you specify `n` as one of `1` or `2`, `join` produces unpaired records from only that file. If you specify both `-a 1` and `-a 2`, it produces unpaired records from both files.

- `-e string`
  Replaces an empty field with `string` on output. In a double-byte locale, `string` can contain double-byte characters.

- `-j[n] m`
  Uses field number `m` as the join key field. By default, the join key field is the first field in each input line. As with the `-a` option, if `n` is present, this option specifies the key field just for that file; otherwise, it specifies it for both files.

- `-o list`
  Specifies the fields to be output. You can specify each element in `list` as either `n.m`, where `n` is a file number (1 or 2) and `m` is a field number, or as 0 (zero), which represents the join field. You can specify any number of output fields by separating them with blanks or commas. The POSIX-compatible version of this command (first form in the syntax) requires multiple output fields to be specified as a single argument; therefore, shell quoting might be necessary. `join` outputs the fields in the order you list them.

- `-t c`
  Sets the field separator to the character `c`. Each instance of `c` introduces a new field, making empty fields possible. In a double-byte locale, `c` can be a double-byte character.

- `-v n`
  Suppresses matching lines. If you specify `n` as one of `1` or `2`, `join` produces unpaired records from only that file. If you specify both `-v 1` and `-v 2`, it produces unpaired records from both files. Lines that are produced by using the `-a` option are not suppressed.
**-1 n**  
Uses the nth field of file1 as the join key field.

**-2 n**  
Uses the nth field of file2 as the join key field.

**Examples**

1. The following script produces a report about files in the working directory that contains the file name, file mode, and an estimate at what the file contains:

   ```sh
   file * | tr -s ':' ';' >temp1
   ls -l | tr -s ' ' ':' >temp2
   join -t';' -j2 9 -o 1.1 2.1 1.2 ---
   temp1 temp2
   rm temp[12]
   ```

2. This example uses the historical implementation of the `join` command. The third line in the POSIX-compatible script could be:

   ```sh
   join -t';' -2 9 -o 1.1,2.1,1.2 -- temp1 temp2
   ```

**Localization**

`join` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Exit values**

0  
Successful completion.

1  
Failure due to any of the following situations:

- Incorrect syntax.
- The wrong number of command-line arguments.
- Inability to open the input file.
- Badly constructed output list.
- Too many -o options on the command line

2  
Failure due to an incorrect command-line argument.

**Messages**

Most diagnostics deal with argument syntax and are self-explanatory. For example:

**Badly constructed output list at list**

Indicates that the list for a -o option did not have the proper syntax.
kill - End a process or job, or send it a signal

Format

```
kill -l [exit_status]
kill [ -s signal_name] [ -K [pid...][job-identifier...]]
kill [ -signal_name] [pid ...][job-identifier ...]
kill [ -signal_number] [pid ...][job-identifier ...]
```

tcsh shell:

```
kill [-signal] %job|pid ...
kill -K %job|pid ...
kill -l
```

Description

kill ends a process by sending it a signal. The default signal is SIGTERM.
kill is a built-in shell command.

In the tcsh shell, `kill [-signal] %job|pid ...` sends the specified signal (or if none is given, the TERM (terminate) signal) to the specified jobs or processes. `job` can be a number, a string, ",", ",", + or -. Signals are either given by number or by name. Enter the `signal_name` with uppercase characters. For example, if you want to send the SIGTERM signal, you would enter `kill -TERM pid` not `kill -SIGTERM pid`.

When using the tcsh `kill` command, do not use the first three characters (SIG) of the `signal_name`.

There is no default `job`. Specifying `kill` alone does not send a signal to the current job. If the signal being sent is TERM or HUP (hangup), then the job or process is sent a CONT (continue) signal as well.

`kill -l` lists the signal names. See “tcsh: Invoke a C shell” on page 671.

The `signal_numbers` and `signal_names` described in “Options” on page 364 are also used with the tcsh `kill` command.

Options

- `K`
  Sends a superkill signal to force the ending of a process or job that did not end as a result of a prior KILL signal. The process is ended with a non-retryable abend. The regular KILL signal must have been sent at least 3 seconds before the superkill signal is sent. The superkill signal cannot be sent to a process group (by using `pid` of 0 or a negative number) or to all processes (by using a `pid` of -1).

- `l`
  Displays the names of all supported signals. If you specify `exit_status` and it is the exit code of an ended process, `kill` displays the ending signal of that process.
-s signal_name
Sends the signal signal_name to the process instead of the SIGTERM signal. When using the kill command, do not use the first three characters (SIG) of the signal_name. Enter the signal_name with uppercase characters. For example, if you want to send the SIGABRT signal, enter:

```
kill -s ABRT pid
```

-signal_name
(Obsolete.) Same as -s signal_name.

-signal_number
(Obsolete.) A non-negative integer representing the signal to be sent to the process, instead of SIGTERM.

The signal_number represents the following signal names:

0  SIGNULL
1  SIGHUP
2  SIGINT
3  SIGQUIT
4  SIGILL
5  SIGPOLL
6  SIGABRT
7  SIGSTOP
8  SIGFPE
9  SIGKILL
10 SIGBUS
11 SIGSEGV
12 SIGSYS
13 SIGPIPE
14 SIGALRM
15 SIGTERM
16 SIGUSR1
17 SIGUSR2
18 SIGABND
The signal_numbers (3 and 6) associated with SIGQUIT and SIGABRT, respectively, differ from the values of SIGQUIT and SIGABRT used by the z/OS kernel, but they are supported for compatibility with other UNIX platforms. (The kill command will send the SIGQUIT or SIGABRT to the process.) This note is also true for kill in the tcsh shell.

**Operands**

*job-identifier*

Specifies the job identifier reported by the shell when a process is started with &. It is one way to identify a process. It is also reported by the `jobs` command. When using the job identifier with the `kill` command, the job identifier must be prefaced with a percent (%) sign. For example, if the job identifier is 2, the `kill` command would be entered as follows:

```
kill -s KILL %2
```

*pid*

Specifies the process ID that the shell reports when a process is started with &. You can also find it using the `ps` command. The `pid` argument is a number that may be specified as octal, decimal, or hexadecimal. Process IDs are reported in decimal. `kill` supports negative values for `pid`. 
If \textit{pid} is negative but not -1, the signal is sent to all processes whose process group ID is equal to the absolute value of \textit{pid}. The negative \textit{pid} is specified in this way:

\begin{verbatim}
kill -KILL -- -nn
\end{verbatim}

where \textit{nn} is the process group ID and may have a range of 2 to 7 digits (\textit{nn} to \textit{nnnnnnn}).

\begin{verbatim}
kill -s KILL -- -9812753
\end{verbatim}

The format must include the -- before the --nn in order to specify the process group ID.

If \textit{pid} is 0, the signal is sent to all processes in the process group of the invoker.

The process to be killed must belong to the current user; however, any process can be killed by a superuser.

\textbf{Localization}

\texttt{kill} uses the following localization environment variables:

- \texttt{LANG}
- \texttt{LC\_ALL}
- \texttt{LC\_CTYPE}
- \texttt{LC\_MESSAGES}
- \texttt{NLSPATH}


\textbf{Usage notes}

1. Restrictions for \texttt{z/OS UNIX} signal delivery are documented in the Environmental Restrictions section of \textit{z/OS UNIX System Services Programming: Assembler Callable Services Reference}.

\textbf{Exit values}

0

Successful completion.

1

Failure due to one of the following reasons:
- The job or process did not exist.
- There was an error in command-line syntax.

2

Failure due to one of the following reasons:
- Two jobs or processes did not exist.
- Incorrect command-line argument.
- Incorrect signal.

>2

Tells the number of processes that could not be killed.

\textbf{Messages}

Possible error messages include:

\textit{job-identifier is not a job}

You specified an incorrect ID.
**signal_name is not a valid signal**

You specified a noninteger signal for `kill`, or you specified a signal that is outside the range of valid signal numbers.

**Portability**

POSIX.2, X/Open Portability Guide.

**Related information**

`jobs`, `ps`, `sh`, `tcsh`

---

**left bracket - Test for a condition**

For the `[ (left bracket command), see the `test` command.

**Note:** When working in the shell, to view man page information about `[ (left bracket), type: `man left`.**

---

**ld - Link object files**

**Format**

`ld [-cVv]`  
`[-b option[,...]]...`  
`[-e function]`  
`[-f filename]...`  
`[-L directory]...`  
`[-l libname]...`  
`[-O name[,...]]`  
`[-o outfile]`  
`[-S syslibdset]...`  
`[-u function]`  
`[-x sidefile]`  
`[file.o ] ... [file.a ]... [file.x]... [-l libname] ...`

**Description**

The `ld` utility combines object files and archive files into an output executable file, resolving external references. `ld` runs the Program Management Binder.

To export the trace output to a z/OS UNIX file, use the `_LD_DEBUG_TRACE` environment variable.

**Options**

**-b option[,...]**

Specifies options to be passed to the binder. For more information about the binder and its options, see `z/OS MVS Program Management: User's Guide and Reference`. Binder options that are not specified will take the binder default, except when other `ld` options affect binder options.

**-c**

Causes pseudo-JCL to be written to standard output without actually running the binder. Pseudo-JCL provides information about exactly which binder options are being passed, and also which data sets are being used.  
Also see `-v`.  

-e function
  Specifies the name of the function to be used as the entry point of the program.

  The default value of the -e option is affected by the setting of environment variable
  _LD_ENTRY_POINT (see Environment Variables). If the -e option is not specified and
  _LD_ENTRY_POINT is null, or if function is null, the default rules of the binder will determine the entry
  point of the program. For more information about the binder and the ENTRY control statement, see

  Also see -O.

-f filename
  Specifies a file that contains a list of the names of object and archive files to be used as input. The
  listed files will be used in addition to any files specified as operands.

-L directory
  Specifies the directories to be used to search for archive libraries that are specified by the -l operand.
  The directories are searched in the order specified, and then in the directories that are specified by
  the _LD_LIBDIRS environment variable or its default. You cannot specify an MVS data set as an
  archive library directory.

-l libname
  Specifies the name of an archive library. ld searches for the file libname.a in the directories
  specified on the -L option and then in the directories specified by the _LD_LIBDIRS environment
  variable or its default. The first occurrence of the archive library is used.

  You can also specify an MVS data set; you must specify the full data set name, because there are no
  rules for searching library directories.

  The data set specified must be a C370LIB object library or a load library. If a data set specified as a
  library has undefined (U) record format, then it is assumed to be a load library. For more information
  about the object library utility, see z/OS XL C/C++ User's Guide.

-o name[, name]...
  Specifies the name of the code topic to be ordered to the beginning of the executable. The binder
  control statement order will be generated. The default value of -o can be provided by the environment
  variable _LD_ORDER (see Environment Variables).

-o outfile
  Specifies the name of the executable file produced by ld. The default output file is a.out.

-S syslibdset
  Specifies the name of a system library (SYSLIB) data set that will be used to resolve symbols.

-u function
  Specifies the name of the function to be added to the list of external symbols to be resolved. This
  option can be useful if the only input is archive libraries. If this option is not specified, no external
  symbol is added.

-V
  This verbose option produces binder listings and directs them to standard output.

-v
  This verbose option causes pseudo-JCL to be written to standard output before the binder is run. It
  provides information about exactly which binder options are being passed, and also which data sets
  are being used.

-X sidefile
  Specifies the name of a sidedeck file or data set that ld will write to when producing a DLL (dynamic
  link library).
Operands

file.a
Specifies the name of an archive file, as produced by the ar command, to be used by the binder for resolving external references. To specify an MVS data set name, precede the name with double slash (/ /), in which case the last qualifier of the data set name must be LIB. The data set specified must be a C370LIB object library or a load library. See the description of the -l libname operand for more information about using data sets as libraries.

file.o
Specifies the name of an object file, produced by the C/C++ run-time compiler or assembler, to be link-edited.

To specify an MVS data set name to be link-edited, precede the file name with double slash (//), in which case the last qualifier of the data set name must be OBJ.

If a partitioned data set is specified, more than one member name can be specified by separating each with a comma. For example:

ld //file.OBJ(mem1,mem2,mem3)

file.x
Specifies the name of a definition sidedeck produced by ld when creating a DLL (dynamic link library), and used by ld when linking an application using the DLL. See the description of side-deck processing in z/OS MVS Program Management: User's Guide and Reference.

To specify an MVS data set name, precede the file name with double slash (//), in which case the last qualifier of the data set name must be EXP.

If a partitioned data set is specified, more than one member name can be specified by separating each with a comma. For example:

ld //file.EXP(mem1,mem2,mem3)

Environment variables

You can use environment variables to specify necessary system and operational information to ld. When a particular environment variable is not set, ld uses the default shown. For information about the JCL parameters that are used in these environment variables, see z/OS MVS JCL User's Guide.

_LD_ACCEPTABLE_RC
The maximum allowed return code (result) of the binder invocation. If the result is between zero and this value (inclusive), then it is treated internally by ld exactly as if it were a zero result, except that message IEW5033 might be issued. For more information about binder return codes, see z/OS MVS Program Management: User's Guide and Reference. The default value is:

"4"

_LD_ASUFFIX
The suffix by which ld recognizes an archive file. This environment variable does not affect the treatment of archive libraries specified as -l operands, which are always prefixed with lib and suffixed with .a. The default value is:

"a"

_LD_ASUFFIX_HOST
The suffix by which ld recognizes a library data set. This environment variable does not affect the treatment of data set libraries specified as -l operands, which are always used exactly as specified. The default value is:

"LIB"
_LD_DAMPLEVEL
The minimum severity level of dynamic allocation messages returned by dynamic allocation message processing. Messages with severity greater than or equal to this number are written to standard error. However, if the number is out of the range shown here (that is, less than 0 or greater than 8), then ld dynamic allocation message processing is disabled. The default value is:

"4"

Following are the values:

0 Informational
1-4 Warning
5-8 Severe

_LD_DAMPNAME
The name of the dynamic allocation message processing program called by ld. It must be a member of a data set in the search order used for MVS programs. The default dynamic allocation message processing program is described in z/OS MVS Programming: Authorized Assembler Services Guide. The default value is:

"IEFDB476"

_LD_DCBU
The DCB parameters used by ld for data sets with the attributes of record format undefined and data set organization partitioned. This DCB is used by ld for the output file when it is to be written to a data set. The default value is:

"(RECFM=U, LRECL=0, BLKSIZE=6144, DSORG=P0)"

_LD_DCB80
The DCB parameters used by ld for data sets with the attributes of record format fixed blocked and logical record length 80. The default value is:

"(RECFM=FB, LRECL=80, BLKSIZE=5680)"

_LD_DEBUG_DUMP
The name of a data set to be used for capturing diagnosis data during execution of the binder. An unformatted dump is written to this data set by the binder when it encounters a binder ABEND situation (usually accompanied by message IEW2900W) or when the binder option DUMP is specified. The data set must have been created before ld is invoked, and must be created with RECFM=VBA and LRECL=125. If this environment variable is null, the binder dump will not be captured. The default value is

"" (null)

_LD_DEBUG_TRACE
The name of a data set or UNIX file to be used for capturing diagnosis data during the execution of the binder. An unformatted trace is written to the data set or UNIX file, as applicable, by the binder. If the variable begins with two leading slashes, it is a data set name. In this case, this data set must have been created before ld was invoked, and must be created with RECFM=VB and LRECL=84. If this environment variable is null, the binder trace will not be captured. The default value is

"" (null)

The trace output file can be a z/OS UNIX file. The binder writes the unformatted trace to the data set. If the file name does not have a leading slash, the file name is treated as a UNIX file name with relative directory. If the file name has a leading slash, it is treated as a UNIX file name with absolute directory.
_LD_ENTRY_POINT
The value to be used as the -e option if -e is not specified. The default value is

"" (null)

LD_EXTRA_SYMBOL
The value to be used as the -u option if -u is not specified. The default value is

"" (null)

LD_LIBDIRS
The directories that are used by ld as the default place to search for archive libraries that are specified using the -l operand. The default value is:

"/lib /usr/lib"

LD_ORDER
The value to be used as the -O option if -O is not specified. The default value is

"" (null)

LD_NEW_DATACLAS
The DATACLAS parameter that is used by ld for any new data sets it creates. The default value is

"" (null)

LD_NEW_DSNTYPE
The DSNTYPE parameter that is used by ld for any new data sets it creates. The default value is

"LIBRARY"

which means that new data sets are created as type PDSE.

LD_NEW_MGMTCLAS
The MGMTCLAS parameter that is used by ld for any new data sets it creates. The default value is

"" (null)

LD_NEW_SPACE
The SPACE parameter that is used by ld for any new data sets it creates. The default value is

"" (null)

LD_NEW_STORCLAS
The STORCLAS parameter that is used by ld for any new data sets it creates. The default value is

"" (null)

LD_NEW_UNIT
The UNIT parameter that is used by ld for any new data sets it creates. The default value is

"" (null)

LD_OPERANDS
These operands are parsed as if they were specified after all other operands on the ld command line. The default value is

"" (null)
These options are parsed as if they were specified after all other operands on the ld command line. The default value is

"" (null)

The suffix by which ld recognizes an object file. The default value is

"o"

The suffix by which ld recognizes an object data set. The default value is

"OBJ"

The system library data set concatenation to be used to resolve symbols. The default value is:

" " (null)

The system definition sidedeck list to be used to resolve symbols. A definition sidedeck contains link-editing phase IMPORT control statements naming symbols that are exported by a DLL. The default value is

"" (null)

The suffix by which ld recognizes a definition sidedeck file of exported symbols. The default value is

"x"

The suffix by which ld recognizes a definition sidedeck data set of exported symbols. The default value is

"EXP"

Usage notes

1. Messages generated from the use of the ld command are provided in z/OS MVS System Messages, Vol 8 (IEF-IGD).

2. ld provides similar function to the link-edit step of the c89 command. It does not provide any functions of the compile and assembly phases of c89, nor any steps of the link-edit phase except for the link-edit step - ld merely calls the program management binder. The other main difference is that c89 has default settings that are designed for linking an object file that is produced by the C/C++ run-time compiler for execution in the Language Environment, whereas the default settings of ld do not include compiler or environment assumptions, and it can therefore be more easily used to link objects from other compilers or that are destined for environments other than Language Environment.

You can use ld options, operands, or environment variables to cause ld to create executable modules that are compatible with those c89 produces by default:
<table>
<thead>
<tr>
<th>Option or operand</th>
<th>Environment variable</th>
<th>Value for c89 compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e</td>
<td>_LD_ENTRY_POINT</td>
<td>CEESTART or, for AMODE 64 code, CELQSTRRT</td>
</tr>
<tr>
<td>-O</td>
<td>_LD_ORDER</td>
<td>CEESTART or, for AMODE 64 code, CELQSTRRT</td>
</tr>
<tr>
<td>-S</td>
<td>_LD_SYSLIB</td>
<td>//'CEE.SCEELKEX': //'CEE.SCEELKED': //'CBC.CCCNOBJ': //'SYS1.CSSLIB' or installation equivalent names</td>
</tr>
<tr>
<td>-u</td>
<td>_LD_EXTRA_SYMBOL</td>
<td>CEEMAIN or, for AMODE 64 code, CELQMAIN</td>
</tr>
<tr>
<td>file.a</td>
<td>_LD_OPERANDS</td>
<td>//CEE.SCEE0BJ</td>
</tr>
</tbody>
</table>

3. When a data set name is specified, the argument must start with double-slash (/ /) followed by the data set name. If the name is enclosed in single quotation marks, it is assumed to be fully qualified and is taken as is. Otherwise, the user login name followed by a period is prefixed.

4. To be able to specify an operand that begins with a dash (-), you must use the double dash (--) end-of-options delimiter.

5. When ld is invoked from the shell, any option-arguments or operands specified that contain characters with special meaning to the shell must be escaped. For example, source files that are specified as PDS member names contain parentheses; and if they are specified as fully qualified names, they contain single quotation marks. To escape these special characters, either enclose the option-argument or operand in double quotation marks, or precede each character with a backslash.

6. Options and arguments are processed in the order read (from left to right). Where there are conflicts, the last specification is used. If options that require arguments are specified more than once, the last specification is used except as follows:

- **-b**
  - Binder options are appended in the order they are specified

- **-f**
  - Each file is processed when the -f option is encountered

- **-L**
  - Library directories are appended in the order they are specified

- **-l**
  - Libraries are searched when the -l option is encountered

- **-S**
  - SYSLIB data sets are appended in the order they are specified

All operands are processed in the order they are specified.
7. Because archive library files are searched when their names are encountered, the placement of -l operands and file.a operands is significant. You may have to specify a library multiple times on the command string, if subsequent specification of file.o files requires that additional symbols be resolved from that library.

8. The following environment variable specifies the name of an MVS program to be executed and can be at most eight characters in length. You can dynamically alter the search order used to find MVS programs by using the STEPLIB environment variable. For more information about the STEPLIB environment variable, see the section on commonly used environment variables in z/OS UNIX System Services Planning. It is also described under the sh command.

   • _LD_DAMP_NAME

9. The following environment variables can be at most 15 characters in length. Do not specify any periods (.) when setting these environment variables because they would then never match their corresponding operands:

   • _LD_ASUFFIX
   • _LD_ASUFFIX_HOST
   • _LD_OSUFFIX
   • _LD_OSUFFIX_HOST
   • _LD_XSUFFIX
   • _LD_XSUFFIX_HOST

10. The following environment variable is a parsed colon-delimited data set name, and represents a data set concatenation or a data set list:

    • _LD_SYSLIB

11. The following environment variables specify the names of MVS databases and can be at most 44 characters in length:

    • _LD_DEBUG_DUMP
    • _LD_DEBUG_TRACE

12. The following environment variables can be at most 63 characters in length:

    • _LD_NEW_DATAACLAS
    • _LD_NEW_DSNTYPE
    • _LD_NEW_MGMTCLAS
    • _LD_NEW_SPACE
    • _LD_NEW_STORCLAS
    • _LD_NEW_UNIT

13. The following environment variable is for specification of the SPACE parameter, and supports only the syntax as shown, including all commas and parentheses (example: "((10,10,10))"). PRIMARY is the number of tracks of primary to be allocated, SECONDARY the number of secondary tracks, and DIRBLKS the number of directory blocks. DIRBLKS must be specified even when it is not needed.

    • _LD_NEW_SPACE

14. The following environment variable is for specification of the DSNTYPE parameter, and supports only the sub-parameters LIBRARY or PDS (or null for the default, LIBRARY):

    • _LD_NEW_DSNTYPE

15. The following environment variables can be at most 127 characters in length:

    • _LD_DCBU
    • _LD_DCB80

These environment variables are for specification of DCB information, and support only the following DCB subparameters, with the noted restrictions:
Incorrect values are ignored.

LRECL
None

BLKSIZE
None

DSORG
Incorrect values are treated as if no value had been specified.

16. The following environment variables are parsed as blank-delimited words, and therefore no embedded blanks or other white space is allowed in the value specified. The maximum length of each word is 1024 characters:

- _LD_LIBDIRS
- _LD_OPTIONS
- _LD_OPERANDS

Localization

ld uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES


Exit values

0
Successful completion.

>0
An error occurred during processing.

Related information

c89

**let - Evaluate an arithmetic expression**

**Format**

```
let expression ...
((expression))
```

**Description**

`let` evaluates each arithmetic expression from left to right, with normal algebraic precedence (multiplication before addition, for example). `let` uses long integer arithmetic with no checks for overflow. No output is generated; the exit status is 0 if the last expression argument has a nonzero value, and 1 otherwise.
The following two lines are equivalent: the second form avoids quoting and enhances readability. These two forms are extensions to the POSIX standard.

```bash
let "expression"
((expression))
```

The POSIX version of this command is as follows:

```bash
($(expression))
```

Expressions consist of named variables, numeric constants, and operators. Characters in the names of named variables must come from the POSIX portable character set.

See “Arithmetic substitution” on page 600.

**Examples**

Examples of the three forms of the `let` command are as follows:

1. The example

```bash
let a=7
echo $a
```

produces:

```
7
```

2. The example

```bash
echo $((a=7*9))
```

produces:

```
63
```

3. The example

```bash
((a=3+4))
echo $a
```

produces:

```
12
```

**Usage notes**

`let` is a built-in shell command.

**Localization**

`let` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

0

The last argument evaluated to a nonzero value.
The last argument evaluated to a zero value, or the expression contained a syntax error or tried to divide by zero.

Portability

POSIX.2. `let` and `((expression))` are extensions to the POSIX.2 standard. The POSIX.2 portable facility for arithmetic expression evaluation is `$((expression))`. See “Arithmetic substitution” on page 600 for more information.

The `((  ))` syntax only works if the `set -o korn` option is in effect.

Related information

expr, sh, test

**lex - Generate a program for lexical tasks**

**Format**

```
lex [-achLntV] [-o file.c] [-P proto] [-p prefix] [file.l ...]
```

**Description**

`lex` reads a description of a lexical syntax, in the form of regular expressions and actions, from `file.l`. If you do not provide `file.l`, or if the file is named `-`, `lex` reads the description from standard input (standard input). It produces a set of tables that, together with additional prototype code from `/etc/yylex.c`, constitute a lexical analyzer to scan those expressions. The resulting recognizer is suitable for use with `yacc`. You can find detailed information about the use of `lex` in `z/OS UNIX System Services Programming Tools`.

For a description of the typedefs, constants, variables, macros, and functions in the table file, which can be used to access the lexical analyzer's variables or to control its operations, see `z/OS UNIX System Services Programming Tools`.

A locale is the subset of a user's environment that depends on language and cultural conventions. A locale defines such things as the definition of characters, and the collation sequence of those characters. POSIX.2 defines a POSIX locale, which is essentially USASCII. Because `lex` generates code that is then compiled before being executed, it is difficult for `lex` to act properly on collation information. Therefore, the POSIX.2 standard does not require `lex` to accept any locales other than the POSIX locale. `lex` accepts regular expressions in this locale only.

**Options**

- `-a`
  Generates 8-bit tables instead of 7-bit tables. On systems with 8-bit character sets (such as this one), this option is always enabled.

- `-c`
  Generates C code. Because this is the default, this option is provided only for compatibility with other implementations.

- `-h`
  Prints a brief list of the options and quits.

- `-l`
  Suppresses `#line` directives in the generated code.

- `-n`
  Suppresses the display of table sizes by the `-v` option. If you did not specify `-v` and there are no table sizes that are specified in `file.1`, `lex` behaves as though you specified `-n`. 
-o file.c
  Writes the lexical analyzer (internal state tables) onto the named output file, instead of the default file lex.yy.c.

-P proto
  Uses the named code file, instead of the default prototype file /etc/yylex.c.

-p prefix
  Uses the given prefix instead of the prefix yy in the generated code.

-T
  Writes a description of the analyzer onto the file l.output.

-t
  Writes the lexical analyzer onto standard output instead of the file lex.yy.c.

-v
  Displays the space that is used by the various internal tables. Normally lex displays these statistics on stdout, but if you also specified the -t option, it displays them on standard error. If you did not choose this option and file.1 specifies table sizes, lex still displays these statistics unless you specified the -n option.

The lex library contains a number of functions essential for use with lex. These functions are described in z/OS UNIX System Services Programming Tools. The actual library to use depends on your system and compiler. For z/OS programs, you should use -ll.

Some lex programs can cause one or more tables within lex to overflow. These tables are the NFA, DFA, and move tables; lex displays an appropriate message if an overflow occurs. Change table sizes by inserting the appropriate line into the definition section of the lex input, with the number size giving the number of entries to use. This is shown in Table 17 on page 379.

<table>
<thead>
<tr>
<th>Line</th>
<th>Table size affected</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>%esize</td>
<td>Number of NFA entries</td>
<td>1000</td>
</tr>
<tr>
<td>%nsize</td>
<td>Number of DFA entries</td>
<td>500</td>
</tr>
<tr>
<td>%psize</td>
<td>Number of move entries</td>
<td>2500</td>
</tr>
</tbody>
</table>

You can often reduce the NFA and DFA space to make room for more move entries.

Files

lex uses the following files:

l.output
  Scanner machine description

lex.yy.c
  Tables and action routines

/etc/yylex.c
  The prototype lex scanner

/usr/lib/libl.a
  lex archive library with functions that are compiled for 31-bit addressing mode.

/usr/lib/libxpl.a
  lex archive library with functions compiled with XPLINK. Includes two versions: 64-bit addressing mode and 31-bit addressing mode.

Localization

lex uses the following localization environment variables:
• LANG  
• LC_ALL  
• LC_COLLATE  
• LC_CTYPE  
• LC_MESSAGES  
• LC_SYNTAX  
• NLSPATH  


**Exit values**

0  
Successful completion  

1  
Failure due to any of the following reasons:  
• Inability to create an output file.  
• Inability to open the file.  
• Missing output file name after -o.  
• Missing prefix after -p.  
• No **lex** rules.  
• No memory for DFA moves.  
• Out of NFA state space.  
• Out of DFA move space.  
• Out of DFA state space.  
• Push-back buffer overflow.  
• Read error on file.  
• Table too large for machine.  
• Too many character classes.  
• Too many translations.  
• Unknown option.  
• Write error on file.  
• Incomplete %{ declaration.  
• Token buffer overflow.

**Limits**

The parser stack depth is limited to 150 levels. Attempting to process complicated syntaxes might result in an overflow, causing an error.

**Portability**

POSIX.2, POSIX.2 C-Language Development Utilities Option, UNIX systems.

The -a, -h, -l, -o, -p, -P, and -T options are extensions of the POSIX standard.

**Related information**

yacc  
For more information, see *z/OS UNIX System Services Programming Tools*.  

---

**lex**  
380 z/OS: UNIX System Services Command Reference
line - Copy one line of standard input

**Format**

```bash
line
```

**Description**

The `line` utility is fully supported for compatibility with older UNIX systems. However, it is recommended that the `read` utility be used instead because it may provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

**Examples**

```bash
echo "Enter name:\c"
NAME='line'
```

**Localization**

`line` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

0
  A line was read successfully

1
  `line` reached end-of-file before finding a newline character

**Portability**

X/Open Portability Guide, UNIX System V.

**Related information**

cat, head, read, sh, tail

link - Create a hard link to a file

**Format**

```bash
link oldfile newfile
```

**Description**

`link` creates a hard link to an existing file. A link is a new directory entry that refers to the same file. This entry can be in the same directory that currently contains the file or in a different directory. The result is that you get a new path name that refers to the file. You can access the file under the old or new path name since both path names are of equal importance. If you use `rm` to remove one path name, the other
remains and the file contents are still available under that name. The contents of the file do not disappear until the last remaining link associated with the file is removed.

Following the format, new becomes a new path name for the existing file old. If old names a symbolic link, link creates a hard link to the file that results from resolving the path name contained in the symbolic link.

Links are allowed to files only, not to directories. A file can have any number of links to it. Thus, you can establish any number of different path names for any file.

link is implemented as a shell built-in.

**Localization**

link uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


**Exit values**

0

  Successful completion

1

  Failure due to any of the following reasons:
  - A file specified could not be found.
  - No write permission on the directory intended to contain the link.
  - No search permission on a path name component of old or new.
  - No permission to access old.
  - The path name of one of the arguments is a directory.
  - The new link file already exists.

2

  Failure due to incorrect number of arguments

**Related information**

link, ln, rm

**ln - Create a link to a file**

**Format**

```
ln [-fiRrs] old new
ln [-fiRrs] old old ... dir
ln -e [-fi] old new
```
Description

`ln` creates a link to an existing file or set of files. A *link* is a new directory entry that refers to the same file. This entry can be in the same directory that currently contains the file or in a different directory. The result is that you get a new path name that refers to the file. You can access the file under the old path name or the new one. Both path names are of equal importance. If you use `rm` to remove either name, the other one still remains and the file contents are still available under that name. The contents of the file do not disappear until you remove the last link.

A file can have any number of links to it. Thus you can establish any number of different path names for any file.

In the first form given in the syntax, `new` becomes a new path name for the existing file `old`. In the second form, `ln` creates entries for all the `old` files under the directory `dir`. For example,

```
ln yourdir/* mydir
```

creates links under `mydir` to all the files under `yourdir`. The files have the same names under `mydir` that they had under `yourdir`. `ln` always assumes this directory form when the last operand on the command line is the name of a directory. In this case, none of the `old` names can be a directory, unless `-r` or `-R` is specified.

There could already be a file with the same name as the link you are trying to set up: a conflicting path name. To deal with a conflicting path name, `ln` follows these steps.

- If you have specified `-i`, `ln` writes a prompt to standard error to ask if you want to get rid of the conflicting path name. If you answer affirmatively, `ln` attempts to remove it.
- Otherwise, if you have specified `-f`, `ln` attempts to remove the existing file without a warning.
- Otherwise, `ln` prints a diagnostic message.
- `ln` gets to this point if it is going to get rid of the conflicting path name. It therefore attempts to get rid of the conflicting path name in the same way that `rm` does. `ln` deletes the file associated with the path name if this path name is the last link to the file. If `ln` can't get rid of the conflicting path name, it does not attempt to establish the new link; it simply prints an error message on standard error and goes on to process any other files.
- If `ln` successfully gets rid of the conflicting path name, it then establishes the link.

Options

- `-e`
  Specifies that the link created by `ln` is to be an external link. One purpose for creating an external link is to create a mount point that an NFS client can use to access a data set through the Network File System feature. If you specify one of these options with `-e`, the command will fail. The normal content of an external link is a name that refers to an object outside the hierarchical file system, such as a data set. The data set that the Network File System feature uses can be any type of MVS data set. For a partitioned data set, however, you specify a fully qualified name in all caps. For example:

```
ln -e NOLL.PLIB.PGMA /u/noll/plib/pgma
```

The `-e` option is mutually exclusive with `-r`, `-R` and `-s`.

Due to the NFS protocol limitation, `-e` does not create an external link on NFS. If you want to create an external link on NFS, see the topic on creating an external link for details in `z/OS Network File System Guide and Reference`.

External links can also be used to map a z/OS UNIX file name to a PDS or PDSE member name for an executable load module. An example of how you would define the external link is:

```
ln -e MYPGM /u/smorg/mylongpgmname
```
If an application attempts to access /u/smorg/mylongpgmname as an executable file, the kernel will attempt to load MYPGM from the current MVS search order (Job Pack Queue, STEPLIB/JOBLIB, LPA, LINK LIST). The kernel services which behave this way for external links are:

- exec() (all flavors)
- spawn() (including _spawn2, spawnp, _spawnp2)
- loadhfs which is used for all DLL processing and locales

An external link can be used as a shell command to invoke a program in the current MVS search order.

- **f**
  Deletes any conflicting path names without asking you for confirmation. If -i is also specified, regardless of the order in which -i and -f appear on the command line, -i is ignored.

- **i**
  Checks with you before deleting conflicting path names. If -f is also specified, regardless of the order in which -i and -f appear on the command line, -i is ignored.

- **R**
  Links files recursively. That is, you can link an entire hierarchy of subdirectories at once. -R is mutually exclusive with the -e option.

- **r**
  Is identical to -R. -r is mutually exclusive with the -e option.

- **s**
  Creates a symbolic link. The -r option is mutually exclusive with the -e option.

  For a symbolic link, old refers to the file you want to create the link to. That file does not have to exist. The name of the symbolic link that you are creating is new.

  If you have a file called f1 and you want to create a symbolic link to it called my_sym, issue:

  ```
  ln -s f1 my_sym
  ```

  The locale settings for LC_COLLATE, LC_CTYPE, and LC_MESSAGES affect the program's interpretation of what constitutes a Yes answer when ln asks if you want to delete a conflicting path name.

**Examples**

If you define /u/user1/name1 as a symbolic link to /u/user1/name2, and then invoke name1:

1. The shell will spawn name1.
2. spawn() will access the file for name1 unaware that there is a symbolic link already established. It will access the name2 file by its underlying vnode, not the name2 handle.
3. If the sticky bit is on for the name2 file, spawn() will do the MVS search for name1 (the only name it has to work with).

Symbolic and external links with a sticky bit:

**Note:** DLLs, and all flavors of spawn() and exec(), follow the same processing as described in this section. Where it says exec(), it covers all forms of module loading.

1. External links:
   
   exec() does a stat() on the passed filename. stat() does the search, not exec(). If the filename is an external link, then stat() fails with a unique reason code which causes exec() to read the external link. If the external link name is a valid PDS member name (1–8 alphanumeric or special characters), then exec() will attempt to locate the module in the MVS search order. If it cannot be found, exec() fails.

   The external link is normally used when you want to set the sticky bit on for a file name which is longer than 8 characters or contains characters unacceptable for a PDS member name.
2. Symbolic links:

If the filename you specify is a symbolic link, and exec() sees the sticky bit on, then it will truncate any dot qualifiers. So, as long as the base filename is an acceptable PDS member name, the need to set up links in order to get exec() to go to the MVS search order should not be an issue.

For example, if you have a file named java.jll, when you put the sticky bit on, exec() will attempt to load JAVA. If exec() cannot find JAVA, it will revert to using the java.jll file in the file system.

The important thing to understand is that exec() never sees the name that the symbolic link resolves to even though it can see the stat() data for the final file.

Localization

ln uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH


Exit values

0

All requested links were established successfully.

1

Failure due to any of the following:

• An argument had a trailing / but was not the name of a directory.
• A file could not be found.
• An input file could not be opened for reading.
• An output file could not be created or opened for output.
• The new link file already exists.
• A link could not be established.
• A read error occurred on an input file.
• A write error occurred on an output file.
• The input and output files were the same file.
• Inability to access a file when using -r.
• Inability to read a directory when using -r.
• Inability to create a directory when using -r.
• A target is not a directory when using -r.
• Source and destination directory are the same when using -r.

2

Failure due to any of the following:

• Incorrect command-line option.
• Too few arguments on the command line.
• A target that should be a directory but isn't.
• No space left on target device.
• Out of memory to hold the data to be copied.
• Inability to create a directory to hold a target file.

Messages
Possible error messages include:

- **link to target name failed**
  - `ln` could not establish the link to the given file or directory. This may be because you do not have appropriate permissions, or because the target did not exist.

- **source name and target name are identical**
  - The source and the target are actually the same file (for example, because of links, on UNIX systems). In this case, `ln` does nothing.

- **target directory name on different file system than source name**
  - You cannot establish a normal link between files that are two different file systems.

- **target name must be a directory**
  - The target name must be a directory

- **cannot find file name**
  - The file name could not be found.

- **target file name already exists**
  - The target file name already exists.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
Only the `-f` option is part of the POSIX standard.

Related information
cp, locale, mv, rm

locale - Get locale-specific information

Format
`locale [-a|-m]locale [-ck] name ...`

Description
`locale` displays information about the current locale and all locales accessible to the current application.
`locale` searches directory `/usr/lib/nls/locale` for all the compiled locales.

Invoking `locale` with no options or operands displays the values of the LANG and LC_* environment variables. If a LC_* variable is not set or is overridden by LC_ALL, `locale` displays its implied value in double quotes.

The operand name can be a category name, keyword name, or the reserved name charmap. If it is a category name, `locale` selects the given category and all keywords within it for output. If name is a keyword name, `locale` selects the given keyword and its category for output. If name is charmap, `locale` displays the name of the charmap used with the `localedef` `-f` option when the locale was created.

Options
- `-a`
  - Displays information about all accessible locales including POSIX and the POSIX locale.
Note: Some of the accessible locales that are displayed will only work in specific environments. For example, the XPLINK locales will not work in the z/OS shell environment. When using this information to specify a locale, the convention is to use the descriptive locale name. See the section on locale naming conventions in z/OS XL C/C++ Programming Guide.

-c
Displays the names of selected categories.

-k
Displays the names of selected keywords. If you do not specify the -k option, locale displays the values of selected keywords but not their names. With -k, strings are written in an unambiguous form using the escape character from the current locale.

-m
Displays a list of all available character maps.

The following list contains valid locale keywords:

- abday
  Abbreviated day names
- abmon
  Abbreviated month names
- alpha
  All alphabetic characters (upper and lower case)
- am_pm
  A.m. and p.m. string
- backslash
  Encoding of \n- blank
  How a blank is represented
- character-collation
  The collating sequence
- charmap
  Mapping of character symbols to actual character encodings
- circumflex
  Encoding of ^
- cntrl
  Control characters
- codeset
  Same as code_set_name
- code_set_name
  Name of the coded character set used
- commercial_at
  Encoding of @
- currency_symbol
  Local currency symbol of the current locale
- d_fmt
  Date format
- d_t_fmt
  Date and time format
- day
  Full day names
- decimal_point
  Decimal-point characters
digit
   All numeric characters

dollar_sign
   Encoding of $

daylight_name
   The name of the daylight saving time zone (DST). The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

end_day
   Day of the week when daylight saving time (DST) ends. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

end_month
   Week of the month when daylight saving time (DST) ends. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

end_time
   Number of seconds after midnight when daylight saving time (DST) ends. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

end_week
   Month of the year when daylight saving time (DST) ends. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

exclamation_mark
   Encoding of!

frac_digits
   Number of digits to the right of the decimal place in monetary quantities

graph
   Graphic characters

grave_accent
   Encoding of` 

grouping
   String indicating the size of each group of digits in formatted nonmonetary quantities

int_curr_symbol
   International currency symbol for the current locale

int_frac_digits
   The number of displayed digits to the right of the decimal place for internationally formatted monetary quantities

left_brace
   Encoding of { 

left_bracket
   Encoding of [ 

lower
   Lowercase alphabet

mb_cur_max
   Maximum number of bytes used to represent a character

mon
   Full month names

mon_decimal_point
   Decimal-point character used to format monetary quantities

mon_grouping
   String indicating the size of each group of digits in formatted monetary quantities

mon_thousands_sep
   Separator for digits in formatted monetary quantities
n_cs_precedes
  1 if the currency_symbol precedes the value for a negative formatted monetary quantity; 0 if it does not

n_sep_by_space
  1 if the currency_symbol is separated by a space from the value of a negative formatted monetary quantity; 0 if it does not; 2 if a space separates the symbol and the sign string–if adjacent

n_sign_posn
  Value indicating the position of the negative_sign for a negative formatted monetary quantity

negative_sign
  String indicating the negative sign used in monetary quantities

noexpr
  Expression for negative

number_sign
  Encoding of #

p_cs_precedes
  1 if the currency_symbol precedes the value for a nonnegative formatted monetary quantity; 0 if it does not

p_sep_by_space
  1 if the currency_symbol is separated by a space from the value of a nonnegative formatted monetary quantity; 0 if it does not; 2 if a space separates the symbol and the string–if adjacent

p_sign_posn
  Value indicating the position of the positive_sign for a nonnegative formatted monetary quantity

positive_sign
  String indicating the positive sign used in monetary quantities

print
  Printable characters

punct
  Punctuation characters

right_brace
  Encoding of }

right_bracket
  Encoding of ]

shift
  DST time shift in seconds. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

space
  How white space is represented

start_day
  Day of the week when daylight saving time (DST) starts. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

start_month
  Month of the year when daylight saving time (DST) starts. The value ranges from 1 through 12. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

start_time
  Number of seconds after midnight when daylight saving time (DST) starts. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

start_week
  Week of the month when daylight saving time (DST) starts. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.
locale

**t_fmt**
Time format

**t_fmt_ampm**
Long date format

**tilde**
Encoding of ~

**timezone_difference**
Time zone difference in minutes. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

**timezone_name**
Time zone name. The value will be overwritten if it is different from the keyword setting defined by the TZ environment variable.

**tolower**
Uppercase to lowercase conversion

**thousands_sep**
Character used to separate groups of digits to the left of the decimal-point character in formatted nonmonetary quantities

**toupper**
Lowercase to uppercase conversion

**uctname**
Coordinated Universal Time. The value will be overwritten if it is different from the keyword setting may be overridden if TZ is set.

**upper**
Uppercase alphabet

**vertical_line**
Encoding of |

**xdigit**
Hexadecimal digits

**yesexpr**
Expression for affirmative

**Examples**
In the following examples, let's assume that locale environment variables are set as follows:

```
LANG=locale_x
LC_COLLATE=locale_y
```

1. The command:

```
locale
```

produces the following output:

```
LANG=locale_x
LC_COLLATE=locale_y
LC_CTYPE="locale_x"
LC_MONETARY="locale_x"
LC_NUMERIC="locale_x"
LC_TIME="locale_x"
LC_TOD="locale_x"
LC_MESSAGES="locale_x"
LC_ALL=
```

2. The command:

```
LC_ALL=POSIX locale -ck decimal_point
```
produces:

```bash
LC_NUMERIC
decimal_point="."
```

3. The following command shows an application of `locale` to determine whether a user supplied response is affirmative:

```bash
if printf "s\n" "$response" | grep -Eq "$(locale yesexpr)"
then
  affirmative processing goes here
else
  nonaffirmative processing goes here
fi
```

**Environment variables**

`locale` uses the following environment variable:

**TZ**

Contains the time zone to be used when displaying date and time strings.

**Localization**

`locale` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


**Exit values**

- **0**
  - Successful completion
- **1**
  - An error occurred
- **2**
  - A usage message was printed

**Portability**

POSIX.2, UNIX System V.

**Related information**

`localedef`

---

**localedef**

**- Define the locale environment**

**Format**

```bash
```
localedef

Description
localedef converts source definitions for locale categories into a format usable by functions and utilities.
localedef, which is installed as part of the Language Environment element of z/OS, utilizes c89, which is installed as part of z/OS Runtime Library Extensions.
c89 requires the installation of the C/C++ optional feature of z/OS (which provides among other things a C compiler).
A TSO/E utility called LOCALDEF is installed as part of Language Environment.

• It is not supported by the z/OS shell; for more information see z/OS XL C/C++ User’s Guide.
• The TSO/E BATCH versions of the utility do not support ASCII (-A) nor AMODE-64 (-6) options.

For information about the charmap file and locale definition source file formats, see z/OS XL C/C++ Programming Guide.

Options
-c
Creates permanent output even if there were warning messages. Normally, localedef does not create permanent output when it has issued warning messages.

-f charmap
Specifies a charmap file that contains a mapping of character symbols and collating element symbols to actual character encodings.

-i sourcefile
Specifies the file that contains the source definitions. If there is no -i, localedef reads the source definitions from the standard input.

-m methodfile
Specifies the name of a method file that describes the methods to be overridden when constructing a locale. localedef reads the method file and uses entry points when constructing the locale objects. The code set methods specified are also used in parsing the file pointed to by the CharMap variable. This requires that you provide the overriding methods in a DLL which is explicitly loaded by localedef before processing the charmap file. User method files are supported only for ASCII locales. The -m option is invalid without the -A option.

-w
Produces warning messages for duplicate definitions.
-A
Instructs localedef to generate an ASCII locale object. The -X option is implied when this option is specified.

-L binderoptions
Instructs localedef to pass additional binder options (mostly for diagnostic purposes).

-X
Instructs localedef to generate an XPLINK AMODE 31 locale object (DLL).

-6
Instructs localedef to generate an XPLINK AMODE 64 locale object (DLL). The -X option is implied when this option is specified.

name
Is the target locale. If it contains no slashes, the locale is public and localedef converts name to a full path name using the NLSPATH environment variable. If name contains one or more slashes, localedef interprets it as a full path name of where to store the created definition.

See locale for related information.
Localization
localedef uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH

The LC_COLLATE and LC_CTYPE environment variables do not affect localedef. localedef always behaves as though these variables were set to the POSIX locale.


Exit values

0
No errors occurred; the locale was successfully created.

1
Warnings occurred; the locale was successfully created.

2
The locale specification exceeded implementation limits, or the coded character set used was not supported by implementation. No locale was created.

3
The capability to create new locales is not supported by the implementation. (POSIX2_LOCALEDEF is not defined.)

>3
Warnings or errors occurred; no output was created.

localedef issues warnings when:

• The LC_CTYPE or LC_COLLATE category description uses a symbolic name that was not found in the charmap file.
• The number of operands to the order keyword exceeds the COLL_WEIGHTS_MAX limit.

Portability
POSIX.2; UNIX System V.

Related information
locale

logger - Log messages

Format

logger [-IisSTu] [-u dest] [-f filename] [-p priority] [-P facility.level|number] [-t tag] [-a tag2] string ...

Description

logger saves a message in the console log; the message consists of the string operand on the command line. Some options might be in effect by default. If they are on by default, they cannot be disabled.

logger can also write messages onto the system log that is maintained by the syslogd command when the -S option is in effect.

The -u and -i options are in effect by default. All messages from logger are prefixed by the process ID and user login user name.
If a message is not specified on the command line, the standard input is read. Because each line of standard input is treated as a log message, all terminal input is logged as a message. To prevent all subsequent input from being processed by `logger`, enter the designated escape character, such as `c`, followed by a capital C. For example: `cC`.

If `-f filename` is specified, the file is read instead of the standard input.

By default, `logger` saves the message in the console log. If the invoker does not have appropriate privileges, a message ID (BPXM023I) and the login name of the invoker are prefixed to the specific message text.

**Note:** An invoker has appropriate privileges if the BPX.CONSOLE resource in the FACILITY class is defined and the invoker is permitted to that resource, or if the invoker is running with an effective UID of 0.

**Options**

- `-f filename`  
  Reads log messages from the file `filename` rather than from the standard input.

- `-i`  
  Adds the process ID (PID) of `logger` to the message. This option is in effect by default, so all messages from `logger` are prefixed by the PID.

- `-I`  
  Adds the parent process ID (PPID) of `logger` to the message.

- `-p facility.level | number`  
  Logs the message with the specified priority. This option takes effect when the `-S` option is specified. The priority can be specified numerically or as a `facility.level` pair. For example, `"-p local3.info"` logs the messages as informational level in the local3 facility. The default is `user.notice`.

  When `number` is specified, the facility is "user" and the level is the number that is specified. That number should be 0-7, which represents `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, and `debug` separately with the same meanings of the following possible values for the level.

  Possible facility values include:

  - `user`  
    Message that is generated by random processes. This is the default facility identifier if one is not specified.

  - `local[0-7]`  
    Reserved for local use.

  Possible values for `level` include:

  - `alert`  
    A condition that should be corrected immediately, such as a corrupted system database.

  - `crit`  
    Critical conditions, such as hard device errors.

  - `debug`  
    Messages with information that normally is useful only you when you are debugging a program.

  - `emerg`  
    A panic condition. This condition is normally broadcast to all processes.

  - `err`  
    Errors.

  - `info`  
    Informational messages.

  - `notice`  
    Conditions that are not error conditions but might require special handling.

  - `warning`  
    Warning messages.
-s
Logs the message to the standard error output.

-S
Logs the message to an implementation-specific logging facility, which loads it in an appropriate system log.

-T
Adds a time stamp (%x %X format, per date) to the message. This time stamp is always in the POSIX locale, no matter the locale of the message.

-u
Adds the login name of the controlling terminal to the message. This option is in effect by default, so all messages from logger are prefixed by the login name.

Note: The following options work on z/OS systems. However, because they are system-specific, they might not work on another system.

-d destination
Specifies the routing codes. destination must be a list of numbers that are separated by spaces, tabs, or commas. The routing codes are 1 to 128 for privileged users and 1 to 28 for nonprivileged users. The default destination value is 0.

If you use d1, the message goes to the system console.

-p priority
Specifies the descriptor codes. priority must be a list of numbers that are separated by spaces, tabs, or commas. The allowed descriptor codes are 1 to 13. Descriptor codes 1 through 6, 11, and 12 are mutually exclusive. Codes 7 through 10 and 13 can be assigned in combination with any other code. For more information about descriptor codes, see z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO. The default priority value is 0.

-t tag
Adds tag to the start of the message.

-a tag2
 Adds tag2 in front of all the options and the message.

For more information about the destination and priority options, refer to z/OS MVS JCL Reference.

Examples
1. If a privileged user issues the following command (note the number 1):

   logger -d 1 This is a message

   The following display appears:

   WELLIE: 16777252: This is a message

2. If a privileged user issues the following command (note the number 1):

   logger -d 1 -a TheTag A message

   The following display appears:

   TheTag: WELLIE: 33554468: A message

3. If an unprivileged user issues the following command (note the number 1):

   logger -d 1 -a TheTag A message

   The following display appears:

   BPXMO23I (WELLIE1) TheTag: wellie1: 83886123: A message
4. If you issue the following command:

```
logger -s -a TheTag a message
```

The following display appears on standard error (stderr):

```
TheTag: WELLIE: 16777255: a message
```

**Localization**

`logger` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

0  
Successful completion.

>0  
An error occurred.

**Messages**

Possible error messages include:

- `-f filename invalid if message given`
  Both a file name and message was specified; only one is allowed.

- `file filename: system error`
  The file that is specified by `-f filename` could not be opened.

- `Formattted log message too long -- limit LINE_MAX (number)`
  The log message specified was longer than the limit specified by LINE_MAX.

- `Unknown option option`
  You specified an incorrect option to `logger`.

**Portability**

POSIX.2, X/Open Portability Guide.

All the options are extensions of the POSIX standard.

---

**logname - Return a user's login name**

**Format**

`logname`

**Description**

`logname` displays the login name of the person who issued the command. It obtains the login name through the `getlogin()` function defined in the POSIX standard. The login name is displayed as all uppercase letters, regardless of how it was entered.
Environment variables

`logname` uses the following environment variable:

**LOGNAME**

Contains your user name.

Localization

`logname` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_TYPE`
- `LC_MESSAGES`
- `NLSPATH`


Exit values

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion</td>
</tr>
<tr>
<td>1</td>
<td><code>logname</code> could not determine the login name</td>
</tr>
</tbody>
</table>

Portability

POSIX.2, X/Open Portability Guide, UNIX System V.

Related information

`env`, `id`

**lp - Send a file to a printer**

Format

```
lp [-c|msw] [-d dest] [-n number] [-o printer-option] [-t title] [file ...]
```

Description

The `lp` command prints one or more input files on a printer. If you do not specify any files on the command line, or if you specify a file name of – (dash), `lp` reads and prints the standard input. The files are printed in the same order that they are specified on the command line.

**Note:** If you are using the z/OS Infoprint Server Feature, your system automatically uses that version of the `lp` command.

Options

- `-c`
  
  Immediately copies the files to be printed. This ensures that the version of the file that exists when the print request is made is the version that is printed.

- `-d dest`
  
  Specifies `dest` as the output device. –d takes precedence over the LPDEST environment variable, which in turn takes precedence over the PRINTER environment variable.
dest is a comma-separated list of arguments that is passed to JES. The first item must be the “destination_name”. The destination name can take the form NODE.USER. The second item must be the “class”. The third item must be the “forms”. Not all items must be specified, but the items must be specified in the proper order. The definition of “destination_name”, “class”, and “forms” is defined by JES.

For more information about the dest option, see z/OS MVS JCL Reference.

-m
This option is not implemented.

-n number
Prints number copies of each input file (the default is 1 copy).

-o printer-option
This option is not implemented.

-s
This option is not implemented.

-t
This option is not implemented.

-w
This option is not implemented.

Examples
1. To send a previously formatted file to a JES printer:

   lp filename

   You can specify more than one file name with the command.

2. The following prints the file temp prt using the default printer destination and specifying class c (where c is the locally designated class for secured information):

   lp -d ,c temp prt
   lp -d,c temp prt
   lp -d,c temp prt

   The parameters on the -d option are positional, so if you omit a destination, you must still include the comma.

Environment variables

lp uses the following environment variables:

LPDEST
   Names the output device. This variable takes precedence over PRINTER.

PRINTER
   Names the output device if LPDEST is not defined.

Localization

lp uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

Exit values

0
  Successful completion.
>0
  An error occurred.

Portability
POSIX.2, X/Open Portability Guide.
The -m, -o, -s, -t, and -w options are extensions to the POSIX standard.

Ipstat - Show status of print queues (stub command)

Format
[-drst] [-a [list]] [-c [list]] [-o [list]] [-p [list]] [-u [list]] [-v] [list] [queue_id ...]

Description
Ipstat shows the status of print queue or queues, which are specified by queue_id. If no queue_id is given, Ipstat lists information for all of the printers on the system.
Ipstat is recognized, but its functions are not supported.
If you are using the z/OS Print Server Feature, your system automatically uses that version of the Ipstat command.

ls - List file and directory names and attributes

Format
ls [-AabCcDdEFfgHikLlmMnopqRrsTtuWx1] [pathname ...]

Description
ls lists files and directories. If the pathname is a file, ls displays information about the file according to the requested options. If it is a directory, ls displays information about the files and subdirectories therein. You can get information about a directory itself using the -d option.
If you do not specify any options, ls displays only the file names. When ls sends output to a pipe or a file, it writes one name per line; when it sends output to the terminal, it uses the -C (multicolumn) format.

Note: Code sets that are aliases of each other exist which might cause the test to fail because the file inquiry operator might return an alias of the code set that you are testing.

Options
ls displays at least the file name; you can request more information with the following options:
-A
  Lists all entries including those starting with periods ( .); but excluding any . or .. entries.
-a
  Lists all entries including those starting with a period ( .).
-b
  Displays nonprintable characters as octal bytes with the form \ooo.
-C
  Puts output into columns, sorted vertically; this is the default output format to the terminal.

-c
  Uses the time of the last change of the file's attributes for sorting (-t) or displaying (-l).

-D
  Displays from directories.

-d
  Does not display the contents of named directories, but information about the directories themselves.

-E
  Displays extended attributes for regular files:
    a
      Program runs APF-authorized if linked AC=1.
    p
      Program is considered program-controlled.
    s
      Program is enabled to run in a shared address space.
    l
      Program is loaded from the shared library region.
    -
      Attribute not set
  See “Long output format” on page 402.

-F
  Puts a / after each directory name, a * after every executable file, a | after every FIFO file, a @ after every symbolic link, and a = after every socket. It also puts an & character after an external link name.

-f
  Forces the pathname argument to be a directory; turns off sorting. ls gives the ordered list of file names in a directory file. The directory file is read and the file names are listed in the same order as they are returned. The contents of a directory file are shown.

-g
  Same as -l except that it does not display owner.
  -g turns on the Long Output Format. See “Long output format” on page 402 for details.

-H
  Displays file formats for regular files:
    --
      Not specified.
    bin
      Binary data.
    rec
      Record. (File data consists of records with prefixes. The record prefix contains the length of the record that follows. From the shell command perspective, files with this format will be treated as if they were binary files.)
  Or the following text data delimiters:
    nl
      Newline character.
    cr
      Carriage return.
    lf
      Line feed.
crlf
Carriage return followed by line feed.
lfcr
Line feed followed by carriage return.
crnl
Carriage return followed by new line.


-i
Displays file serial (inode) numbers along with file names.

-k
Uses 1024 bytes for block size.

-L
Follows symbolic links.

-l
Displays permissions, links, owner, group, size, time, name. See “Long output format” on page 402 for details.

-M
Displays the security label of the file, as in this example:

```
> ls -M has_seclabel no_seclabel
SECLABEL    has_seclabel
            no_seclabel
```

ls -M does not turn on the -l option. ls -M can be used with other options. See “Long output format” on page 402 for details.

-m
Displays names in a single line, with commas separating names.

-n
Displays UID number and GID number.

-o
Same as -l except that it does not display group.
-o turns on the long output format. See “Long output format” on page 402 for details.

-p
Puts / after directory names.

-q
Displays nonprintable characters as ?.

-R
Lists subdirectories recursively.

-r
Sorts in reverse of typical order; you can combine this with other options that sort the list.

-s
Displays size in blocks, after the file serial (inode) number, but before other information. The block size is 512 bytes unless the -k option is used.

-T
Displays file tag information associated with the file. The format of this output will be similar to the output from chtag -p.

An example output:

```
> ls -T file
t  IBM-1047    T=on  file1
```
ls -T does not turn on the -l option. ls -T can be used with other options. See “Long output format” on page 402 for details.

-t
Sorts by time. By default, this option sorts the output by the modification times of files. You can change this with the -c and -u options.

-u
Uses the last access time for sorting (-t) or displaying (-l).

-W
Enables the audit bits to be displayed. This option turns on the -l option.

These bits are printed in a 6-character field directly after the field displaying the file permission bits. These 6 characters are really two groups of 3 bits each. The first group of 3 describes the user-requested audit information. The second group of 3 describes the auditor-requested audit information. Each 3 characters displayed are the read, write, and execute (or search) audit options. Each character indicates the audit option as:

s (Audit successful access attempts)
f (Audit failed access attempts)
a (Audit all accesses)
- (No audit)

-W turns on the long output format. See “Long output format” on page 402 for details.

-x
Puts output into sorted columns, with output going across the rows.

-1
Forces output to be one entry per line.

Note: When you specify options that are mutually exclusive (for example, -c and -u), the option that appears last on the command line is used.

Long output format
The output from ls -l summarizes the most important information about the file on a single line. If the specified pathname is a directory, ls displays information about every file in that directory (one file per line). It precedes this list with a status line that indicates the total number of file system blocks occupied by files in the directory (in 512-byte chunks or 1024-bytes if -k option is used). Following is a sample of the output along with an explanation:

```
  total 11
  drwxr-xr-x 3 ROOT SYS1 0 Mar 12 19:32 tmp
  drwxr-xrwx 4 ROOT SYS1 0 Mar 12 19:32 usr
  drwxr-xr-x 2 ROOT SYS1 0 Mar 12 19:32 bin
  -rwxr--r-- 1 ROOT SYS1  572 Mar 12 19:32 foo
  -rwxr--r-- 1 ROOT SYS1  640 Mar 12 19:33 abc
```

If -T is specified, file tag information is displayed first on the line.

The first character identifies the file type:

- Regular file

b Block special file (not supported for z/OS UNIX System Services)

c Character special file

d Directory
The next 9 characters are in three groups of 3; they describe the permissions on the file. The first group of 3 describes owner permissions; the second describes group permissions; the third describes other (or world) permissions. Characters that might appear are:

- **r**: Permission to read the file.
- **w**: Permission to write on the file.
- **x**: Permission to execute the file.

The following characters appear only in the execute permission (x) position of the output.

- **S**: Same as **s**, except that the execute bit is turned off.
- **s**: If in owner permissions section, the set-user-ID bit is on; if in group permissions section, the set-group-ID bit is on.
- **T**: Same as **t**, except that the execute bit is turned off.
- **t**: The sticky bit is on.

The following character appears after the permissions if the file contains extended ACL entries:

- **+**: n/a

For example:

```
ls -l file
-rwxrwxrw-+ WELLIE SYS 167 Jan 11 09:54 file
```

Use `getfacl` to display the extended ACL entries. You can set permissions with either `chmod` or `setfacl`.

After the permissions are set, `ls` displays the following (using the preceding example), in order:

- The number of links to the file.
- The name of the owner of the file or directory.
- The name of the group that owns the file or directory.
- The size of the file, expressed in bytes. For character special files, it displays the major and minor device types.
- For a file, the date and time the file was last changed; for a directory, when it was created. The `-c` and `-u` options can change which time value is used. If the date is more than 6 months old or if the date is in the future, the year is shown instead of the time.
- The name of the file or directory.
**Note:** When files owned by user ID 0 (UID=0) are transferred from any UNIX-type system across an NFS connection to another UNIX-type system, the UID changes to -2 (UID = -2). Because -2 is not a valid UID on a z/OS system, `ls` displays a -2 in place of the user name.

If `ls -E` is issued, an additional four characters follow the original 10 characters:

```
<table>
<thead>
<tr>
<th>Total</th>
<th>Mode</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-rwxr-xr-x</td>
<td>-ps-</td>
<td>ROOT</td>
<td>SYSi</td>
<td>101 Mar 12 19:32</td>
<td>her</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rwxrwxrwx</td>
<td>a-s-</td>
<td>ROOT</td>
<td>SYSi</td>
<td>654 Mar 12 19:32</td>
<td>test</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rwxr-xr-x</td>
<td>a--</td>
<td>ROOT</td>
<td>SYSi</td>
<td>40 Mar 12 19:32</td>
<td>temp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rxr-x---</td>
<td>ap-l</td>
<td>ROOT</td>
<td>SYSi</td>
<td>572 Mar 12 19:32</td>
<td>foo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rxr-x---</td>
<td>sl</td>
<td>ROOT</td>
<td>SYSi</td>
<td>640 Mar 12 19:33</td>
<td>abc</td>
<td></td>
</tr>
</tbody>
</table>
```

If `ls -H` is issued, an additional four characters follow the original 10 characters:

```
<table>
<thead>
<tr>
<th>Total</th>
<th>Mode</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-rwxr-xr-x</td>
<td>----</td>
<td>ROOT</td>
<td>SYSi</td>
<td>0 Mar 26 08:47</td>
<td>tmp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>drwxr-xr-x</td>
<td></td>
<td>ROOT</td>
<td>SYSi</td>
<td>8192 Mar 26 08:50</td>
<td>usr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rwxr-x---</td>
<td>cr</td>
<td>ROOT</td>
<td>SYSi</td>
<td>40 Mar 26 08:55</td>
<td>abc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>drwxr-x---</td>
<td>rec</td>
<td>ROOT</td>
<td>SYSi</td>
<td>80 Dec 26 09:55</td>
<td>newfmt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>drwxr-x---</td>
<td>----</td>
<td>ROOT</td>
<td>SYSi</td>
<td>0 Mar 26 08:47</td>
<td>tmp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>drwxr-x---</td>
<td></td>
<td>ROOT</td>
<td>SYSi</td>
<td>0 Mar 26 08:50</td>
<td>usr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rwxr-x---</td>
<td>cr</td>
<td>ROOT</td>
<td>SYSi</td>
<td>40 Mar 26 08:55</td>
<td>abc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rwxr-x---</td>
<td>rec</td>
<td>ROOT</td>
<td>SYSi</td>
<td>80 Dec 26 09:55</td>
<td>newfmt</td>
<td></td>
</tr>
</tbody>
</table>
```

If `ls -E` is used in conjunction with `-H`, then the four characters will follow the four characters that are normally associated with `ls -E`:

```
ls -EH
<table>
<thead>
<tr>
<th>Mode</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rwxr-xr-x</td>
<td>ap-l</td>
<td>bin</td>
<td>1 ROOT</td>
<td>SYSi</td>
<td>101 Mar 12 19:21</td>
<td>foo</td>
</tr>
</tbody>
</table>
```

If `ls -W` is issued, an additional 6 characters, in two groups of 3, follow the original 10 characters. The first group of 3 describes the user-requested audit information; the second group describes auditor-requested audit information.

```
<table>
<thead>
<tr>
<th>Total</th>
<th>Mode</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>drwxr-xr-x</td>
<td>fff---</td>
<td>3 ROOT</td>
<td>SYSi</td>
<td>0 Mar 12 19:32</td>
<td>tmp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>drwxr-xr-x</td>
<td>fff---</td>
<td>4 ROOT</td>
<td>SYSi</td>
<td>0 Mar 12 19:32</td>
<td>usr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>drwxr-xr-x</td>
<td>fff---</td>
<td>2 ROOT</td>
<td>SYSi</td>
<td>0 Mar 12 19:32</td>
<td>bin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rwxr-x---</td>
<td>fff---</td>
<td>1 ROOT</td>
<td>SYSi</td>
<td>572 Mar 12 19:32</td>
<td>foo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-rwxr-x---</td>
<td>fff---</td>
<td>1 ROOT</td>
<td>SYSi</td>
<td>640 Mar 12 19:33</td>
<td>abc</td>
<td></td>
</tr>
</tbody>
</table>
```

**Usage notes for the `ls` command**

1. To display information about a directory from a symbolic link to the directory, either add a trailing slash to the symbolic link name, or use the `-L` option. For example, if the `/etc` directory was converted into a symbolic link, issuing an `ls` on `/etc` without a trailing slash gives you the following information:

   ```
   > ls -l /etc
   lrwxrwxrwx 1 BPXROOT BIN 12 Oct 18 19:46 /etc -> $SYSNAME/etc
   ```

   However, if you add the trailing slash, the following information about `/etc` is displayed:

   ```
   > ls /etc/
   ```

**Information**

- IBM
- NetQ
- Printsrv
- TextTools
- booksrv
- bpe
- cmx
- csh.cshrc
- csh.login
- dfs
- imoisinf
- ioepdcf
- ldap
- log
- magic
- mailx.rc
- init.options
- rc
- recover
- security
- socks.conf
- startup.mk
- yylex.c
- yyparse.c
- zoneinfo
- utmpx
The same information is displayed when the `-L` option is used:

```bash
ls -L /etc
```

### Information

<table>
<thead>
<tr>
<th>IBM</th>
<th>cmx</th>
<th>ioepdcf</th>
<th>rc</th>
<th>yylex.c</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetQ</td>
<td>csh.cshrc</td>
<td>ldap</td>
<td>recover</td>
<td>yyparse.c</td>
</tr>
<tr>
<td>Printsrv</td>
<td>csh.login</td>
<td>log</td>
<td>security</td>
<td>zoneinfo</td>
</tr>
<tr>
<td>TextTools</td>
<td>dfs</td>
<td>magic</td>
<td>socks.conf</td>
<td></td>
</tr>
<tr>
<td>booksrv</td>
<td>imoisinf</td>
<td>mailx.rc</td>
<td>startup.mk</td>
<td></td>
</tr>
<tr>
<td>bpe</td>
<td>init.options</td>
<td>profile</td>
<td>utmpx</td>
<td></td>
</tr>
</tbody>
</table>

2. When issuing the `ls` command against a large directory structure, the following message might be returned:

```
FSUM6786 too many directory entries in "dir"
```

To alleviate this problem, set the `_CEE_RUNOPTS="HEAP(,,FREE)"` environment variable before invoking the `ls` command. Language Environment will free all unused storage to avoid exhausting the user heap. For more information about heap tuning, see [z/OS Language Environment Programming Reference](https://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.3.0/com.ibm.zos.v10.doc/ceplang/ceplang HEAP.html).

### Environment variables

`ls` uses the following environment variables:

- **COLUMNS**
  - Contains the terminal width in columns. `ls` uses this value to determine the number of output columns to write using the `-C` option.

- **TZ**
  - Contains the time zone to be used when displaying date and time strings.

### Localization

`ls` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_COLLATE`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_TIME`
- `LC_SYNTAX`
- `NLSPATH`


### Exit values

- **0**
  - Successful completion.
- **1**
  - Failure due to any of the following reasons:
• Out of memory.
• Inability to find a file's information.
• Too many directories.
• File or directory not found.
• Specified on the command line.

2
Incorrect command-line option

Messages
Possible error messages include:

File or directory name is not found
The requested file or directory does not exist.

Cannot allocate memory for sorting
To sort its output, ls needs to allocate memory; this message says that there was not enough memory for the sorting operation.

Too many directory entries in dir
This message appears only when ls runs out of dynamically allocated memory.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The -A, -b, -E, -f, -g, -L, -m, -n, -o, -p, -s, -W, and -x options are extensions of the POSIX standard.

Related information
Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.

ls -f, sh, tcsh

mail - Read and send mail messages

Format

mail [-e | -p] [-qr] [-f file...]
mail [-t] name...

Note: The mail utility is fully supported for compatibility with older UNIX systems. However, it is recommended that the mailx utility be used instead because it may provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

Description

mail lets you read mail sent to you and sends mail to other users. It has two modes of operation, one for reading mail and one for sending mail. If you start mail without any arguments, it checks for mail to be read and then presents the messages in read mode. If you start it with an argument or arguments, it assumes you are sending a message to the address named as the argument and enters send mode. The text of the message is taken from standard input until mail encounters either EOF or a line consisting of only a single dot (.)

For example, to read mail, enter:

mail
To send a mail message to the users Chris and Lee, enter:

```
mail chris lee
```

**Options**

The `-t` option is used only when sending mail; the others only when reading mail.

- `-e`
  Tests for the existence of mail and exits. If there is mail in the system mailbox, the return status is successful.

- `-f`
  Reads mail from file instead of the system mailbox. This option is often used to read mail saved in other files.

- `-p`
  Prints all mail to standard output without querying.

- `-q`
  Quits the mail session after an interrupt signal; normally, an interrupt ends only the message being written.

- `-r`
  Saves messages in first-in, first-out order, the reverse of the default. Normally, the most recently received message is written first.

- `-t`
  Lists the recipients at the beginning of the message (default).

**Reading mail**

When you start `mail` without arguments, `mail` checks your system mailbox for mail. If there is no mail, `mail` exits with a return code of 1; if there is waiting mail, `mail` displays the first message. (If you specify `-p` on the command line, it displays all messages.)

Commands within `mail` control how messages are handled. The following commands are available:

- `d`
  Deletes the current message.

- `m[name...]`
  Sends the current message to the specified user. If a user is not specified, the mail is sent to you.

- `p`
  Prints the message on the screen again.

- `q`
  Quits mail, storing any undeleted messages in the file `$HOME/mbox`.

- `s[file]`
  Saves the message in the specified file. If a file is not specified, `mail` saves the message in `mbox` in your home directory.

- `w[file]`
  Saves the message (same as `s`), but without header lines.

- `x`
  Exits mail without changing the mailbox file.

**ENTER (or newline)**

Displays the next message.

**! command**

Runs `command` using the shell.

**+**

Displays the next message (same as ENTER or newline).
- Displays the previous message.

* Displays a summary of internal commands.

Because the commands are read from standard input, you can create mail command files and use input redirection to have mail execute them.

**Sending mail**

To send mail, start mail with a list of addresses as arguments. Enter the text of the message, and end the message with either EOF or with a single dot (.) on a line followed by a <newline>.

The –t option inserts at the beginning of the message a list of the addresses; a path name beginning with a slash (/) is recognized as a valid address (assuming you have the correct permissions).

If the address is not valid or recognized, or if the message is interrupted (see the –q option), mail stores the message in the file dead.letter in the current directory. If it can't create dead.letter in the current directory, it creates the file in your home directory. If dead.letter already exists, the new contents overwrite the old.

The mail program modifies the message text slightly; because lines beginning with From (including the trailing space) are used to separate files in the mailbox, mail changes any lines in the message that begin with From to read >From.

**Examples**

To send the file how2mail to user Chris, enter:

```
mail chris < how2mail
```

**Usage notes**

1. Wherever the POSIX standard doesn't define the behavior of mail, this implementation resembles mailx.
2. mail doesn't require a delivery path or mechanism to the destination, though for most uses, this is preferable.

**Environment variables**

mail uses the following environment variables:

**HOME**
- Specifies your home directory; used to locate the mbox and dead.letter files.

**TZ**
- Specifies the time zone to be used in date and time strings.

**Localization**

mail uses the following localization environment variables:

- LC_CTYPE
- LC_MESSAGES
- LC_TIME
- NLS‐PATH


The ability of mail to handle double-byte characters (or even 8-bit ASCII) depends on the underlying mail transport mechanism. You should restrict all messages to the POSIX portable character set. To send messages containing double-byte characters or even binary files, encode them first with uuencode.
Files
mail uses the following files:

**dead.letter**
The most recently canceled message.

**mbox**
The default file for saving read mail, stored in the directory specified by HOME.

Exit values

0  
The session was successfully completed; if reading, there was mail.

1  
There was no mail, or the session could not be started.

2  
An error occurred after starting the session, or you supplied an invalid option, resulting in a usage message.

Portability

POSIX.2
Because this utility is due to be withdrawn from POSIX, you may want to use mailx for portable applications. The ability to write directly to a file is an extension to POSIX.

Limits

Any individual line is limited to LINE_MAX bytes; of course, transport mechanisms between systems may impose shorter limits.

Related information

mailx, uudecode, uuencode

Appendix I, “Format of the TZ environment variable,” on page 987 also explains how to set the local time zone with the TZ environment variable.

mailx - Send or receive electronic mail

Format

**mailx** [-efHiNn] [-u user] [filename]

**mailx** [-FinU] [-h number] [-r address] [-s subject] user ...

Description

**mailx** helps you read electronic mail messages. It can also send messages to users on your system, but it has no built-in facilities for sending messages to other systems.

The command line:

```
mailx [options] user user user ...
```

sends a mail message to the given users. If you do not specify any users on the command line, **mailx** lets you read incoming mail interactively. For more information, see **sendmail**.

In a double-byte locale, aliases, variables, and addresses can contain double-byte characters.

This description of **mailx** is divided into several sections:
Options
You can use the following options when reading messages:

-e
Checks to see if you have any messages waiting to be read. With this option, nothing is displayed. If you have waiting messages, mailx exits with a successful status return; otherwise, mailx exits with a failure return.

-f filename
Looks for messages in the specified file instead of in your current mailbox. If you do not specify filename, mailx reads messages from $HOME/mbox.

-H
Displays only the header summary of a message.

-N
Does not display the header summary of messages.

-u user
Looks for messages in the system mailbox of the specified user. This works only if you have read permission on the user's system mailbox.

You can use the following options only when sending messages:

-F
Records your message in a file with the same name as the first user specified on the command line. This option overrides the record variable, if it has been set.

-H number
Indicates how many “hops” a message has already made from one machine to another (in a network of machines). This option is not intended for most users; network mail software uses the option to prevent infinite loops (the same message cycling through a sequence of machines without ever getting to its intended destination).

-r address
Passes the given address to network mail software. If this option is present, it disables all input mode commands. Again, this option is not intended for most users.

-s subject
Uses the given subject string in the Subject heading line of the message. If the subject contains spaces or tab characters, the string should be enclosed in double quotation marks or single quotation marks. If you specify this option on the command line, mailx does not prompt you to enter a subject line when you type in the text of the message. The subject accepts at most LINE_MAX-10(2038) bytes. Any subjects longer than that will be truncated at 2038.

-U
Converts the address from UNIX-to-UNIX Copy Program (UUCP) style to Internet Protocol standards. This option overrides the effect of the conv variable.
This option is not supported.

You can use these options for both sending and reading messages:

- **i**
  Ignores interrupts (for example, from pressing <Break> or <Ctrl-c>.

- **n**
  Does not initialize your *mailx* session from the system's `/etc/mailx.rc` file.

**General overview**

This section describes the default behavior of *mailx*.

The simplest command to send a message is:

```
mailx address address address ...
```

where each `address` names someone who is to receive the message. The simplest kind of address is the `login name` of someone else who uses your shell.

You can also send messages as input to commands. To do this, use an address that consists of a pipe symbol (`|`) followed by a command line that invokes the appropriate command; enclose this whole address in single quotation marks. For example:

```
mailx ROBIN '|cat >save'
```

mails a message to ROBIN and also copies the message into a file called save.

After you type in the command to send a message, *mailx* asks you to enter the subject of the message (a brief description of what the message is about), and then lets you type in the text of the message. This brief description can be up to 256 characters long. Your message can consist of any number of lines and can include blank lines. When you finish entering the message, type a line consisting only of a tilde (`~`), followed by a period (`.`); then press the enter key. This tells *mailx* that the message is ready to be sent.

*mailx* puts the completed message into a file called the recipient's system mailbox. The message stays in the system mailbox until the recipient asks to read the message. At that point, the message is obtained from the system mailbox and displayed on the recipient's workstation. The message is then saved in the recipient's personal mailbox. Since this is typically a file named `mbox` in the recipient's home directory, we use the name `mbox` to represent the personal mailbox and `mailbox` for a system mailbox.

The simplest way to read incoming messages is to type the command *mailx* (with no addresses on the command line). This starts an interactive session in which *mailx* lets you read your mail and perform other operations. For example, you can display new messages, delete old ones, reply to messages, or forward them to someone else, and so on. When you are performing operations in this way, you are in command mode. When you are typing in the text of a message, you are in input mode.

A message consists of a sequence of header lines, followed by the body of the message. The header lines tell who sent the message, the time and date that the message was sent, the subject of the message, and so on. *mailx* automatically creates header lines. Some of the common header lines are:

**Cc:** `name name ...
` Stands for "carbon copies". This indicates that copies of this message are to be sent to the specified recipients. The names of these recipients appear in the header lines of everyone receiving the message.

**Bcc:** `name name ...
` Stands for "blind carbon copies". This is similar to Cc:, but the names of people receiving carbon copies do not appear in the header lines of the message. Recipients do not know that these people received a copy of the message.

**Subject:** `text`

Gives the subject of the message.

**To:** `name name ...
` Gives the names of people who were sent the message directly.
All messages are in one of the following states:

**deleted**
You used a delete, dp, or dt command to delete the message. When mailx quits, messages in this state are deleted.

**new**
The message is in the system mailbox and you have not yet read it or otherwise changed its state. When mailx quits, messages in this state are kept in your system mailbox.

**preserved**
You used a preserve command on the message. When mailx quits, messages in this state are kept in their current locations.

**read**
You used one of the following commands on the message:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>~F</td>
<td>copy</td>
</tr>
<tr>
<td>-f</td>
<td>mbox</td>
</tr>
<tr>
<td>-M</td>
<td>next</td>
</tr>
<tr>
<td>-m</td>
<td>pipe</td>
</tr>
<tr>
<td></td>
<td>Print</td>
</tr>
<tr>
<td></td>
<td>type</td>
</tr>
<tr>
<td></td>
<td>undelete</td>
</tr>
</tbody>
</table>

or you used delete, dp, or dt on the preceding message and the autoprint variable was set. When mailx quits and you are in your system mailbox, read messages are kept in your personal mailbox—unless the hold variable is set, in which case, read messages are kept in your system mailbox. If you are in your personal or a secondary mailbox when mailx quits, read messages are kept in their current location.

**saved**
You used a save or write command on the message. If the current mailbox is the system mailbox and the variable keepsave was set, messages in the state saved are saved to the mbox. If the current mailbox is the system mailbox and you used a quit or file command to exit the current mailbox, messages in the state saved are deleted from the current mailbox.

**unread**
You have run more than one mailx session with the message in the system mailbox and you have not read it or otherwise changed its state. When mailx quits, messages in this state are kept in your system mailbox.

**Command-mode subcommands**

The standard format of a command-mode subcommand is:

```
[subcommand][refs][arguments]
```

If no subcommand is specified, the default subcommand depends on the setting of the _UNIX03 variable:

- If _UNIX03=YES is set, then n[ext] is assumed.
- If _UNIX03 is unset or is not set to YES, then p[rint] is assumed.

The **refs** argument indicates the messages to which you want to apply the subcommand. mailx numbers incoming messages sequentially as they are received. The easiest way to refer to a message is to give its number. For example, the subcommand:

```
p 3
```

displays message number 3. At any point in a mailx session, there is one message that is considered the current message. This is the message you most recently did something with (for example, the one you most recently read). If you omit the **refs** argument in a subcommand that uses **refs**, the subcommand works with the current message.

You can also use special notations as the **refs** value:

**refs**

<table>
<thead>
<tr>
<th>Meaning</th>
</tr>
</thead>
</table>

---

412 z/OS: UNIX System Services Command Reference
n
Message number n

n-m
Messages n through m

.
The current message

^ The first undeleted message (or first deleted message for undelete)

$ The last message

+ Next message

- Previous message

* All messages

user
All messages from the given user

/string
All messages with string in the subject line (the case of characters in string is ignored)

:d
All deleted messages

:n
All new messages

:o
All old messages

:r
All messages that have already been read

:u
All unread messages

Several refs arguments can be specified for the same subcommand, separated by spaces. For example:

```
p alice lewis
```
displays all messages from alice plus all messages from lewis.

The arguments allowed at the end of a command-mode subcommand depend on the subcommand itself.
If a subcommand allows a file name as an argument, you can use the usual file name generation
characters in the file name (see sh).

File names, where expected, are subjected to the following transformation, in sequence:
• If the file name begins with an unquoted plus sign, and the folder variable is defined, the plus sign will
be replaced by the value of the folder variable followed by a slash. If the folder variable is unset or set to
null, the file name will be unchanged.
• Shell word expansions will be applied to the file name. If more than one path name results from this
expansion and the command is expecting one file, the multiple path names will be combined into one
argument.

The following list shows the subcommands that are recognized in command mode. In every subcommand
name, some characters are enclosed in square brackets. These characters are optional. For example, the
p[rint] command may be given as p, pr, pri, prin or print.
alias [alias [name ...]]
Sets up an address alias. If you enter a subcommand to send mail to the given alias, the messages are sent to the given list of names. For example, you might enter the subcommand:

alias joe JSMITH

From this point onward, you can address messages to joe and they are sent to jsmith. You may also set up an alias for several people, as in:

alias choir SOPRANO ALTO TENOR BASS

After you have done this, you can send messages to choir and they are sent to the names that follow choir in the command.

Alias substitution only takes place when alias is used as the whole mail address. Alias substitution doesn't take place when replying to a message that has an alias match in the addresses.

If you use only one argument, alias lists the value of that alias. For example, alias joe would display jsmith. Entering the alias subcommand without any arguments displays a list of the currently defined aliases.

Note: Aliases entered interactively remain in effect only until the end of the current interactive session. To make an alias permanent, include the alias subcommand in your startup file (see “Startup files” on page 421). See also group.

alternates name
Lists a set of alternate names for your own login name. This is useful for people who login under several different names. When you reply to a message, mailx typically sends your reply to the author of the message and all the recipients as well; however, it does not send the message to any of your alternate login names. You don’t have to worry about sending mail to yourself.

Specifying alternates without names displays your list of currently defined alternate names.

cd directory
Makes directory your new working directory. If no directory is specified, cd goes to your HOME directory.

chdir directory
Is the same as cd.

copy [refs] [filename]
Copies the messages referred to by refs into the given file. The filename must be specified. If the file does not exist, it is created.

If no refs are specified, the current message is saved. If no filename is specified, your mbox is saved.

This operation does not mark the message as saved; if it was previously unread, it is still regarded as an unread message. Thus, the original message remains in your system mailbox. See also save.

Copy [refs]
Is similar to the copy command, except that the messages referred to are saved in a file the name of which is derived from the author of the first message referred to. The name of the file is the author's name, stripped of any network addressing. If the folder variable is set, the file is saved to the specified directory. The copied messages are not marked as “saved”. If refs is not specified, the current message is copied.

delete [refs]
Deletes the specified messages from your system mailbox. If refs is not specified, the current message is deleted. After a delete operation, the current message is set to the message after the last message deleted. Deleted messages are not thrown away until you end your session with the current mailbox (see quit and file). Until then, they can be undeleted (see undelete).

discard [header...]
Does not display the given header fields when displaying a message. For example:

discard References
tells `mailx` not to display the References line at the beginning of any mail message. These header lines are retained when the message is saved; they are just not shown when the message is displayed. See also `ignore` and `retain`.

**dp [refs]**
Deletes the specified messages and then displays the message after the last message deleted. If there is no subsequent message, `mailx` displays its command prompt.

**dt [refs]**
Is the same as the dp subcommand.

**ec[ho] string ...**
Echoes the given strings (like the echo subcommand).

**e[dit] [refs]**
Lets you edit the messages that are specified by `refs`. The messages are stored in a temporary file and an editor is invoked to let you edit the file. The default editor is `ed`, but you can change this using the EDITOR environment variable.

**exit**
Quits `mailx` without changing the system mailbox. Contrast this with `quit`, which ordinarily removes from the system mailbox those messages you've read, saved, or deleted.

**file [filename]**
Quits the system mailbox (as if a `quit` subcommand were run) and then reads in the specified file as the new mailbox to examine. If no file name is specified, the default is your current mailbox.

Several special strings can be used in place of `filename`:

- `%`
  Your system mailbox.

- `%user`
  The system mailbox for user.

- `#`
  The previous file.

- `&`
  Your mbox (personal mailbox).

- `+file`
  The named file in the folder directory.

**fold[er] [filename]**
Is the same as the `file` subcommand.

**folders**
Displays the names of the files in the directory that is given by the folder variable. See “Environment variables” on page 422.

**F[ollowup] [refs]**
Replies to the first message given in `refs`; `mailx` sends this reply to the authors of every message that is given in `refs`. The Subject line is taken from the first message in `refs`. Your reply is automatically saved in a file that derives its name from the author of the message to which you are replying.

- If `_UNIX03=YES` is set, then the command overrides the record variable if record is set.
- If `_UNIX03` is unset or is not set to YES, then the command does not override the record variable.

To create your reply, `mailx` puts you into input mode, where you can use all of the input mode commands.

**f[ollowup] [ref]**
Replies to the specified message; if no message `ref` is given, you reply to the current message. Your reply is automatically saved in a file that derives its name from the author of the message to which you are replying. This overrides the record environment variable if record is set.

To create your reply, `mailx` puts you into input mode, where you can use all of the input mode commands.
mailx

**f[rom]** [refs]
Displays the header summary for the specified messages. If *refs* is not given, the current message is used.

**g[roup]** [alias [name ...]]
Is the same as the alias command.

**h[eaders]** [ref]
Displays the headers of a screenful of messages that surround the message given by *ref*. The number of lines in a screen is given by the screen variable. If no *ref* is specified, the current message doesn't change; otherwise the current message is changed to the message specified by *ref*.

**hel[p]**
Displays a summary of the command-mode subcommands.

**ho[ld]** [refs]
Retains the specified messages in your system mailbox. For example, you might decide to hold a message if you read it, but decide not to act upon it immediately. If *refs* is not specified, the current message is held. If any of the specified messages have been marked as deleted, the *hold* subcommand overrides that and still retains the messages. Subsequent delete, dp, and dt commands during the same *mailx* session can delete files that are marked for retention. See also preserve and the variables hold and keepsave.

**i[f]** code *mailx* subcommands | [el[se] *mailx* subcommands] | [en[dif]]
Is primarily intended for use in startup files. The *code* must be the character *r* or *s*. If it is *r*, the first set of *mailx* subcommands are executed if *mailx* is in receive mode, and the second set if *mailx* is in send mode. If *code* is *s*, the opposite is true. The *else* part is optional. See “Startup files” on page 421.

**ig[nore]** [header ...]
Is the same as the discard subcommand.

**l[ist]**
Displays the names of all command-mode subcommands.

**m[ail]** address ...
Sends a message to the specified recipients. *mailx* goes into input mode to let you enter the text of the message.

**mb[ox]** [refs]
Indicates that the given messages are to be saved in your *mbox* (personal mailbox) when *mailx* quits normally (that is, through the quit command as opposed to *exit*).

**n[ext]** [refs]
Goes to the next message in the mailbox that appears in the list of *refs*. For example:

```
  n  user
```

goes to the next message from the specified *user*.

**pi[pe]** [[refs] command]
Pipes the messages that are given by *refs* through the specified shell *command*. These messages are considered read. If *refs* is not specified, the current message is used. If no *command* is specified, *mailx* uses the command that is specified by the cmd variable. See “Environment variables” on page 422. If the page variable has a value, a form feed character is sent into the pipe after every message.

The subcommand | [refs] [command] is equivalent to pipe.

**pre[serve]** [refs]
Is the same as the *hold* subcommand.

**P[rint]** [refs]
Displays the specified messages on the screen. If *refs* is not specified, the current message is displayed. All header fields are displayed; the discard, ignore, and retain subcommands do not affect Print. If the *crt* variable is set to an integer, messages with more lines than that integer are "paginated" using the command that is specified by the PAGER variable.
Displays the specified messages on the screen. If \texttt{refs} is not specified, the current message is displayed. Header fields specified by \texttt{discard}, \texttt{ignore} and \texttt{retain} subcommands affect \texttt{print}. If the \texttt{crt} variable is set to an integer, messages with more lines than that integer are “paginated” using the command specified by the \texttt{PAGER} variable. For more information, see “Environment variables” on page 422.

Ends a \texttt{mailx} session. This is the usual method to leave \texttt{mailx}. Messages that have been read but not saved or deleted are stored in your \texttt{mbox} (personal mailbox). Messages that are still unread are retained in your system mailbox. Messages that have been deleted or explicitly saved in other files are discarded. Typing the end-of-file character has the same effect.

Sends a reply to the authors of each of the messages that are specified by \texttt{refs}. If \texttt{refs} is not specified, the current message is used. The Subject line of the reply message is taken from the first message in \texttt{refs}. If the record environment variable is set to a file name, your reply message is appended to the end of that file.

Normally, you use \texttt{Reply} if you want to send your reply to the author of a message, and \texttt{reply} if you want to send your reply to the author and all recipients. If set, the \texttt{flipr} environment variable reverses the meanings of the \texttt{R} and \texttt{r} commands. See “Environment variables” on page 422.

Sends a reply to the author of a specific message, and all other recipients of the message. If \texttt{ref} is not specified, \texttt{mailx} replies to the current message. If the record environment variable is set to a file name, your reply message is appended to the end of that file.

Is the same as the \texttt{Reply} subcommand.

Is the same as the \texttt{reply} subcommand.

Is the opposite of the \texttt{discard} subcommand. It tells \texttt{mailx} to display the given \texttt{header} fields when displaying a message. The comparison of \texttt{header} fields is not case sensitive. You can use \texttt{retain} to override existing \texttt{discard} and \texttt{ignore} commands. If you do not specify any \texttt{header} fields, \texttt{retain} displays a list of currently retained header fields.

Saves the specified messages in a file the name of which is taken from the author of the first message (the file name is the author's name, without any attached network addressing). If the folder variable is set, the file is saved to the specified directory.

Saves the specified messages in the given file. If \texttt{refs} is not given, the current message is added to the \texttt{mbox}. (The value of the append variable determines whether the message is added to the beginning or the end of the \texttt{mbox}. The file is created if it does not exist. If you do not specify \texttt{filename}, \texttt{mailx} saves the messages in \texttt{mbox} (your personal mailbox). A message that has been saved with \texttt{save} is normally deleted from \texttt{mailbox} when \texttt{mailx} ends (see \texttt{quit}); but see the variables \texttt{hold} and \texttt{keepsave}.

Defines a variable with the given \texttt{name} and assigns it a null value. If you omit \texttt{name}, \texttt{set} displays a list of all defined variables and their values.

Defines a variable with the given \texttt{name} and assigns it the given \texttt{value}, which may be a string or a number.

Is the same as the \texttt{unset} \texttt{name} subcommand.

Invokes the shell given by the \texttt{SHELL} environment variable.
si[ze] [refs]
Displays the size in bytes of each of the specified messages. If no refs are specified, the current message is used.

so[urce] file
Reads the specified text file, executes its contents as command-mode subcommands, and then returns to read more commands from the original source.

to[p] [refs]
Displays the first few lines of each of the specified messages. If refs is not specified, the current message is used. If the toplines variable has a numeric value, that many lines are displayed from each message; otherwise, five lines are displayed from each message.

tou[ch] [refs]
“Touches” the specified messages, making them appear to have been read. This means that when you quit mailx, the messages are saved in your mbox (personal mailbox) if they are not deleted or explicitly saved in another file. If refs is not specified, the current message is touched.

T[ype] [refs]
Is the same as the Print subcommand.

u[ndelete] [refs]
Restores previously deleted messages. When messages are deleted, they are not discarded immediately; they are just marked for deletion and are deleted when mailx ends. Until mailx ends, you can use undelete to restore the specified messages. You cannot undelete messages deleted in previous sessions. If you do not specify refs, this command restores the first deleted (but not yet undeleted) message following the current message; if no such message exists, it restores the last deleted (but not yet undeleted) message preceding the current message. If the autoprint variable is set, the last restored message is displayed. This is the only subcommand that lets you give a ref to a message that has been deleted.

U[nread] [refs]
Marks the specified messages as unread.

uns[et] name ...
Discards the specified variables.

ve[rsion]
Displays version information about mailx.

v[isual] [refs]
Edits the specified messages with a screen editor. If refs is not specified, the current message is edited. The messages are saved in a temporary file and the screen editor is invoked to edit that file. The editor used is given by the VISUAL variable. See “Environment variables” on page 422.

w[rite] [refs] filename
Writes the specified messages into the given file. If refs is not specified, the current message is written. write is the same as save, except that it does not write out the header lines and the blank line at the end of the message.

x[it]
Is the same as the exit command.

z[+|-]
Scrolls the header display forward (if z or z+ is specified) or backward (if z- is specified) one screenful.

! command
Executes the given shell command. For example:

!1c
lists all files in the current directory. The shell that will be used to run the command is given by the SHELL environment variable. See "Environment variables" on page 422.

#comment
Specifies that mailx should ignore everything from the # to the end of the line. This is useful for putting comments into startup files.

?  
Is the same as the help command (it displays a summary of the command-mode subcommands).

=  
Displays the current message number.

**Input-mode subcommands**

You can use input-mode subcommands when entering the text of a message. You must type mode subcommands at the beginning of an input line; you cannot type them in the middle of a line. By default, each input-mode subcommand begins with the tilde (~) character, called the escape character. You can use the escape variable to change the escape character, but in the documentation that follows, the tilde character is used.

~.
Marks the end of input in a mail message.

~?
Displays a summary of the input-mode subcommands.

~A
Inserts the autograph string at this point in the message. This autograph string is given by the Sign variable.

~a
Is similar to ~A, except that it uses the variable sign.

~b name ...
Adds the specified names to the blind carbon copy list.

~c name ...
Adds the specified names to the carbon copy list.

~d
Reads in the dead.letter file.

~e
Invokes an editor on the message that you have composed. The editor variable determines the editor that is invoked.

~F [refs]
Forwards the given messages. The text of the messages is inserted at this point in the message you are composing. The message headers are also inserted with all header fields regardless of the discard, ignore, and retain subcommands. This is valid only when you entered mailx in command mode and then went into input mode to compose a message.

~f [refs]
Is similar to ~F except that the header fields included are determined by the discard, ignore, and retain subcommands.

~h
Prompts you to enter the following header lines:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Cc</th>
<th>Bcc</th>
<th>To</th>
</tr>
</thead>
</table>

For some of these, mailx displays an initial value for the header. You can edit this initial value as if you had just typed it in yourself, using backspaces and line deletes.

~i name
Inserts the value of the named variable followed by a newline at this point in the message.
~M [refs]
Inserts the text of the specified messages at this point in the message. If refs is not specified, the current message is used. Messages inserted in this way have each line prefixed with the value of the indentprefix variable. The message headers are also inserted with all header fields included regardless of the discard, ignore, and retain subcommands. This is valid only when you entered mailx in command mode and then went into input mode to reply to a message.

~m [refs]
Is similar to ~M, except that the header fields are determined by the discard, ignore, and retain subcommands.

~p
Displays the message being composed.

~q
Quits input mode as if you had interrupted the message. If you have already composed part of a message, the partial message is saved in the dead.letter file; the description of the dead environment variable has more information.

~r filename
Reads in the contents of the specified file and adds that text at this point in the message.

~s text
Sets the Subject line to the given text.

~t address address ...
Adds the given addresses to the To: list (people who will receive the message).

~v
Invokes a screen (visual) editor on the message that you have composed. The VISUAL variable determines the editor that is invoked.

~w file
Writes the current text of your message to the specified file. The header lines for the message are not written.

~x
Quits in the same way as ~q, except that the message is not saved in the dead.letter file.

~< filename
Is the same as the ~r command.

~< !command
 Runs the given shell command and adds the standard output of that command at this point in the message. For example, your message might contain:

```
My program is giving me this odd output:
~< !prog
What do you think is causing it?
```

~: mail_command
Runs the given command-mode mail_command. This is valid only when you entered mailx in command mode and then went into input mode to compose a message.

~! mail_command
Is the same as the ~: command.

~! command
Runs the given shell command. For example, you can use:

```
>~! ls
```
to get a list of files in the working directory. The shell that is invoked to run the command is given by the SHELL environment variable. If the bang variable is set, mailx replaces each unescaped exclamation mark (!) in command with the command run by the previous command or ~! command escape.
~ command
Pipes the current message through the specified shell command. If the command ends with a successful exit status, the output of the command replaces the text of the current message. For example:

```
~|fmt
```
fills and justifies the lines of your message and replaces the message with the formatted message. ~| uses the shell that is given by the SHELL environment variable to run command.

Startup files
When you run mailx in command mode, mailx does the following:

• Sets all variables to their default values. mailx processes command-line options, using them to override any corresponding default values.
• Imports appropriate external environment variables, using them to override any corresponding default values.
• Reads commands from the system startup file, /etc/mailx.rc. This sets up variable values and definitions that should be common to all users. If you do not want mailx to read the system startup file, use the –n option on the mailx command line.
• After reading and processing the system startup file, mailx does the same with a personal startup file, which is MAILRC by default. This is a file in your HOME directory. The name of the file is .mailrc.

Startup files typically set up display options and define aliases. However, any command is valid in a startup file except for the following:

• Copy
• edit
• followup
• Followup
• hold
• mail
• preserve
• reply
• Reply
• respond
• Respond
• shell
• visual
• !

If a line in a startup file contains an error or an incorrect command, the rest of the startup file is ignored. mailx ignores blank lines in a startup file.

Examples
The following example composes and sends a message to several users. Items shown in bold are output by mailx.

```
mailx JEAN
Subject: Greetings
This is just a short note to say hello.
~c JUAN JOHN JOHANN
~.
```
On the first line, the message is just addressed to jean. The ~c line adds more people who will receive copies of the message.

Environment variables

A large number of variables are used to control the behavior of mailx. These environment variables are divided into two classes: those that always come from the external environment, and those that may be set up in either the external environment or within a mailx session.

The following variables always come from the external environment; they can be changed inside a mailx session, except where marked.

DEAD
  Gives the name of a file that can be used as the dead.letter file. Partial messages are saved in this file if an interrupt or error occurs during creation of the message or delivery. By default, the name of this file is $HOME/dead.letter.

EDITOR
  Gives a command, possibly with options, that is run when using the command mode edit or the input mode ~e. The default is ed (see “ed - Use the ed line-oriented text editor” on page 272 for more information about ed).

HOME
  Gives the name of your home directory. This cannot be changed inside mailx.

LISTER
  Gives a command, possibly with options, that mailx invokes when displaying the content of the folder directory for the folders subcommand. If this variable is null or unset, mailx uses ls. By default, this variable is unset.

LOGNAME
  Gives your login name.

MAIL
  Gives the path name of the user's mailbox file for purposes of incoming mail notification.

MAILDIR
  Gives the name of the directory where system mailboxes are stored. If this is not set, the default is /usr/mail. The actual name of a user's system mailbox is derived in a system-dependent way by combining MAILDIR and the user's login name. For mailx to work properly, the MAILDIR directory must exist.

MAILRC
  Gives the name of your startup file. This cannot be changed inside mailx. By default, MAILRC has the value $HOME/.mailrc. For more information about startup files, see “Startup files” on page 421.

MBOX
  Gives the name of your mbox (personal mailbox) file. Messages that have been read but not saved elsewhere are saved here when you run quit (but not when you run exit). The default is $HOME/mbox.

PAGER
  Gives a command, possibly including options. mailx sends display output through this command if the output is longer than the screen length given by crt. The default value is more (see “more - Display files on a page-by-page basis” on page 463 for more information about more).

SHELL
  Gives a command, possibly with options. mailx assumes that this command is a command interpreter. mailx invokes this command interpreter whenever it is asked to run a system command (for example, through the ! command-mode command). The default is sh (see “sh: Invoke a shell” on page 588 for more information about sh).

TERM
  Contains the name of the terminal type. This cannot be changed inside mailx.
TZ
This variable may determine the time zone used to calculate date and time strings written in *mailx*. This cannot be changed inside *mailx*.

_UNIX03
For more information about the effect of _UNIX03 on this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

VISUAL
Gives a command, possibly with options, that *mailx* invokes when using the command-mode visual subcommand or the input mode ~v subcommand. The default is vi (see “vi - Use the display-oriented interactive text editor” on page 807 for more information about vi).

The HOME and LOGNAME variables must be set before you enter *mailx*. Otherwise, *mailx* will not work properly. The TZ variables can only be set before you enter *mailx*. If not set or set to null, a default time zone (“UTC0”) will be used. These variables are typically set during shell login. (You can login with theOMVS TSO/E command, telnet, rlogin, or ssh.) If you do not log in, you must set the variables in some other way, using the commands:

```
export LOGNAME=name
export HOME=directory
```

The remaining variables can be set in the external environment or in the course of a *mailx* session. You can set or change the value of a variable with the set subcommand; you can discard a variable with the unset subcommand. You may find it convenient to create a startup file that sets these variables according to your preferences; this eliminates the need to set variables each time you enter *mailx*.

Many of the following variables represent on-off options. If you set the variable itself (to any value), the option is turned on. To turn the option off, you can unset the variable, or set a variable consisting of no followed by the name of the original variable. For example, setting autoprint turns the autoprint option on, and setting noautoprint turns it off.

**allnet**
Assumes that network addresses with the same login component refer to the same person. Network addresses typically consist of several components, giving information that lets a mail server identify a machine on the network, a route to that machine, and the login name of a user on that machine. *mailx* assumes that the login name is the last component. For example:

```
print name
```

displays all messages that originated from the same login name, regardless of the rest of the network address. The default is noallnet, where different addresses are assumed to be different users, even if the login name components are the same.

**append**
Appends messages to the end of the mbox file (your personal mailbox) after termination. The default is noappend; messages are placed at the beginning of the mbox file instead of the end.

**ask**
Prompts you for a Subject: line when composing a message (if you have not already specified one with the –s option). This option is on by default; to turn it off, set noask. ask is the same as asksub. noask is the same as noasksub.

**askbcc**
Prompts you for a Bcc: list when composing a message. The default is noaskbcc; you are not prompted.

**askcc**
Prompts you for a Cc: list when composing a message. The default is noaskcc; you are not prompted.

**asksub**
Prompts you for a Subject: line when composing a message (if you have not already specified one with the –s option). This option is turned on by default; to turn it off, set noasksub. asksub is the same as ask. noasksub is the same as noask.
**autoprint**
Automatically displays the last message deleted with the `delete` subcommand or the last message undeleted with `undelete`. The default is noautoprint; you are not shown messages that you delete or undelete.

**bang**
Records shell commands run inside the `mailx` session (for example, through the `~!` input-mode command). Then, if you issue a shell command and the shell command contains a `!` character, `mailx` replaces that character with the command line for the previous shell command. The default is nobang, in which case a `!` in a shell command line is not treated specially.

**cmd**
Contains a command, possibly with options. This specifies a default command line to be used for the command-mode `pipe` subcommand. For example:

```
set cmd="cat"
```

pipes messages through `cat` when the `pipe` subcommand is invoked. The default is nocmd.

**crt**
Contains an integer number. If a message has more than this number of lines, the message is piped through the command given by the PAGER variable, whenever the message is displayed. `crt` is not set; the default is nocrt.

**debug**
Enables verbose diagnostics for debugging. Messages are not delivered. The default is nodebug.

**dot**
Accepts a line consisting only of a dot (.) to indicate the end of a message in input mode. Thus . is equivalent to `~.`. The default is nodot. If ignoreeof is set, `mailx` ignores a setting of nodot; the period is the only way to end input mode.

**escape**
Gives the character that is used to begin input-mode subcommands. The default is the tilde (~). If this variable is unset, tilde is used as the escaping character. If this variable is set to null, `mailx` disables command escaping.

**flipr**
Reverses the meanings of the `R` and `r` subcommands. The default is noflipr. See also Replyall.

**folder**
Contains the name of a directory in which `mailx` saves mail files, if you use a plus sign (+) in front of the filename. This lets you specify a standard directory for saving mail files. Whenever you specify a filename for a `mailx` command, putting a plus sign (+) in front of the name specifies that the file is to be accessed or stored in the folder directory.

If the value of folder begins with a slash, it is taken as an absolute path name; otherwise, `mailx` assumes that the directory is directly under your HOME directory. The default is nofolder. If you want to use + in file names that appear on the `mailx` command line itself (as opposed to commands in a `mailx` session), you must make folder an exported shell environment variable.

**header**
Displays a summary of message headers at the beginning of a `mailx` command-mode session. This is the default.

**hold**
Keeps all messages in your system mailbox instead of saving them in your personal `mbox`. The default is nohold.

**ignore**
Ignores interrupts received while composing a message. The default is noignore.

**ignoreeof**
Ignores end-of-file markers found while entering a message. The message can be ended by “." or ~. on a line by itself. The default is noignoreeof.
indent
Contains a string that mailx uses as a prefix to each line in messages that ~m and ~M insert. The default is one tab character.

indentprefix
As with indent, contains a string that mailx uses as a prefix to each line in messages that ~m and ~M insert. The default is one tab character. If both indent and indentprefix are set, indentprefix takes precedence.

keep
Does not remove your system mailbox if the mailbox contains no messages. The mailbox is truncated to zero length—that is, it is merely emptied, although it still exists. If the default nokeep is in effect, empty mailboxes are removed.

keepsave
Keeps messages in your system mailbox even if they have been saved in other files. The default, nokeepsave, deletes messages from the system mailbox if they have been saved elsewhere.

mailserv
Identifies the mail server being used for remote mail.

metoo
When replying to a message with your login name in the recipient list, sends a reply to all other recipients, the author, and you. If nometoo is set, you are not to be sent the reply. The default is nometoo.

onehop
Attempts to send replies directly to the recipients instead of going through the original author's machine. When you reply to a message, your reply is sent to the author and to all recipients of the message. On a network, mailx normally specifies the recipient addresses so that all the replies go to the original author's machine first, and then on to the other recipients. The default is noonehop.

outfolder
Causes files that are used to record outgoing messages (see the description of record) to be located in the directory that is given by folder unless folder contains an absolute path name. The default is nooutfolder.

page
Tells the pipe subcommand to insert a form-feed character after each message that it sends through the pipe. The default is nopage.

prompt
Contains a string that mailx displays to prompt for output in command mode. The default is a question mark followed by a space (? ).

quiet
Does not display the opening message and version number when mailx begins a session. The default is noquiet.

record
Contains a file name where every message you send is to be recorded. If record is not an absolute path name and the outfolder variable has not been set, the file is located in the current directory. If the outfolder variable is set, the file is located in your folder directory. The default is norecord.

replyall
Reverses the senses of the reply and Reply subcommands (so that reply replies only to the author of a message, and Reply replies to the author and all other recipients). See also flipr.

save
Saves messages in your dead.letter file if they are interrupted while being composed. The name of your dead.letter file is given by the dead variable. Setting nosave disables this automatic save feature. The default is save.
screen
Gives the number of headers that are to be displayed by the headers and z subcommands. If screen is not specified, the current window size shall be used to determine the number of headers displayed.

sendmail
Contains a command, possibly with options, that mailx invokes to send mail. The default is /usr/lib/tsmail. It can be any command that takes addresses on the command line and message contents on standard input.

sendwait
When sending a message through a network, mailx waits for the mail server to finish before returning to your session. Normally, it just submits the message to the server and then returns immediately. The default is nosendwait.

showto
When displaying a header summary, displays the recipient's name instead of the author's for messages where you are the author. The default is noshowto.

sign
Contains a string that is inserted into a message when you use the input mode ~a subcommand. mailx interprets \n and \t in this string as the newline and tab characters, respectively. The default is nosign.

Sign
Contains a string that is inserted into a message when you use the input mode ~A subcommand. The default is noSign.

topolines
Gives the number of header lines that the top subcommand is to display. The default is 5.

Files
mailx uses the following files:

/etc/mailx.rc
System-wide startup file.

$MAILRC
Personal startup file. By default, MAILRC has the value $HOME/.mailrc.

$HOME/mbox
Default location to save read messages. You can choose a different file by assigning the file name to the environment variable MBOX.

$MAILDIR
Directory containing system mailboxes. By default, this is /usr/mail. The system programmer must create the MAILDIR directory if it does not exist. See z/OS UNIX System Services Planning for information about electronic mail.

If you use a system mailbox directory other than /usr/mail, identify it in the $MAIL environment variable in /etc/profile. See the section on customizing /etc/profile in z/OS UNIX System Services Planning.

$HOME/dead.letter
Default location to save partial letters.

Localization
mailx uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
Exit values

0

- Successfully sent. (However, this does not guarantee that the mail was successfully received.)
- 0 is returned if -e is specified and mail was found.

1

- Returned if -e is specified and mail was not found or an error occurred. Also returned to indicate failure due to any of the following:
  - There is no mail to read.
  - Inability to create temporary file name or temporary file.
  - Receipt of user interrupt while message was being composed.
  - Inability to determine the user's identity.

2

- Failure due to any of the following:
  - Missing number after -h.
  - Missing address after -r.
  - Missing subject after -s.
  - Missing user after -u.
  - Incorrect command-line option
  - Use of interactive options when not using command interactively.

Portability

POSIX.2, X/Open Portability Guide, UNIX System V.

UNIX System V has a compatible mailx utility, whereas Berkeley Software Distribution (BSD) has a similar utility, which is known as Mail.

The -F, -r, and -U options; the Copy, echo, followup, Followup, Save, Unread, and version subcommands; and the allnet, conv, mailserv, onehop, replyall, sendmail, and sendwait variables are extensions of the POSIX standard.

Related information

echo, ed, sh, vi

make - Maintain program-generated and interdependent files

Format

```make [-EeinpqrstuVvx] [-k|-S] [-c dir] [-f file] ...
[macro definition ...] [-D macro definition ...] [target ...]
```

Description

make helps you manage projects containing a set of interdependent files, such as a program with many source and object files, or a document built from source files, macro files, and so on. make keeps all such
files up-to-date with one another. If one file changes, `make` updates all the other files that depend on the changed file.

**Note:** This implementation of `make` features the .POSIX special target to provide maximum portability. When you specify this target, `make` processes the makefile as specified in the POSIX standard. For details, see the description of .POSIX in “Targets” on page 429.

In a double-byte locale, environment variable values, here-documents, and the command line can all contain environment values.

### Options

- **-c dir**
  Attempts to change into the specified directory when `make` starts up. If `make` cannot change to the directory, an error message is printed. This option is useful for recursive makefiles when you are building in a different directory.

- **-D macro definition**
  Define `macro` on the command line before reading any `makefile`. Use the same form as a normal macro definition (`macro=string`). If you use this option, `make` assigns the value to the macro before reading the makefile; any definition of the same macro contained in the makefile supersedes this definition.

  **Note:** `make` uses any macros defined in this way before reading any makefile, including the startup file. This allows you to define a startup file by providing a value for `MAKESTARTUP` on the command line:

  ```
  make -D MAKESTARTUP=$HOME/project/startup.mk
  ```

- **-E**
  Suppresses reading of the environment. If you do not specify either -E or -e, `make` reads the environment before reading the makefile.

- **-e**
  Reads the environment after reading the makefile. If neither -E nor -e are specified, `make` reads the environment before reading the makefile, except for the SHELL environment variable, which you must explicitly export. This option does not affect the value of `MAKEFLAGS`.

- **-f file**
  Uses `file` as the source for the makefile description. `make` ignores the makefiles specified as prerequisites to the .MAKEFILES target. If you specify a minus sign (-) in place of `file`, `make` reads the standard input. (In other words, `make` expects you to enter the makefile from the terminal or to redirect it from a file.) You can use more than one -f option.

- **-i**
  Tells `make` to ignore all errors and continue making other targets. This is equivalent to the .IGNORE attribute or macro.

- **-k**
  Makes all independent targets, even if an error occurs. Ordinarily, `make` stops after a command returns a nonzero status. Specifying -k tells `make` to ignore the error and continue to make other targets, as long as they are not related to the target that received the error. `make` does not attempt to update anything that depends on the target that was being made when the error occurred.

- **-n**
  Displays the commands that `make` would execute to update the chosen targets, but does not actually execute any recipe lines unless they have a plus sign (+) command prefix. `make` displays recipe lines with an at sign (@) command prefix on standard output (stdout). For more information about recipe lines, see z/OS UNIX System Services Programming Tools.

  With group recipes, `make` displays the commands that it uses to update a given target, but it also executes the commands.

  **Note:** z/OS `make` supports group recipes, but traditional implementations of `make` do not. A group recipe signifies a collection of command lines fed as a unit to the command interpreter. By contrast,
**make** executes commands in a normal recipe one by one. For more information about group recipes, see *z/OS UNIX System Services Programming Tools*.

If **make** finds the string $ (MAKE) in a recipe line, it expands it, adds –n to the MAKEFLAGS, and then executes the recipe line. This enables you to see what happens when recursive calls are made to **make**. The output correctly shows line breaks in recipes that are divided into several lines of text using the <newline> sequence.

- **p**  
  Prints the makefile after it has been processed to include macro and target definitions. This display is in human-readable form useful for debugging, but you cannot use it as input to **make**.

- **q**  
  Checks whether the target is up-to-date. If it is up-to-date, **make** exits with a status of 0; otherwise, it exits with a status of 1 (typically interpreted as an error by other software). No commands are run when –q is specified.

- **r**  
  Does not read the startup file. Various control macros and default rules will not be defined.

- **S**  
  Ends **make** if an error occurs during operations to bring a target up-to-date (opposite of -k). This is the default.

- **s**  
  Specifies that recipe commands, warning messages, or touch messages (see the –t option) not be displayed. This is equivalent to the .SILENT attribute or macro.

- **t**  
  Touches the target to mark them as up-to-date, but only executes commands to change a target if the target has a plus sign (+) command prefix. **make** does not touch up-to-date targets or targets that have prerequisites but not recipes. **make** displays a message for each touched target file indicating the file name.

- **u**  
  Forces an unconditional update: **make** behaves as if all the prerequisites of the given target are out of date.

- **V**  
  Prints the version number of **make** and a list of built-in rules.

- **v**  
  Causes **make** to display a detailed account of its progress. This includes what files it reads, the definition and redefinition of each macro, metarule and suffix rule searches, and other information.

- **x**  
  Exports all macro definitions to the environment. This happens just before **make** begins making targets (but after it has read the entire makefile).

**Targets**

A **target** is normally a file that you want to ensure is up-to-date with the files on which it is dependent (the prerequisites). For example, you might want to check to see if a is based on the most recent version of the corresponding source code. If it is not, then have the source code recompiled to get an up-to-date version. In this case, the compiled program file is the target and the corresponding source files are prerequisites (that is, the files on which a target is dependent).

**make** updates all targets that are specified on the command line. If you do not specify any target, **make** updates the targets in the first rule of the makefile. A target is out of date if it is older than any of its prerequisites (based on modification times) or if it does not exist. To update a target, **make** first recursively ensures that all the target’s prerequisites are up-to-date, processing them in the order in which they appear in the rule. If the target itself is out of date, **make** then runs the recipe that is associated with the target. If the target has no associated recipe, **make** considers up-to-date.
**BRACEEXPAND**

This target may have no prerequisites and no recipes that are associated with it. If set, the target enables the outdated brace expansion feature used in older versions of `make`. Older `make` commands would expand a construct of the following form, beginning with each token in the token list:

```
string1{token_list}string2
```

Older `make` commands would append `string1` to the front of each token in the list, and `string2` to the end of each token in the list. A more productive means for achieving the same result with modern versions of `make` relies on macro expansion with `prefix` and `suffix` modifiers:

```
$(TOKEN_BASE:^"prefix:"suffix"
```

The double quotation marks are required. Brace expansion is an outdated feature that is available in past versions of `make`.

**CYCLECHECK**

This special target cannot have any prerequisite or recipes that are associated with it. If set, it determines how `make` treats circular dependencies (see Circular dependencies). You can specify one of five attributes with this target. If you specify more than one attribute, an error message results. The five attributes are:

- **SILENT**
  - `make` remains silent about any within-rule and between-rule circular dependencies, removes the offending dependency from the list of prerequisites, and continues.

- **WARNTARG**
  - `make` issues warnings for named targets with circular dependencies. If the name of the dependency is the same as the named target, it is removed from the list of prerequisites and `make` continues. This is the default behavior if `.CYCLECHECK` is not specified or is specified with no attributes.

- **WARNALL**
  - `make` issues warnings for all within-rule circular dependencies regardless of whether the target is being built or not and for all between-rule circular dependencies for the named targets. The offending dependency is removed from the list of prerequisites and `make` continues.

- **FATALTARG**
  - `make` treats all circular dependencies for named targets as fatal errors. It issues an error message and exits.

- **FATALALL**
  - `make` treats all within-rule circular dependencies as fatal errors regardless of whether the target is being built or not. It also treats all between-rule circular dependencies for named targets as fatal errors. `make` issues an error message and exits.

For example, to set the circular dependency check to `make`'s default, use the rule:

```
.CYCLECHECK .WARNTARG:
```

**DEFAULT**

This target has no prerequisites, but it does have a recipe. If `make` can apply no other rule to produce a target, it uses this rule if it has been defined.
.ERROR
make runs the recipe that is associated with this target whenever it detects an error condition.

.EXPORT
All prerequisites associated with this target that correspond to macro names are exported to the environment at the point in the makefile at which this target appears.

.GROUPEPILOG
make adds the recipe that is associated with this target after any group recipe for a target that has the .EPILOG attribute.

.GROUPPROLOG
make adds the recipe associated with this target after any group recipe for a target that has the .PROLOG attribute.

.IMPORT
make searches in the environment for prerequisite names specified for this target and defines them as macros with their value taken from the environment. If the prerequisite .EVERYTHING is given, make reads in the entire environment (see –e and –E options).

.INCLUDE
make reads one or more additional makefiles (specified in the prerequisite list), as if their contents had been inserted at this point. If the prerequisite list contains more than one file, make reads them in order from left to right.

make uses the following rules to search for extra makefiles:

- If a relative file name is enclosed in quotation marks, or is not enclosed with angle brackets (< and >), make looks in the current directory. If the file isn't present, make then looks for it in each directory specified by the .INCLUDEDIRS special target.
- If a relative name is enclosed with angle brackets (< and >), make only searches in directories specified by the .INCLUDEDIRS special
- If an absolute path name is given, make looks for that file and ignores the list associated with the .INCLUDEDIRS special target.

.INCLUDEDIRS
The list of prerequisites specified for this target defines the set of directories to search when including a makefile.

.MAKEFILES
The list of prerequisites is the set of files to try to read as the user makefile. These files are made in the order they are specified (from left to right) until one is found to be up-to-date. This is the file that is used.

.NOAUTODEPEND
Disables the autodependency feature when building libraries. When this special target is used, only library members that have been explicitly given as dependents are considered prerequisites.

.POSIX
make processes the makefile as specified in the POSIX.2POSIX.2 draft standard. This target may have no prerequirecipes that are associated no recipes with it. This special target must appear before the first non-comment line in the makefile. If this special target is present, the following facilities are disabled:

- All recipe lines are run by the shell, one shell per line, regardless of the setting of SHELLMETAS.
- Metarule inferencing is disabled.
- Conditionals are disabled.
- Dynamic prerequisites are disabled.
- Group recipes are disabled.
- Disables brace expansion (set with the .BRACEEXPAND special target).
- make does not check for the string $(MAKE) when run with the -n options specified.
make uses the recipe of this target to remove any intermediate files that it creates if an error is encountered before the final target is created. This .REMOVE target only deletes files that satisfy all of the following criteria:

- The file didn't exist when make began running.
- The file is named as an intermediate target, produced by invoking a metarule that was produced by transitive closure.
- The file is not explicitly named in the makefile.
- The generated target doesn't have the .PRECIOUS attribute.
- The file is a prerequisite of a rule that is actually used.

.SOURCE
The prerequisite list of this target defines a set of directories to check when trying to locate a target file name. make defaults to creating target files in the same directory that it finds the source file.

.SOURCE.x
Same as .SOURCE, except that make searches the .SOURCE.x list first when trying to locate a file matching a target with a name that ends in the suffix .x.

.SUFFIXES
mk appends the prerequisite list of this target to the set of suffixes used when trying to infer a prerequisite for making a target using suffix rules. If you specify no prerequisites, make clears the list of suffixes, effectively disabling suffix rules from that point on.

A name of the form library(member) indicates a member of a library. The library portion is a target with the .LIBRARY attribute, and the member portion is a prerequisite of the library target.

A name of the form library((entry)) indicates the library module that contains the given entry point. Once again, the library portion is a target with the .LIBRARY attribute. make regards the library member that contains the entry point entry as a prerequisite of the library target.

Makefiles
A makefile is a text file that describes the dependencies between various files. It normally contains a list of targets and identifies the prerequisites on which each depends. It also contains a series of instructions, called recipes, which describe the actions to be taken if a given target is out of date with its prerequisites.

By default, if you do not specify the -f option, make looks for a file in your current directory named makefile. If it does not find this file, it searches your current directory for a file named Makefile. If make finds either file, it uses this file as your makefile.

You can change the default makefiles with the .MAKEFILES special target. This target is already specified in the startup.mk file. See “Targets” on page 429 for more information.

Makefile contents
Comments begin with the pound (#) character and extend to the end of the line. make discards all comment text.

Inside makefiles, you can split long lines over several lines of text. To do this, put a backslash (\) at the very end of the line. You can use this technique to extend comments as well as recipe lines and macro definitions, for example.

If a rule or macro definition must contain a # character, use \#; otherwise, make mistakes the # for the beginning of a comment. Also, if a macro definition must contain a single $ character, use $$.

File names that contain a colon must always be enclosed in quotes, as in:

"a:target" : "a:prereq"

You can use a target that has prerequisites but no recipes to add the given prerequisites to that target's list of prerequisites. You can preface any recipe line with a command prefix immediately after the tab
character –, @, + or all three. The method of entering tab characters using an ISPF editor is discussed in z/OS UNIX System Services User's Guide.

- indicates that make is to ignore nonzero exit values when it runs this recipe line.
@ indicates that make is not to display the recipe line before running it.
+ tells make to always run this line, even when –n, –p, or –t is specified.

Group recipes begin with [ in the first non-white-space position of a line, and end with ] in the first non-white-space position of a line. Recipe lines in a group recipe need not have a leading tab. make executes a group recipe by feeding it as a single unit to a shell. If you immediately follow the [ at the beginning of a group recipe with one of –, @ or +, they apply to the entire group in the same way that they apply to single recipe lines.

**Macro definitions**

Macro definitions can appear on the command line or in makefiles. Macro definitions on the command line overrule definitions in makefiles; makefile definitions never overrule command-line definitions. Macro definitions on the command line may not have any white space between the macro name and the = character.

Macro definitions may take several forms. macro = string is the usual form. If string contains macro references, make does not expand them when the macro is defined, but when the macro itself is expanded.

```
macro := string
```

expands macros inside string before assigning a value to macro.

```
macro += string
```

adds string to the previous value of macro.

You can use any amount of white space on both sides of macro operators. make defines the name macro to have the value string and replaces it with that value whenever it is used as $(macro) or ${macro} within the makefile. It is possible to specify a $(macro_name) or ${macro_name} macro expansion, where macro_name contains more $(. . .) or ${. . .} macro expansions itself.

Typically, make does not include white space at the beginning and end of string in the definition of macro; however, it never strips white space from macros imported from the environment.

If you want to include white space in a macro definition specified on the make command line, you must enclose the definition in quotes.

**make** resol...
Possible modifiers are:

- `^"string"`
  - Prefix tokens

- `+"string"`
  - Suffix tokens

- `b`
  - File portion of all path names, without suffix

- `d`
  - Directory portion of all path names

- `f`
  - File portion of all path names, including suffix

- `l`
  - All characters mapped to lowercase

- `s/pat/string/`
  - Simple pattern substitution (you can use any character to separate the pattern from the substitution text)

- `suffix=string`
  - Suffix replacement

- `t"separator"`
  - Tokenization with given separator

- `u`
  - All characters mapped to uppercase

You can specify macro modifiers in either uppercase or lowercase. For example, the macro assignment:

```
test = D1/D2/d3/a.out f.out d1/k.out
```

produces the following expansion:

```
$(test:d)                 →→ D1/D2/d3 . d1
$(test:b)                 →→ a f k
$(test:f)                 →→ a.out f.out k.out
$(test:db){}              →→ D1/D2/d3/a f d1/k
$(test:s/out/in{)}       →→ D1/D2/d3/a.in f.in d1/k.in
$(test:t:"+"{)}           →→ a.out+f.out+k.out
$(test:t:"+"{)}           →→ D1/D2/d3/a.out+f.out+d1/k.out
$(test:1)                 →→ d1/d2/d3/a.out f.out d1/k.out
```

Runtime macros can take on different values for each target.

- `$@`
  - The full target name. When building a normal target, this macro evaluates to the full name of the target. When building a library, it expands to the name of the archive library. For example, if the target is:
    ```
    mylib(member)
    ```
  - `$@` expands to:
    ```
    mylib
    ```

- `$%`
  - The full target name. When building a normal target, this macro evaluates to the full name of the target. When building a library, it expands to the name of the archive member. For example, if the target is:
    ```
    mylib(member)
    ```
  - `$%` expands to:
    ```
    mylib
    ```
$% expands to:

    member

$&
The list of all prerequisites.

$?
The list of all prerequisites that are newer than the target.

$^ The list of all prerequisites taken from the list specified on the rule line of the recipe where the $^ appears.

$< In inference rules, it evaluates to the single prerequisite that caused the execution of the rule. In normal rules it evaluates the same as $?.

$> The name of the library if the current target is a library member.

$$* The target name with no suffix ($(%:db)) or the value of the stem in a metarule.

The constructs $$@, $$%, $$>, and $$* can appear in a prerequisite list as dynamic prerequisites. $$@ stands for the target currently being made. For example:

    fred : $$@.c
    fred : fred.c

are equivalent. The construct can be modified, as in:

    fred.o : $$(@:b).c

The runtime macros can be modified by the letters D and F to indicate only the directory portion of the target name or only the file portion of the target name. (The working directory is represented by a dot.) If define.h is the only prerequisite that is newer than the target, the macros $$?D and $$?F expand to dot ( .) and to define.h.

If you are building a library, $$% stands for the name of the archive member being made. If you are building a normal target, $$% stands for the name of the target currently being made.

$$* stands for the name of the current target being made, but with no suffix.

If you are building a library, $$> stands for the name of the archive library being made. If you are not building a library, $$> is not valid.

Rules
The general format of a rule is:

```
targets [attributes] ruleop [prerequisites]
[:<tab> recipe]
```

where the items enclosed in square brackets are optional. (This is just a documentation convention; you do not actually enter the square brackets.) The parts of the rule are described as follows:

**targets**
One or more target names.

**attributes**
A list, possibly empty, of attributes to apply to the list of targets.

**ruleop**
An operator token, usually a colon (:), that separates the target names from the prerequisite names and may also affect the processing of the specified targets.
prerequisites
A list of zero or more names on which the specified targets depend.

recipe
A command to execute to update targets. May follow on the same line as the prerequisites, separated from them by a semicolon. If such a recipe is present, make takes it as the first in the list of recipe lines defining how to make the named targets. Additional recipe lines may follow the first line of the rule. Each subsequent recipe line must begin with a tab character.

The possible rule operators are listed as follows:

targets : prereqs
Is a simple rule definition. For explicit targets, at most one simple rule may have a recipe, in contrast with the :: rule operator, whose description follows.

targets :! prereqs
Executes the recipe for the associated targets once for each recently changed prerequisite. In simple rules, the recipe is executed only once, for all recently changed prerequisites at the same time. The $< macro expands to the current recently changed prerequisites if it appears in rules with this rule operator.

targets :^ prereqs
Inserts the specified prerequisites before any other prerequisites already associated with the specified targets.

targets :- prereqs
Clears the previous list of prerequisites before adding the new prerequisites.

targets :: prereqs
If no prerequisites are specified, the targets are always remade. Otherwise it is used for multiple rules applying to the same targets. Each rule can specify a different set of prerequisites with a different recipe for updating the target. Each rule is treated independently; the target is remade for each rule with recently changed prerequisites, using the corresponding recipe.

targets : | prereqs
Can only be used in metarules. It tells make to treat each metadependency as an independent rule. For example:

```make
$0 : | archive/%.c rcs/%.c /srcarc/RCS/%period.c
recipe...
```

is equivalent to

```make
$0 : archive/$.c
recipe:
$0 : rcs/%.c
recipe:
$0 : /srcarc/rcs/%.c
recipe:
```

Circular dependencies
There are two types of circular dependencies: within-rule and between-rule.

A within-rule circular dependency occurs when the target's name is included in the list of prerequisites for that target. For example,

```
c.o : a.o b.o c.o
```

is a within-rule circular dependency. make detects a within-rule circular dependency when it is parsing the makefile to build the dependency tree.
A *between-rule circular dependency* occurs when you have two targets, each of which includes the other's name in its prerequisite list. For example,

```
a.o : b:o
b:o : a.o
```

is a between-rules circular dependency. *make* detects a between-rule circular dependency when it is processing the dependency tree built during the parse phase.

Typically, *make* only detects circular dependencies for those targets actually being built. When a circular dependency is encountered, *make* issues a warning message, removes the offending prerequisite from the list, and continues parsing the makefile. You can use the `.CYCLECHECK` special target to alter *make*'s treatment of circular dependencies.

**Inference rules**

With inference rules you can specify general rules for building files rather than creating a specific rule for each target.

*make* provides two forms of inference rules: suffix rules and metarules. It includes suffix rules to ensure compatibility with older makefiles. Metarules, however, provide a more general mechanism for specifying *make*'s default behavior. They provide a superset of the functionality of suffix rules.

*make* searches all metarules before using suffix rules.

*make* uses the inference rules to infer how it can bring a target up-to-date. A list of inference rules defines the commands to be run. The default `startup.mk` file contains a set of inference rules for the most common targets. You can specify additional rules in the makefile.

When *make* finds no explicit target rule to update a target, it checks the inference rules. If *make* finds an applicable inference rule with an out-of-date prerequisite, it runs on that rule's recipe. See “Targets” on page 429 for information about the `.DEFAULT` special target).

**Metarules**

Metarules have one target with a single percent symbol that matches an arbitrary string called the stem; the `%` in a dependency stands for the stem.

The inference rule to update a target matching pattern `p1%s1`, where `p1` and `s1` are prefix and suffix strings of the target, having a prerequisite `p2%s2`, where `%` is the stem from the target, is specified as a rule:

```
p1%s1 : p2%s2 ; recipe...
```

Either the prefix or suffix string may be empty.

**Transitive closure**

Metarules provide a mechanism that allows several metarules to chain together to eventually create the target. This mechanism is called *transitive closure*. For example, if you have metarules:

```
%.o : %.c
    ... rule body...
```

and:

```
%.c : %.y
    ... rule body...
```

When you specify:

```
make file.o
```
**make** uses the first metarule to look for `file.c`. If it can't find an explicit rule to build `file.c`, it again looks through the metarules and finds the rule that tells it to look for `file.y`.

**make** allows each metarule to be applied only once when performing transitive closure to avoid a situation where it loops forever. (For example, if you have the rule:

```plaintext
% : %.c  
  ... rule body ...
```

the command:

```plaintext
make file
```

causes **make** to look for `file.c`. If the metarules were not restricted and `file.c` did not exist, then **make** would look for `file.c.c`, and then `file.c.c.c`, and so on. Because each metarule is applied only once, this can't happen.)

Transitive closure is computed once for each metarule head the first time the pattern matches a target. When transitive closure is computed, all the computed rules are added to the rule set for that metarule head. For example, if you have the rules:

```plaintext
% : %.o  
  recipe 1...
%.o : %.c  
  recipe 2...
```

and you are making `file`, this target matches successfully against `%` causing transitive closure to be computed for `%`. As a result of this computation, a new rule is created:

```plaintext
% : %.c  
  recipe 2...  
  recipe from .REMOVE target for %.o, if not .PRECIOUS  
  recipe 1...
```

which is executed if `file.o` doesn't exist. When the computation for the rule head has been done, it is marked as transitive closure computed. Since all possible new rules have been added to the rule set the first time the computation is done, it is not necessary to do it again: Nothing new is added. The term transitive closure is adapted from the mathematical set theory.

**Note:** In set theory, if you have a set composed of pairs \((a,b)\) and \((b,c)\), then the set would be transitively closed if \((a,c)\) is also in the set.

The best way to understand how this works is to experiment with little **make** files with the `-v` flag specified. This shows you in detail what rules are being searched, when transitive closure is calculated, and what rules are added.

### Order of rule generation

Since transitive closure allows **make** to generate new rules, it is important to understand the order in which this is done:

1. **make** searches for explicit rules in the order in which they appear, so explicit rules always take precedence.
2. **make** reads metarules in the order in which they appear in the makefile. The first rule that appears in the makefile is the first one checked.
3. New explicit metarules (as distinct from metarules generated by transitive closure) replace old ones.

   In other words, if your makefile contains an explicit rule like this one, it replaces the default rule in `startup.mk`:

   ```plaintext
   %%$O : %.c  
   rule1
   ```

   If you use the `-v` option, **make** prints a warning when it replaces a metarule.
4. When transitive closure is calculated, the new metarules generated are added to the end of the list of possible metarules. Thus, make always finds the explicit rules first, so they take precedence over generated rules. You can use the -v option to see what rules make generates and the order in which they appear.

5. make performs two passes through the rules. On the first pass it tries to find a match with an explicit rule in the makefile; if this does not succeed, make performs a second pass to find a match with an existing file.

Suffix rules

make treats targets that begin with a period and contain no slashes or percent signs as suffix rules. If there is only one period in the target, it is a single suffix inference rule. Targets with two periods are double-suffix inference rules. Suffix rules do not have prerequisites but do have commands associated with them.

When make finds no explicit rule to update a target, it checks the suffix of the target (.s1) to be built against the suffix rules. make examines a prerequisite based on the basename of the target with the second suffix (.s2) appended, and if the target is out of date with respect to this prerequisite, make runs the recipe for that inference rule.

Metarules take precedence over suffix rules.

If the target to be built does not contain a suffix and there is no rule for the target, make checks the single suffix inference rules. The single suffix inference rules define how to build a target if make finds a rule with one of the single suffixes appended. A rule with one suffix .s2 defines how to build target from target.s2. make treats the other suffix (.s1) as null.

For a suffix rule to work, the component suffixes must appear in the prerequisite list of the .SUFFIXES special target. You can turn off suffix rules by placing the following in your makefile:

```make
.SUFFIXES:
```

This clears the prerequisites of the .SUFFIXES target, which prevents suffix rules from being enacted. The order that the suffixes appear in the .SUFFIXES rule determines the order in which make checks the suffix rules.

The search algorithm used for suffix rules depends on whether the .POSIX special target is specified. When .POSIX is specified, the following steps describe the search algorithm for suffix rules:

1. Extract the suffix from the target. If that target does not have a suffix, go to step 6.
2. Is it in the .SUFFIXES list? If not, quit the search.
3. If it is in the .SUFFIXES list, look for a double suffix rule that matches the target suffix.
4. If there is a match, extract the base name of the file, add on the second suffix, and determine if the resulting file exists. If the resulting file does not exist, keep searching the double suffix rules.
   
   If the resulting file does exist, use the recipe for this rule.
5. If a successful match is not made, the inference has failed.
6. If the target did not have a suffix, check the single suffix rules in the order that the suffixes are specified in the .SUFFIXES target.
7. For each single suffix rule, add the suffix to the target name and determine if the resulting file name exists.
8. If the file name exists, execute the recipe that is associated with that suffix rule. If the file name doesn't exist, continue trying the rest of the single suffix rules. If a successful match is not made, the inference has failed.

When the .POSIX special target is not specified, make handles suffix rules in the same manner as traditional implementations of make. The following steps describe the search algorithm for suffix rules in this situation.

1. Extract the suffix from the target. If that target does not have a suffix, go to step 8.
2. Is it in the .SUFFIXES list? If not, then quit the search.
3. If it is in the .SUFFIXES list, look for a double suffix rule that matches the target suffix.
4. If you find one, then extract the base name of the file, add on the second suffix and see if the resulting file exists. If it does, go to step 7. If not, continue with step 5.
5. Is there an inference rule for the resulting file? If yes, run the recipe that is associated with that rule (which should describe how to make the file exist) and go to step 7.
6. Search for the next double-suffix rule that matches the target suffix and return to step 4. If the double-suffix rules are exhausted, then the inference has failed.
7. Use the recipe for the target rule.
8. If the target did not have a suffix, then check the single-suffix rules in the order that the suffixes are specified in the .SUFFIXES target.
9. For each single-suffix rule, add the suffix to the target name and see if the resulting file name exists.
10. If the file exists, then run the recipe that is associated with that suffix rule. If it doesn't exist, continue trying the rest of the single-suffix rules.
11. If a successful match is not made, then the inference has failed.

**make** also provides a special feature in the suffix rule mechanism for archive library handling. If you specify a suffix rule of the form:

```
.suf.a:
  recipe
```

the rule matches any target having the LIBRARY attribute set, regardless of what the actual suffix was. For example, if your makefile contains the rules:

```
.SUFFIXES: .a .c
  echo adding $< to library $@
```

then if mem$0 exists, then the following command:

```
make "mylib(mem.o)"
```

causes:

```
adding mem.o to library mylib
```

to be printed.

**Attributes**

**make** defines several target attributes. Attributes can be assigned to a single target, a group of targets, or to all targets in the makefile. Attributes affect what **make** does when it needs to update a target. You can associate attributes with targets by specifying a rule of the form:

```
attribute_list : targets
```

This assigns the attributes in attribute_list to the given targets. If you do not specify any targets, the attributes apply to every target in the makefile. You can also put attributes inside a normal rule, as in:

```
targets attribute_list : prerequisites
```

The recognized attributes are:

- **.EPILOG**
  Insert shell epilog code when running a group recipe associated with any target having this attribute set.
- **.IGNORE**
  Ignore an error when trying to make any target with this attribute set.
.LIBRARY
  Target is a library.

.PRECIOUS
  Do not remove this target under any circumstances. Any automatically inferred prerequisite inherits this attribute.

.PROLOG
  Insert shell prolog code when running a group recipe associated with any target having this attribute set.

.SETDIR
  Change the working directory to a specified directory when making associated targets. The syntax of this attribute is .SETDIR=path, where path is the path name of desired working directory. If path contains any : characters, the entire attribute string must be quoted, not just the path name.

.SILENT
  Do not echo the recipe lines when making any target with this attribute set, and do not issue any warnings. You can use any attribute with any target, including special targets.

Control macros

make defines a number of control macros that control make’s behavior. When there are several ways of doing the same thing, control macros are usually the best. A control macro that has the same function as a special target or attribute also has the same name.

Macros that are said to be defined internally are automatically created by make and can be used with the usual $(name) construct. For example, $(PWD) can be used to obtain the current directory name.

Recognized control macros are:

DIRSEPSTR
  Contains the characters used to separate parts in a path name and can be set by the user. make uses the first character in this string to build path names when necessary.

.EPILOG
  If assigned a nonnull value, the .EPILOG attribute is given to every target.

GROUPFLAGS
  Specifies option flags to pass to GROUPSHELL when make invokes it to run a group recipe.

GROUPSHELL
  Gives the path name of the command interpreter (shell) that make calls to process group recipes.

GROUPSUFFIX
  Specifies a string for make to use as a suffix when creating group recipe files to be run by the command interpreter.

.IGNORE
  If this is assigned a nonnull value, make assigns the .IGNORE attribute to every target.

INCDPTH
  This is the current depth of makefile inclusion. It is set internally.

MAKE
  This is set by the startup file and can be changed by the user. The standard startup file defines it as:

$(MAKECMD) $(MFLAGS)

The MAKE macro is not used by make itself, but the string $(MAKE) is recognized when using the -n option for single-line recipes.

MAKECMD
  This is the name with which make was invoked.

MAKEDIR
  This is the full path name of the initial directory in which make began execution.
MAKEFLAGS
The MAKEFLAGS macro contains all the options (flags) and macros specified in the MAKEFLAGS environment variable plus all of the options and macros specified on the command line, with the following exceptions.

- Specifying –c, –f, or –p in the environment variable results in an error
- These same options specified on the command line do not appear in the MAKEFLAGS macro.

Options in the MAKEFLAGS environment variable may have optional leading dashes and spaces separating the options. These are stripped out when the MAKEFLAGS macro is constructed.

Note: make always reads the MAKEFLAGS environment variable before reading the makefile. The –E and –e options do not affect this.

MAKESTARTUP
This has the default value:

/etc/startup.mk

To change this value, you can set the MAKESTARTUP environment variable before running make. You can also specify a value for this control macro on the command line if you use the -D option:

make -DMAKESTARTUP=$HOME/project/startup.mk

Because make processes command-line macros after reading the startup file, setting this macro on the command line does not have the desired effect.

MFLAGS
This is the same as MAKEFLAGS, except that it includes the leading switch character.

NULL
This is permanently defined to be the null string.

.PRECIOUS
If this is assigned a nonnull value, make assigns the .PRECIOUS attribute to every target.

.PROLOG
If this is assigned a nonnull value, make assigns the .PROLOG attribute to every target.

PWD
This is the full path name of the working directory in which make is executing.

SHELL
Specifies the full path name of the command interpreter that make calls to process single-line recipes, when necessary. make passes recipe lines to this shell only if they contain one or more of the characters given in SHELLMETAS; otherwise, it runs them directly. By default, the value of the SHELL environment variable does not affect the value of this macro; however, you can use the .IMPORT special target to assign the environment variable's value to this macro. You can also use the .EXPORT special target to assign this macro's value to the SHELL environment variable.

SHELLFLAGS
Specifies option flags to pass to the shell when invoking it to runs a single-line recipe.

SHELLMETAS
Specifies a list of metacharacters that can appear in single recipe lines. If make finds any metacharacter, it invokes the recipe using the shell specified by SHELL; otherwise, it runs the recipe without the shell.

.SILENT
If this is assigned a nonnull value, make assigns the .SILENT attribute to every target.

Making libraries
A library is a file containing a collection of object files. To make a library, you specify it as a target with the .LIBRARY attribute and list its prerequisites. The prerequisites should be the object members that are to go into the library.
make tries to handle the old library construct format in a sensible way. When it finds `lib(member)`, it declares the `lib` portion as a target with the .LIBRARY attribute and the `member` portion as a prerequisite of the `lib` target. To make the library properly, old makefile scripts using this format must name the `lib` as a target and must try to bring it up-to-date. The same thing happens for any target of the form `lib((entry))`. These targets have an additional feature in that the `entry` target has the .SYMBOL attribute set automatically.

### Conditional expressions

Specify the conditional expression as follows:

```
.IF expression
... if text ...
.ELSE
... else text ...
.END
```

or:

```
.IF expression
... if text ...
.ELSIF expression2
... elsif text ...
.ELSE
... else text ...
.END
```

The .ELSE or .ELSIF portion is optional, and you can nest the conditionals (that is, the text may contain another conditional). The .IF, .ELSE, .ELSIF, and .END conditionals must start in the first column of the line. `expression` or `expression2` can have one of three forms:

- `string` is true if the given string is nonnull,
- `string == string` is true if the two strings are equal, and:
- `string != string` is true if the two strings are not equal. Typically, one or both strings contain macros, which make expands before making comparisons. make also discards white space at the start and end of the text portion before the comparison. This means that a macro that expands to nothing but white space is considered a null value for the purpose of the comparison. If a macro expression needs to be compared with a null string, compare it to the value of the macro `$NUL$`.

The text enclosed in the conditional construct must have the same format that it would have outside the conditional. In particular, make assumes that anything that starts with a tab inside the conditional is a recipe line. This means that you cannot use tabs to indent text inside the conditional (except, of course, for recipe lines, which always begin with tabs).

### Files

make uses the following file:

```
/etc/startup.mk
```

The default startup file containing default rules.

### Environment variables

make uses the following environment variables:
MAKEFLAGS
Contains a series of make options that are used as the default options for any make command. You can specify the options with or without leading minus signs (-) and blanks between them. It can also include macro definitions of the form usually found on the command line.

MAKESTARTUP
Contains the path name of the make stamp file. By default, make uses the file /etc/startup.mk as its startup file. To use a different file, set this environment variable before running make.

SHELL
Contains a name of a command interpreter. To assign this value to the SHELL control macro, use the .IMPORT special target. You can also use the .EXPORT special target to assign the value of the SHELL macro to the environment variable.

Localization
make uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH


Exit values
If a command in a recipe line fails (exits with a nonzero status), make returns the exit status of that command. Because most commands use exit status values between 0 and 10, make uses exit status values below 10 only for failures that do not run recipe lines.

0
Successful completion

1
Returned if you specified –q and file is not up-to-date

2
Failure due to any of the following:
• Unknown command-line option.
• Missing argument to option, such as no file name for -f.

126
Recipe command was not executable.

127
Recipe command was not found.

129–254
make was interrupted by a signal; the error code is the signal number ORed with 128. For example, SIGINT is frequently signal 1; the return code from make is 128|1, or 129.

255
Failure due to any of the following:
• Macro cannot be redefined
• Macro variables not assigned with :=
• Special target cannot be a prerequisite
• Too many makefiles specified
• Configuration file not found
• No makefile present
• Missing .END for .IF
• No target
• Inability to return to directory
• Too many open files
• Open failed
• File not found
• Inability to change directory
• No more memory
• Line too long
• Circular macro detected
• Unterminated pattern string
• Unterminated replacement string
• Token separator string not quoted
• Unterminated separator string
• Expansion too long
• Suffix too long
• Unmatched quote
• .IF .ELSE ... END nesting too deep
• .ELSE without .IF
• Unmatched .END
• Inference rules resulting in circular dependency
• No macro name
• Write error on temp file
• Target not found, and cannot be made
• Inability to make NAME
• <+ diversion unterminated
• <+ diversion cannot be nested
• <+ missing before +>
• Incomplete rule recipe group detected
• Inability to mix single and group recipe lines
• Unmatched ] found
• Macro or rule definition expected but not found
• Name too long
• Inability to determine working directory
• Only one NAME attribute allowed in rule line
• Multiple targets not allowed in % rules
• Special target must appear alone
• Duplicate entry in target list
• Syntax error in % rule, missing % target
• Duplicate entry in prerequisite list
• Missing targets or attributes in rule
• Multiply defined recipe for target
• Empty recipe for special target
• Imported macro NAME not found in environment
• No .INCLUDE files specified
• Include file NAME, not found
• NAME ignored on special target
• Attributes possibly ignored
• Inability to find member defining SYMBOL((NAME))
• Incorrect library format
• Inability to touch library member
• SHELL macro not defined
• Too many arguments
• Inability to export NAME
• Inability to open file
• Circular dependency detected
• Inability to stat /
• Inability to stat .
• Inability to open ..
• Read error in ..
• Metarule too long: "rule"

Usage notes
1. The length of a single makefile script line cannot exceed 32768 characters.
2. The length of an argument string cannot exceed 32768 characters.
3. The length of a macro name is truncated after 256 characters.
4. When the .SETDIR special target is used, make checks the file attributes of targets and prerequisites on every pass through a rule. This can significantly increase the number of system accesses.
5. In a double-byte environment, any character interpreted by make can be a double-byte character, including those in macro definitions and targets.
6. In a double-byte locale, if make encounters an incorrect double-byte sequence, it ends with an error message.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The following features of make are enhancements to POSIX.2:
• The options: -cdir, -D macro definition, -E, -u, -V, -v, and -x.
• The -n option has enhanced functionality not covered by the standard; for more information, see the -n option and the POSIX special target for make.
• The runtime macros: $&, $^ , $>.
• The dynamic prerequisites: $$$, $$$>, $$*, $$@.
• All macro expansions.
• Macro assignments of the following form:

  macroname := stringassigned
  macroname += stringassigned
• Brace expansion.
• Backslash continuation.
• The quoting mechanism, as in the following example:

```
"a:target" : "a:prerequisite"
```
• All rule operators except the colon (:).
• Conditionals.
• Metarules.
• All `make` attributes except `.IGNORE`, `.PRECIOUS`, `.SILENT` (referred to in POSIX.2 as special targets).
• All `make` special targets except `.DEFAULT`, `.POSIX`, `.SUFFIXES` (referred to in POSIX.2 as special targets).
• All `make` macros except `SHELL` (referred to in POSIX.2 as control macros).

Related information

`c89`, `cc`, `c++`, `makedepend`


**makedepend - Generate source dependency information**

**Format**

```
makedepend [-S directory] [-W m,option[,option]...] sourcefile ([sourcefiles]...)
makedepend [-s directory]
makedepend [-W m,a]
makedepend [-W m,c89 | -W m,cc]
makedepend [-W m,file(MakeFile) | -W m,f(MakeFile)]
makedepend [-W m,list(FileName) | -W m,list (FileName)]
makedepend [-W m,o(ObjSuffix)]
makedepend [-W m,p(ObjPrefix)]
makedepend [-W m,s(String)]
makedepend [-W m,showinc | -W m,show]
makedepend [-W m,type(c|C(t1,t2,...))] | -W m,t(c/C(t1,t2,...))]
makedepend [-W m,V(OSVvRr) | -W m,V(zOSVvRr)]
makedepend [-W m,w(Width)]
```

For z/OS UNIX `makedepend` `[c89|cc|c++options]`:

```
makedepend [+ -]
makedepend [-D name[=value]]
makedepend [-I dir1,[dir2]...]
makedepend [-O, -O (-1), -2, -3]
makedepend [-U name]
makedepend [-W phase,option[,option]...]
```

**Description**

As of z/OS V1R11, consider using the z/OS XL C/C++ -qmakedep compiler option instead of the `makedepend` tool. The -qmakedep compiler option improves the accuracy of source dependency information and is intended to replace the stand-alone `makedepend` tool. For more information about -qmakedep, see *z/OS XL C/C++ User’s Guide*. 
The `makedepend` tool is used to analyze source files and determine source dependencies. `makedepend` calls files, which are directly or indirectly included by a source file, "dependencies." If the `makedepend -W m,-list` option is specified, this tool produces a listing file with the following topics:

- The list of compiler options and variables applied to all C source.
- The list of compiler options and variables that are applied to all C++ source.
- The list of `makedepend` options applied.
- The list of include and source search paths.
- Messages.
- Message summary.
- Statistics (in other words, total number of source files processed, number of ignored sources files, and so forth).

**Options**

- **-S directory**
  Specifies the directory or directories where you can locate the source files. The default location for source files is the current directory, "/".

- **-W m,a**
  Instructs `makedepend` to append the source dependencies to the end of the makefile rather than replacing any existing ones. If `-W m,a` is not specified, then `makedepend` will erase any source dependencies after the marker line and write the new determined source dependencies instead. If there are no existing makefiles, then this option is ignored.

- **-W m,c89 | -W m,cc**
  Instructs `makedepend` to use either the c89 or the cc compiler mode for the c source files. The c89 mode is the default. The c89/cc mode is overridden if the c++ [+-] option is specified.

- **-W m,makefile (MakeFile) | -W m,makefile (MakeFile)**
  Specifies the name of the makefile to which `makedepend` writes the determined source dependencies. If this option is specified on the `makedepend` command line, then the string value of the `Makefile` is used as the name for the makefile. Otherwise, `makedepend` will search in the current directory for a file named "makefile". If no "makefile" exists, then `makedepend` searches for a file named "Makefile". If no "Makefile" exists, then `makedepend` creates a new file with the name "makefile" in the current directory and writes the default marker string (see `–W m, s(String)` at the beginning of the new file. If file (./) is specified, then the option is ignored silently.

- **-W m,list(FileName) | -W m,list(FileName)**
  Instructs `makedepend` to generate a listing file with the specified `FileName` name. The name `depend.1st` is the default file name if `FileName` is not specified with the `–W m,list` option. If `–W m,list(./)` is specified, the default listing file name (depend.1st) is used. If the `–W m,list` option is not specified, listings are not generated.

- **-W m,osuffix(ObjSuffix)**
  Specifies a suffix (file name extension) for the object file names in the source file dependencies. If the environment variable `_OSUFFIX` is defined, then its value will be the default. If it is not defined, the default suffix is o.

- **-W m,oprefix(ObjPrefix)**
  Prefixes object file names in the source dependencies with a path name. The default object file name prefix is an empty string.

- **-W m,s(String)**
  Specifies a new string literal to be used as a marker in the output makefile. All source dependencies are placed after that marker. The default marker string value is "# DO NOT DELETE THIS LINE, makedepend depends on it." If the `–W m,s(String)` is specified on the `makedepend` command line, then the marker line and anything after it will be erased from the output makefile, the new marker string literal will be written instead, and the newly determined source dependencies will be written after the new marker line. If both `–W m,a` and `–W m,s(String)` are specified on the `makedepend` command line, then `–W m,s(String)` will be ignored if a makefile exists.
The following options correspond to the z/OS UNIX `makedepend` command line, as either C source files if the C is used with the type, or as C++ source files if the C is used. Default types are as follows:

- Any source file with a file name extension of c will be treated as a C source file. If the `-W m, type(c(t1,t2,...))` option is specified on the `makedepend` command line, then any source file with a file name extension that belongs to the set {c, t1, t2,...} will be treated as a c file. Notice that the types {t1,t2,...} that are specified with the `-W m, type option are added to the default c file name extension type.

- Any source file with a file name extension of C, cpp, or cxx will be treated as a C++ source file. If the `-W m, type(C(t1,t2,...))` option is specified on the `makedepend` command line, then any source file with a file name extension that belongs to the set {C, CPP, cpp, CXX, cxx, t1, t2,...} is treated as a C++ file. Notice that the types {t1,t2,...} that are specified with the `-W m, type option are added to the default C file name extension types.

- If both `-W m,type(c(...))` and `-W m,type(C(...))` options are specified on the `makedepend` command line with conflicting file names, then whichever option is specified last becomes the overriding value, including the default file types. For example, when both `-W m, type(c(t1,t2))` and `-W m, type(C(c,t1,t3))` are specified, only files with extension t2 will be treated as c files and files with extensions {c, C, cpp, Cpp, cxx, CXX, t1, t3} will be treated as C++ files. When `-W m, type(c(cpp,t1,t2))` and `-W m, type(C(t1,t2))` options are specified, files with extensions `{c, cpp}` will be treated as c source files and files with extensions `{C, CPP, cxx, CXX, t1, t2}` will be treated as C++ source files.

For C source files, if the environment variables `{_CSUFFIX} and `{_CSUFFIX_HOST` are defined, the variable value updates the default value. The default C source file extension is c.

For C++ source file, if the environment variables `{_CXXSUFFIX} and `{_CXXSUFFIX_HOST` are defined, its value updates the default value. The default C++ source file extensions are {C, CPP, cpp, CXX, cxx}. For example, if `{_CXXSUFFIX}` is defined as `{cdd}` and the default C++ source file extensions are {C, CPP, cpp, CXX, cxx}, then the resulting set would be {C, CPP, cpp, CXX, cxx, cdd}.

```
-W m, V(OSVvRr) | -W m, V(zOSVvRr)
```

Specifies the compiler version that will be used, where v and r represent the compiler’s version and release respectively. The default version is the current C/C++ compiler version if `{_CVERSION}` is not defined. If `{_CVERSION}` is defined, then its value is used as the default compiler version. This option is used to set the _COMPILER_VER_ macro.

```
-W m, w(Width)
```

Sets the maximum line width of the output source dependencies lines. The default value is 78.

The following options correspond to the z/OS UNIX `c89`, `cc`, `c++` compiler options.

```
-+  
```

Specifies that all source files are to be recognized as C++ source files. All `file.s`, `file.o`, and `file.a` files will continue to be recognized as assembler source, object, and archive files respectively. However, any `C file.c` or `file.i` files will be processed as corresponding C++ file.C or `file.i` files, and any other file suffix which would otherwise be unrecognized will be processed as a `file.C` file.

```
-D name=[value]  
```

Defines a C or C++ macro for use in compilation. If only `name` is provided, a value of 1 is used for the macro it specifies.

```
-I dir1[,dir2]...  
```

Note: The -I option is an uppercase I, not a lowercase L.
-I specifies the directories to be used during compilation in searching for include files (also called header files).

Absolute path names specified on #include directives are searched exactly as specified. The directories specified using the -I option or from the usual places are not searched.

If absolute path names are not specified on #include directives, then the search order is as follows:

1. Include files enclosed in double quotes (") are first searched for in the directory of the file containing the #include directive. Include files enclosed in angle-brackets (< >) skip this initial search.
2. The include files are then searched for in all directories specified by the -I option, in the order specified.
3. Finally, the include files are searched for in the usual places. (See Usage Note “1” on page 452 for a description of the usual places.)

You can specify an MVS data set name as an include file search directory. Also, MVS data set names can explicitly be specified on #include directives. You can indicate both by specifying a leading double slash (/ /). For example, to include the include file DEF that is a member of the MVS PDS ABC.HDRS, code your C or C++ source as follows:

```c
#include <//'abc.hdrs(def)'>
```

MVS data set include files are handled according to z/OS XL C/C++ compiler conversion rules (see Usage Note “1” on page 452). When specifying an #include directive with a leading double slash (//), the specified name is paired only with MVS data set names specified on the -I option. That is, when you explicitly specify an MVS data set name, any hierarchical file system (HFS) directory names specified on the -I option are ignored.

-0, –O (–1), –2, –3

Specifies the level of compiler optimization (including inlining) to be used. The level –1 (number one) is equivalent to -O (letter capital O). The level –3 gives the highest level of optimization. The default is –0 (level zero), no optimization and no inlining, when not using IPA (Interprocedural Analysis).

-Uname

Undefines a C or C++ macro specified with name. This option affects only macros defined by the -D option, including those automatically specified by c89/cc/c++.

-Wphase,option[,option]...

Specifies options to be passed to the steps associated with the compile, assemble, or link-editing phases of c89/cc/c++. The valid phase codes are:

0

Specifies the compile phase (used for both non-IPA and IPA compilation).

c

Same as phase code 0.

I

Enables IPA (Interprocedural Analysis) optimization.

**Note:** I is an uppercase I, not a lowercase L.

Unlike other phase codes, the IPA phase code I does not require that any additional options be specified, but it does allow them. In order to pass IPA suboptions, specify those suboptions using the IPA phase code. For example, to specify that an IPA compile should save source line number information, without writing a listing file, specify:

```
c89 -W I,list file.c
```

To specify that an IPA link-edit should write the map file to stdout, specify:

```
c89 -W I,map file.o
```
Note: c89/cc/c++ options other than the ones listed are ignored by makedepend.
Any compiler option can be passed to makedepend through the -W option. For more information about
the compiler options, see z/OS XL C/C++ User's Guide.

Examples

1. makedepend file1.c file2.c
   Imagine you are compiling two files, file1.c and file2.c, and each includes the header file header.h. The
   header.h file includes the files def1.h and def2.h. When you run the command makedepend file1.c
   file2.c, makedepend parses file1.c and consequently, header.h, and then def1.h and def2.h. It then
decides that the dependencies for this file are:
   • file1.o: header.h def1.h def2.h
   • file2.o: header.h def1.h def2.h

2. Imagine you are compiling a file, file1.c, and it includes the header file header.h. The header.h file
includes the files def1.h and def2.h. When you run the command makedepend file1.c, makedepend
parses file1.c and consequently, header.h, and then def1.h and def2.h. It then decides that the
dependencies for this file are:
   file1.o: header.h def1.h def2.h

Environment variables

makedepend uses the following environment variables.

{ACCEPTABLE_RC}
Used by c89/cc/c++ to determine the maximum allowed return code (result) of any step (compile,
assemble, IPA link, prelink, or link-edit). If the result is between zero and this value (inclusive), then it
is treated internally by makedepend exactly as if it were a zero result. The default value is 4.

{CLASSLIB_PREFIX}
Provides a prefix for the data sets used during the compilation and execution phases. For
makedepend, the focus is the {CLASSLIB_PREFIX}.SCLBH.+ data set that contains the z/OS C/C++
Class Library include files.

{CSUFFIX}
Used by c89/cc/c++ to recognize a C source file. The default value is c.

{CSUFFIX_HOST}
Used by c89/cc/c++ to recognize a C source file. The default value is C.

{CXXSUFFIX}
Used by c++ to recognize a C++ source file. The default is C. This variable is only supported by the c++
command.

{CXXSUFFIX_HOST}
Used by c++ to recognize a C++ source data set. The default is CXX. This variable is only supported by
the c++ command.

{CSYSLIB}
Used for system library data set concatenation, which resolves #include directives during
compilation.

{INCDIRS}
Provides directories used by c89/cc/c++ as a default place to search for include files during
compilation (after searching {_INCDIRS} and before searching {_CSYSLIB}).

{INCLIBS}
The directories used by c89/cc/c++ as a default place to search for include files during compilation
(after searching {_INCDIRS} and before searching {_INCLIBS} and {_CSYSLIB}).

{OSUFFIX}
Provides a suffix by which c89/cc/c++ recognizes an object file.
The version of the C/C++ compiler to be invoked by `c89/cc/c++`. The setting of this variable allows `c89/cc/c++` to control which C/C++ compiler program is invoked. It also sets default values for other environment variables. The format of this variable is the same as the result of the Language Environment C/C++ Run-Time Library function _librel(). The default value is the result of the C/C++ Run-Time library _librel() function. Actual variable names are: _C89_CVERSION, _CC_CVERSION, _CXX_CVERSION

NO_CMDOPTS

Controls how the compiler processes the default options set by `c89`. Setting this variable to 1 reverts the compiler to the behavior that was available prior to z/OS V1R5, when the compiler was unable to distinguish between the `c89` defaults and the user-specified options. Setting this variable to 0 (zero) results in the default behavior where the compiler is now able to recognize `c89` defaults. The default value is: 0 (zero). Actual variable names are: _C89_NO_CMDOPTS, _CC_NO_CMDOPTS, _CXX_NO_CMDOPTS

Localization

`makedepend` uses the LC_ALL localization environment variable, which specifies the locale to be used to override any values for locale categories specified by LANG or certain LC_ variables.


Usage notes

1. MVS data sets may be used as the usual place to resolve C and C++ #include directives during compilation.

   Such data sets are installed with Language Environment. When it is allocated, searching for these include files can be specified on the -I option as //DD:SYSLIB. (See the description of environment variable _CSYSLIB} for information.

   When include files are MVS PDS members, z/OS XL C/C++ uses conversion rules to transform the include (header) file name on a #include preprocessor directive into a member name. If the "//dataset_prefix.+" syntax is not used for the MVS data set which is being searched for the include file, then this transformation strips any directory name on the #include directive, and then takes the first 8 or fewer characters up to the first dot (.).

   If the "//dataset_prefix.+" syntax is used for the MVS data set which is being searched for the include file, then this transformation uses any directory name on the #include directive, and the characters following the first dot (.), and substitutes the "+" of the data set being searched with these qualifiers.

   In both cases the data set name and member name are converted to uppercase and underscores (_) are changed to at signs (@).

   If the include (header) files provided by Language Environment are installed into the hierarchical file system in the default location (in accordance with the {_INCDIRS} environment variable), then the compiler will use those files to resolve #include directives during compilation. `c89/cc/c++` by default searches the directory /usr/include as the usual place, just before searching the data sets just described. See the description of environment variables {CSYSLIB}, {_INCDIRS}, and {_INCLIBS} for information about customizing the default directories to search.

2. Feature test macros control which symbols are made visible in a source file (typically a header file). `c89/cc/c++` automatically defines the following feature test macros along with the errno macro, according to whether or not `cc` was invoked.

   - Other than `cc`
     - `-D "errno=( *__errno())"`
     - `-D _OPEN_DEFAULT=1`
   - `cc`
     - `-D "errno=( *__errno())"`
c89/cc/c++ add these macro definitions only after processing the command string. Therefore, you can override these macros by specifying -D or -U options for them on the command string.

3. For options that have option-arguments, the meaning of multiple specifications of the options is as follows:

- **-D**
  - All specifications are used. If the same name is specified on more than one -D option, only the first definition is used.

- **-I**
  - All specifications are used. If the same directory is specified on more than one -I option, the directory is searched only the first time.

- **-U**
  - All specifications are used. The name is not defined, regardless of the position of this option relative to any -D option specifying the same name.

- **-W**
  - All specifications are used. All options specified for a phase are passed to it, as if they were concatenated together in the order specified.

### Exit values

- **0**
  - Successful completion

- **4**
  - Warning error detected

### Related information

c89, cc, c++, make

---

**man - Display sections of the online reference manual**

**Format**

```
man [-wx] [-M path] [section] entry ...  
man -k [-M path] keyword ...
```

**Description**

**man** displays help information about both shell commands and the z/OS UNIX set of TSO/E commands. You can use it to search help files having the specified keywords associated with them.

**Options**

- **-k**
  - Searches a precomputed database of syntax lines for information about keywords.

- **-M path**
  - Searches the directories indicated by path for help files. If -M is not specified, man uses the path specified in the MANPATH environment variable if set; otherwise man searches /usr/man/%L. The value of the LANG environment variable is substituted for %L in this directory and in the directories specified by MANPATH. All help files are found by searching similarly structured file trees rooted at one or more places. See “Files” on page 455 for a description of what files and directories man should find in each directory that it searches.
Displays only the filename of the file containing the help file.

Displays what files man is searching to find the help file. -x also displays any errors that man encounters while extracting man pages from online book files. See “BookServer exit values” on page 456.

**section**

Is a number (0-9) representing a section of the online help. When you specify a section number, man searches only that section for entry, instead of searching all sections. The online help available for z/OS UNIX contains one section:

1 Commands

To find a given entry, man checks each directory in MANPATH for a file with a specific name. For each section number requested, man searches MANPATH for the following files in this order:

1. catn/entry.n in each directory in MANPATH.
2. mann/*.stfp in each directory in MANPATH.
3. mann/*.book in each directory in MANPATH.
4. mann/entry.n in each directory in MANPATH.

If no section number is specified then man searches all sections in order from 1 to 9, then 0. The first entry found by man is the one displayed.

If output is to the terminal, then man invokes a pager command to filter and display the manual pages. If MANPAGER is defined, it is used. If not, then if PAGER is defined, it is used; otherwise, man defaults to using the command:

```bash
pg -e -p '(Page %d)'
```

If you are running in a double-byte locale, set MANPAGER or PAGER to invoke a command which supports double-byte characters, such as the more command. pg does not support double-byte characters.

**Examples**

To find out which utilities do comparisons, issue:

```bash
man -k compare
```

You can use the man command to view manual descriptions of the z/OS UNIX set of TSO/E commands. To do this, you must prefix all commands with tso. For example, to view a description of the z/OS UNIX variation of the MOUNT command, you would enter:

```bash
man tsomount
```

You can also use the man command to view manual descriptions of commands that support subcommands. To do this, you must prefix all subcommands with the name of the command. For example, to view a description of the dbx alias subcommand, you would enter:

```bash
man dbxalias
```

The same applies for the pdbx subcommands. For example, enter:

```bash
man pdbxcont
```

to display information about the cont subcommand.

To view an online manual description for the tcsh ls-F built-in command, you must type ls-F without the dash. So, to see the man page you would issue:

```bash
man lsF
```
To view an online manual description for the tcsh @ (at) built-in command, you must type at with tcsh in front of it. So, to see the man page you would issue:

```
man tcshat
```

### Environment variables

**man** uses the following environment variables:

- **MANPATH**
  
  Contains a list of paths to search for man pages.

- **MANPAGER, PAGER**
  
  Contains an output filtering command for use when displaying man pages on a terminal.

  If you are running in a double-byte locale, verify that this variable is set to a command which supports double-byte characters, such as the **more** command.

- **TMPDIR**
  
  Identifies the directory where temporary files reside.

### Localization

**man** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


### Files

**man** uses the following files:

- `/usr/man/%L/man[0-9]/*.stfpt`
  
  Single-tagged preformatted plain text file that contains man pages.

- `/usr/man/%L/man[0-9]/*.book`
  
  BookManager® book files containing man pages.

- `/usr/man/%L/cat[0-9]/ * .[0-9]`
  
  Subdirectories containing formatted help files.

- `/usr/man/%L/whatis`
  
  Database used by -k option.

### Exit values

- **0**
  
  Topic not found.

- **1**
  
  Topic contents exceeded the buffer length.

- **-1**
  
  No errors.
BookServer exit values
If the -x option was specified and man encounters errors while extracting man pages from online book files, these exit values are displayed.

112105
The bookread component of BookServer could not locate or read the partitioned dataset of the code page translation tables. Check the bookread configuration as described in the z/OS program directory and the BookServer program directory.

100000-199999
An error occurred while opening the bookread session.

200000-299999
An error occurred while opening the book.

300000-399999
An error occurred while translating the topicid to unicode.

400000-499999
An error occurred while positioning to the topic within the book file.

500000-599999
An error occurred while reading a line of the topic.

600000-699999
An error occurred while translating the text of the book from the internal book code page to the display code page errors.

700000-799999
An error occurred while closing the book file.

800000-899999
An error occurred while closing the session.

900000-999999
An error occurred while positioning to the top of the book file.

1000000-1099999
An error occurred while looking up the CONTENTS topic ID.

Portability
POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
The elements of the environment variable MANPATH are separated by colons.
The -M option, the -x option, the -w option, the MANPAGER environment variable, the default pager, and the ability to specify section on the command line are all extensions to the POSIX standard.

Related information
help, more

mesg - Allow or refuse messages

Format
mesg [y] [n]

Description
mesg determines whether other users can send messages to your terminal with talk, write, or similar utilities.
Options

y
  Specifies that other people can send you messages.

n
  Specifies that other people cannot send you messages.

Examples

1. To let other people send you messages, issue:

   `mesg y`

2. To tell the system not to let other people send you messages, issue:

   `mesg n`

3. To display the current setting without changing it, issue:

   `mesg`

The terminal is determined by the first of standard input, output, or error which is directed to a terminal.

Localization

`mesg` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

See Appendix F, "Localization," on page 965 for more information.

Exit values

0
  Receiving messages is currently allowed

1
  Receiving messages is not currently allowed

2
  Failure due to any of the following:
  - Unknown command-line option
  - Unknown argument
  - An error accessing the terminal

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.

Related information

talk, write
mkcatdefs - Preprocess a message source file

**Format**

```
mkcatdefs [-h] MsgFile [SourceFile ...]
```

**Description**

`mkcatdefs` preprocesses a message source file for input to the `gencat` utility. `SourceFile` is a message file (usually with a `.msf` extension) containing symbolic identifiers. If you do not specify a `SourceFile`, `mkcatdefs` reads from the standard input (stdin). `mkcatdefs` produces two outputs:

- **MsgFile.h**, which contains statements that equate your symbolic identifiers with set numbers and message numbers that `mkcatdefs` assigns. You must include this header file in your application in order to refer to the messages.
- **Message source data, with numbers instead of symbolic identifiers**, is sent to standard output. This output is suitable as input to the `gencat` utility. You should either save standard output to a file using redirection, or pipe the output of `mkcatdefs` to the `gencat` utility.

**Options**

- **-h**
  
  Suppresses the generation of a `MsgFile.h` file. This flag must be the first argument to `mkcatdefs`.

**Extended description**

The format of `SourceFile` is defined as follows: The fields of a message source line must begin in column 1 and are separated by a single blank character. Any other blank characters are considered as part of the subsequent field.

- **$quote**
  
  See `gencat`.

- **$set symbolic_name**
  
  The `symbolic_name` denotes the set identifier that will be used in an application program to reference this set of messages. This name can be up to 255 characters long and can contain any alphanumeric character and the underscore character, but must begin with a non-numeric character. Any string following the set identifier is treated as a comment.

- **$ comment**
  
  See `gencat`.

- **Symbolic_Name message_text**
  
  The `Symbolic_Name` denotes a message identifier that will be used in an application program to reference this message. This name can be up to 255 characters long and can contain any alphanumeric character and the underscore character, but must begin with a non-numeric character. There must be a single blank character separating the `symbolic_name` from the `message_text`. If no quote character is defined, then any blank characters after the separating blank character are considered part of the message text. See `gencat` for more information about how to specify `message_text`.

**Examples**

To process the `comp1.msf` and `comp2.msf` message source files and put the output into the `comp.msg` file, enter:

```
mkcatdefs comp comp1.msf comp2.msf >comp.msg
```
The source message file looks similar to the following:

```
$ This is the message source file for COMP1
$quote " Use double quotation marks to delimit message text
$set MSFAC1 Message set for component comp1
$ SYM_FORM "Symbolic identifiers can only contain alphanumeric \ncharacters or the _ (underscore character)"
SYM_LEN "Symbolic identifiers cannot be more than 65 characters long"
5 "You can mix symbolic identifiers and numbers"
```

The generated comp.h file looks similar to the following:

```
#ifdef _H_COMP_MSG
#include <limits.h>
#include <nl_types.h>
/*
Time stamp: 1994 137 19:09 UTC
*/
/* The following was generated from comp1.msf. */
/* definitions for set MSFAC1 */
/* The following was generated from comp2.msf. */
/* definitions for set MSFAC2 */
#endif
```

mkcatdef creates the comp.h message catalog source file for gencat with numbers assigned to the symbolic identifiers:

```
$timestamp 1994 137 19:09 UTC
$quote " Use double quotation marks to delimit message text
$delset 1
$set 1
1 "Symbolic identifiers can only contain alphanumeric \ncharacters or the _ (underscore character)"
2 "Symbolic identifiers cannot be more than 65 characters long"
5 "You can mix symbolic identifiers and numbers"
```

The assigned message numbers are noncontiguous because the source file contained a specific number. mkcatdefs always assigns the previous number plus 1 to a symbolic identifier.

mkcatdefs inserts a $delset command before a $set command in the output message source file. This means you cannot add, delete, or replace single messages in an existing catalog when piping to the gencat utility. You must enter all messages in the set.

---

**mkdir - Make a directory**

**Format**

`mkdir [-p] [-m mode] directory ...`

**Description**

The `mkdir` command creates a new directory for each named `directory` argument. The mode for a directory created by `mkdir` is determined by taking the initial mode setting of 777 (a=rwx) or the value of `-m` if specified and applying the umask value to it.

**Options**

- **-m mode**
  
  Lets you specify permissions for the directories. The `mode` argument can have the same value as the `mode` for chmod; see chmod for more details.

  You can also set the sticky bit on for directories. For more information, see chmod. The umask value is applied to the `mode` value to determine the new directory's actual mode setting.
Note: A superuser or the file owner can use a chmod command or chmod() function to change two options for an executable file. The options are set in two file mode bits:

- Set-user-ID (S_ISUID) with the setuid option
- Set-group-ID (S_ISGID) with the setgid option

If one or both of these bits are on, the effective UID, effective GID, or both, plus the saved UID, saved GID, or both, for the process running the program are changed to the owning UID, GID, or both, for the file. This change temporarily gives the process running the program access to data the file owner or group can access.

In a new file, both bits are set off. Also, if the owning UID or GID of a file is changed or if the file is written in, the bits are turned off. In shell scripts, these bits are ignored.

If the RACF profile named FILE.GROUPOWNER.SETGID exists in the UNIXPRIV class, then the set-group-ID bit for a directory determines how the group owner is initialized for new objects created within the directory:

- If the set-gid bit is on, then the owning GID is set to that of the directory.
- If the set-gid bit is off, then the owning GID is set to the effective GID of the process.

-p
Creates intermediate directory components that don't already exist. For example, if one of the directory arguments is dir/subdir/subsub and subdir doesn't already exist, mkdir creates it. Such intermediate directories are created with mode bits determined in the following way: Take a default mode setting of 777 (a=rwx), apply the process's setting to it, and then turn on the user write and user execute permissions (u+w). The --mode specification on the command line is not used for computing the mode of intermediate directories.

Localization

mkdir uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values

0
Successful completion

1
Failure due to any of the following:

- Missing mode after -m
- Incorrect mode
- Incorrect command-line option
- Missing directory name
- Inability to create the directory

Messages

Possible error messages include:

Path not found
The preceding structure (parent directory) of the named directory does not exist.
Access denied
   The requested directory already exists or is otherwise inaccessible.

Cannot create directory
   Some other error occurred during creation of the directory.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Related information
rm, rmdir,

**mkfifo - Make a FIFO special file**

Format
mkfifo [-m mode] file [-p]

Description
mkfifo creates one or more FIFO special files with the given names.

Options
-m mode
   Lets you specify file permissions for the files. The mode argument can have the same value as the
   mode argument for chmod; see chmod for more details.

-p
   Creates intermediate directory components that do not already exist. For example, if one of the file
   arguments is dir/subdir/file and if subdir does not exist already, this option creates it. Such
   intermediate directories are created with mode bits determined in the following way: Take a default
   mode setting 777 (a=rwx), apply the umask setting of the process to it, and then turn on user read,
   write, and user execute permissions (u+rwx).

   The -m mode specification on the command line is not used for computing the mode of intermediate
   directories. The resulting mode settings permit the file owner to access the new files without concern
   for any umask setting that may be in place.

Localization
mkfifo uses the following localization environment variables:
   • LANG
   • LC_ALL
   • LC_CTYPE
   • LC_MESSAGES
   • NLSPATH


Exit values
0
   Successful completion.
1
   Failure due to any of the following reasons:
mknod

- A missing mode after -m.
- An incorrect mode.
- An incorrect command-line option.
- A missing file name.
- Inability to create the file.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.
The -p option is an extension of the POSIX standard.

Related information

chmod, create, mkdir

mknod - Make a FIFO or character special file

Format

mknod pathname [b c] major minor
mknod pathname p

Description

mknod creates a FIFO special file or a character special file with the given path name. It is located in the directory /usr/sbin.

Operands

b
Indicates block special files. b is accepted for compatibility with other UNIX implementations. Block special files are not supported on z/OS.

c
Indicates character special files (for example, terminals and other devices). c can only be used by a superuser.

major minor
major gives the major device type; minor, the minor device type. You can specify device types in decimal, hexadecimal, or octal.
mknod differentiates between octal and decimal as follows:
- Any number that starts with 0 but not 0x is octal.
- Any number that starts with 0x is hexadecimal.
- Any number that does not start with 0x or 0 is decimal.

p
Creates a FIFO special file (that is, a named pipe).

Localization

mknod uses the following localization environment variables:
- LANG
- LC_ALL
- LC_CTYPE
Exit values

0
  Successful completion

1
  Failure due to any of the following:
  • Inability to create the desired file
  • Incorrect major or minor number

2
  Failure due to any of the following:
  • Too few command-line arguments
  • A missing major or minor device number

Portability

UNIX systems. Within POSIX, mknod has been superseded by mkfifo for pipes. The POSIX family of standards has not yet designed an alternative to mknod for special files.

Related information

mkfifo

more - Display files on a page-by-page basis

Format

more [-BceiSsU] [-A | -u] [-n number] [-P prompt] [-p command] [-t tag] -W option[option] ... [file ... file ...]
more [-BceiSsU] [-A | -u] [-n number] [-P prompt] [-ttag] -W option[option] ... [+command] [file ... file ...]

Description

more displays files one page at a time. It obtains the number of lines per page from the environment or from the -n option. If standard output (stdout) is not a terminal device, the number of lines per page is infinite. If stdout is not a terminal device, all input files are copied to stdout in their entirety, without modification.

more displays the files that are specified by file ... file ... (that is, a list of file names) one at a time. When more finishes displaying one file, it begins displaying the next one in the list. If you give – as one of the file names, more reads the standard input at that point in the sequence.

more allows paging forwards and backwards (if possible) and searching for strings.

Options

-A
  Displays all characters, including unprintable ones. Typically, unprintable characters are displayed in a format that is printable, such as octal. However, with -A, the actual glyph (graphical character) is displayed. Also, by using this option, ANSI escape sequences for display modes are processed. This option cannot be used with -u.

Note: The character in the upper left corner of the screen is always displayed in normal mode.
-B
Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-w option) are specified.

-c
Displays one page at a time starting at the top of the screen, and clears the screen before displaying a new file. more may ignore this option if the terminal doesn’t support such operations.

-e
Exits immediately after displaying the last line of the last file. Typically, if stdout is a terminal device, more stops after displaying the last line of the last file and prompts for a new command. If the command that displays text causes more to reach the end of the file again, more exits.

-i
Ignores case during searches.

-n number
Specifies the number of lines per page. This overrides any values that are obtained from the environment. Use this option when you need to override the curses screen length or LINES setting to work with your terminal. This option will give incorrect results if used while in the OMVS shell (or another dumb terminal) and specifying number to be something other than the current number of screen lines.

-P string
Sets the prompt that appears at end of each page of text to string. The default prompt is [filename]. more typically displays the prompt in standout mode.

-p command
Initially executes the more command on each file. If it executes successfully and command is a positioning command such as a line number or a regular expression search, more displays the resulting page; otherwise more displays the first page of the file. If both the -t and -p options are specified, the -t option is processed first.

+command
Initially executes the more command on each file. If it executes successfully and command is a positioning command such as a line number or a regular expression search, more displays the resulting page; otherwise more displays the first page of the file. If both the -t and -P options are specified, the -t option is processed first.

-S
Displays the prompt in normal mode rather than standout (reverse video) mode.

-s
Replaces consecutive empty lines with a single empty line.

-t tag
Searches for the named tag and displays the page of text containing it. See ctags for more information.

-U
Allows more to refresh the display screen for each new line.

-u
Displays all backspaces as ^H.
Typically, character-backspace_(underscore) displays character as underlined and character-backspace-character displays character as boldfaced. -u also displays all carriage returns as ^M. This option cannot be used with -A.

-W option[.option]...
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

  filecodeset=csodeset
  Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). The command iconv -1 lists existing CCSIDs
along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

**pgmcodeset=**
codeset

Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command icnv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

**Interactive commands**

more also supports the following interactive commands.

<table>
<thead>
<tr>
<th>Interactive command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[n]b</td>
<td>[n]Ctrl-B</td>
</tr>
<tr>
<td>[n]d</td>
<td>[n]Ctrl-D</td>
</tr>
<tr>
<td>[n]f</td>
<td>[n]Ctrl-F</td>
</tr>
<tr>
<td>[n]G</td>
<td>Goes to the nth line in the file. If you do not specify n, more advances to the end of the file.</td>
</tr>
<tr>
<td>[n]g</td>
<td>Goes to the nth line in the file, with the default being the first line of the file.</td>
</tr>
<tr>
<td>h</td>
<td>Displays a summary of interactive commands.</td>
</tr>
<tr>
<td>[n]j</td>
<td>[n]SPACE</td>
</tr>
<tr>
<td>Interactive command</td>
<td>Action</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
</tr>
<tr>
<td>![n]k ![n]↑ (up arrow)</td>
<td>Scrolls backward n lines, with a default of one line. This command displays the entire n lines even if n is more than the page size.</td>
</tr>
<tr>
<td>mletter</td>
<td>Marks the current position with the lowercase letter. When you view a new file, all previous marks are lost.</td>
</tr>
<tr>
<td>![n]N</td>
<td>Repeats the previous search, but in the opposite direction. If you specify n, more repeats the search n times.</td>
</tr>
<tr>
<td>![n]n</td>
<td>Repeats the previous search. If you specify n, more repeats the search n times. For example if there are eight occurrences of pattern in the file and /pattern found the second occurrence then a follow-up command of 5n finds and sets the current position to the 7th occurrence of pattern.</td>
</tr>
<tr>
<td>q</td>
<td>Exits more.</td>
</tr>
<tr>
<td>:q ZZ</td>
<td>Exits more.</td>
</tr>
<tr>
<td>R</td>
<td>Refreshes the screen and discards any buffered input.</td>
</tr>
<tr>
<td>Ctrl-L</td>
<td>Refreshes the screen.</td>
</tr>
<tr>
<td>![n]s</td>
<td>Skips forward n lines (with a default of one line) and displays one page beginning at that point. If ![n] would cause less than one page to be displayed, more displays the last page in the file.</td>
</tr>
<tr>
<td>![n]u ![n]Ctrl-U</td>
<td>Scrolls backward n lines, with a default of one half of the page size. If you specify n, it becomes the new default for subsequent d and u commands.</td>
</tr>
<tr>
<td>v</td>
<td>Invokes an editor to edit the current file. more uses the editor named by the environment variable EDITOR. The default editor is vi. If the editor is ex or vi, the text conversion that is specified on the more command (for example, the -B or -W option) is used.</td>
</tr>
<tr>
<td>'letter</td>
<td>Returns to the position marked with letter.</td>
</tr>
<tr>
<td>'</td>
<td>Returns to the position from which you last issued a movement command of greater than one page or the beginning of the file if you have issued no such commands.</td>
</tr>
<tr>
<td>![n]/[!]pattern</td>
<td>Searches forward in the file for the nth line containing pattern. n defaults to one if not specified. If pattern is the null regular expression (/) more uses the previous pattern. If the character ! precedes pattern, more searches for lines that do not contain pattern.</td>
</tr>
<tr>
<td>![n]/[!]pattern</td>
<td>Searches backward in the file for the nth line containing pattern. The search begins at the line immediately before the top line displayed. n defaults to one if not specified. If pattern is the null regular expression (?), more uses the previous pattern. If the character ! precedes pattern, more searches for lines that do not contain pattern.</td>
</tr>
<tr>
<td>:e [filename]</td>
<td>Stops viewing the current file and views filename instead. If you do not specify filename, more returns to the beginning of the current file. If filename is #, more returns to the last file viewed before the current one. The text conversion that is specified on the more command (for example, the -B or -W option) is used.</td>
</tr>
<tr>
<td>![n]:n</td>
<td>Views the next file from the list given on the command line. If you specify n, more views the nth next file from the list.</td>
</tr>
<tr>
<td>![n]:p</td>
<td>Views the previous file from the list given in the command line. If you specify n, more views the nth previous file from the list.</td>
</tr>
<tr>
<td>:t tagname</td>
<td>Goes to tagname.</td>
</tr>
<tr>
<td>:w filename</td>
<td>Writes the contents of the current file to the file filename. The text conversion that is specified on the more command (for example, the -B or -W option) is used.</td>
</tr>
<tr>
<td>!&lt;shell command&gt;</td>
<td>Escape to shell and execute shell command.</td>
</tr>
<tr>
<td>=</td>
<td>Displays, where possible, the name of the file currently being viewed, its number (relative to the total number of files specified in the command line), the current line number, the current byte number, the total bytes to display and what percentage of the file has been displayed.</td>
</tr>
</tbody>
</table>
Interactive command | Action
---|---
Ctrl-G | Displays, where possible, the name of the file currently being viewed, its number (relative to the total number of files specified in the command line), the current line number, the current byte number, the total bytes to display and what percentage of the file has been displayed.
Home | Goes to the first line in the file.
End | Goes to the last line in the file.

Examples
1. To display a text file one page at a time starting at line 12:
   ```
   more +12g myTextFile
   ```
2. To display a text file containing UTF-8 characters one page at a time, assuming that:
   • The text file is untagged and you do not want to tag it or enable automatic conversion, and
   • You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file):
   ```
   more -W filecodeset=1208,pgmcodeset=IBM-1047 myutf8File
   ```
3. To display a text file containing EBCDIC characters one page at a time, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as ASCII:
   ```
   more -B myMisTaggedFile
   ```

Environment variables
`more` uses the following environment variables:

>_TEXT_CONV_
Contains text conversion information for the command. The text conversion information is not used when either the `-B` option or the `filecodeset` or `pgmcodeset` option (`-W` option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

COLUMN
Contains the maximum number of columns to display on one line.

EDITOR
Contains the name of the editor that the `v` command invokes.

LINES
Contains the number of lines in a page. This value takes precedence over value from TERM. However, the `-n` value takes precedence over the LINES value.

MORE
Contains a list of options as they would appear on the command line. This variable takes preference over the TERM and LINES variables.

TERM
Contains the name of the terminal type.

Usage notes
`more` is designed for raw-mode terminals. It can be used with 3270 terminals with certain restrictions. Line-mode terminals require a user to press Enter to allow the keys typed to be processed. However, the Enter key has a special meaning to `more`. Specifically, it causes `more` to scroll down a single line. Therefore, when attempting to use `more` while in line-mode, each time a user presses "Enter" to process any command, this causes the screen to scroll down a single line at a time.
Localization

more uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH


Exit values

0
   Successful completion

>0
   Failure due to any of the following:
   • filename is not a text file
   • -n option too large
   • Syntax error in regular expression
   • Inability to create a file
   • Inability to open input file
   • Insufficient memory
   • Incorrect command
   • Inability to access the terminal
   • Missing string after -p option
   • The code set is not valid
   • Could not turn off automatic conversion
   • Could not perform requested text conversion

Portability

POSIX.2 User Portability Extension, UNIX systems.

The -A, -B, -P, -s, -U, and -W options and the :w and ! commands are extensions of the POSIX standard. The Home, End, PgDn, PgUp, ↓ (down arrow), and ↑ (up arrow) commands are extensions to traditional implementations of more, available only on terminal types that support these keys.

Related information

cat, vi

mount - Logically mount a file system

Format

mount [-t fstype][-rv][-a yes|include,sysname1,... sysnameN |exclude,|no|unmount][-o fsoptions][-d destsys][-s nosecurity|nosetuid][-f fsname pathname][-wn]

mount -q [-d destsys][-v] pathname

File tag specific option:

mount [-c ccsid,text|notext]
**Description**

The `mount` shell command in `/usr/sbin` is used to mount a file system or list all mounts over a file system.

You must have mount authority before you can issue the `mount` command. See the section on mounting file systems in *z/OS UNIX System Services Planning*.

**Options**

- **-a yes|include,sysname1,...,sysnameN|exclude,sysname1,...,sysnameN|no|unmount**
  
  The `-a` option specifies the AUTOMOVE attribute of the file system in a sysplex environment where systems are exploiting the shared file system capability.
  
  - `-a yes` allows the system to automatically move logical ownership for a specified file system as needed. This is the default.
  - `-a no` prevents ownership movement in some situations.
  - `-a unmount` unmounts the file system in some situations.
  - `-a include,sysname1,...,sysnameN` specifies a list of systems, in priority order, to which the file system's ownership can be moved. `include` can be abbreviated to `i`.
  - `-a exclude,sysname1,...,sysnameN` specifies a list of systems, in priority order, to which the file system's ownership cannot be moved. `exclude` can be abbreviated to `e`.

  See *z/OS UNIX System Services Planning* for details about the behavior of the AUTOMOVE options.

- **-d destsys**
  
  Specifies the name of the system in a shared file system environment that will be the logical owner of the mount. Note, if `-q` is specified, the `mount -q` output will only list mounts that are owned by `destsys`.

- **-f fsname**
  
  Names the file system to be mounted. All file system names must be unique. File system names are case-sensitive. However, if the file system type is HFS, or if the type was not specified on the command and the file system is zFS/fsname is converted to uppercase. The file system name has a maximum length of 44 characters; any additional characters are truncated. Options `-q` and `-f` are mutually exclusive, but one must be specified.

- **-w n**
  
  Specifies the amount of time the mount will wait in seconds for async mounts to complete. If `n` is specified as a 0 the wait will be indefinite. This option flag is tolerated on any form of the mount command and is ignored if not appropriate (no wait needs to be done).

- **-o fsoptions**
  
  Specifies an option string to be passed to the file system type. NFS, for example, uses this to identify the remote server and the object on that server. The format and content are specified by the physical file system that is to perform the logical mount. You can specify lowercase or uppercase characters. Enclose the string in single quotation marks.

  Refer to the following for the appropriate file system-specific options to specify for `fsoptions`:

  - For HFS-specific options, see the BPXPRMxx section in *z/OS MVS Initialization and Tuning Reference*.
  - For zFS-specific options, see Mount in *z/OS Distributed File Service zFS Administration*.
  - For NFS-specific options, see Mount processing parameters, in *z/OS Network File System Guide and Reference*.
  - For TFS-specific options, see Mounting the TFS, in *z/OS UNIX System Services Planning*.

- **-q**
  
  Prints a list of path names for the mount points of file systems that are mounted over another file system, including that system. Options `-q` and `-f` are mutually exclusive, but one must be specified. If `-v` is not specified, only path names for mount points are printed. Note that the output of `mount -q` can be used by the `unmount` utility as input. See “Examples” on page 471.
If -q and -v are specified then the output consists of a 6-character mode, followed by the file system name, followed by the file system mount point path name. The 6-character mode can be interpreted as shown in Table 18 on page 470.

<table>
<thead>
<tr>
<th>Column</th>
<th>Flags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
<td>Read/write</td>
</tr>
<tr>
<td></td>
<td>- R</td>
<td>Read-only</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>SETUID supported</td>
</tr>
<tr>
<td></td>
<td>- S</td>
<td>SETUID not supported</td>
</tr>
<tr>
<td>3</td>
<td>E</td>
<td>File system not exported</td>
</tr>
<tr>
<td></td>
<td>- E</td>
<td>File system exported by DFS</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>Security checks enforced</td>
</tr>
<tr>
<td></td>
<td>- U</td>
<td>No security checks enforced</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>Noautomove</td>
</tr>
<tr>
<td></td>
<td>- A</td>
<td>Automove</td>
</tr>
<tr>
<td></td>
<td>- U</td>
<td>Unmount</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>Owning system or sysplex-aware</td>
</tr>
<tr>
<td></td>
<td>- C</td>
<td>Client</td>
</tr>
</tbody>
</table>

**-r**
Specifies mounting a file system read-only.

**-s nosecurity|nosetuid**
Specifies that a file system is unsecured. Setuid, setgid, APF and program-controlled attributes are ignored when you use nosetuid. To additionally disable authorization checking, use nosecurity. Minimum unique abbreviations can be used for the option arguments.

**Note:** When a file system is mounted with the NOSECURITY option enabled, any new files or directories that are created are assigned an owner of UID 0, no matter what UID issued the request.

**-t fstype**
Identifies the file system type. fstype can be entered in mixed case but will be treated as uppercase. If this option is not specified, the default is -t HFS.

**-v**
Verbose output. Includes additional information, if available, on output. If -v is specified on the mount command and the mount fails, the file system name that had the mount failure will be included in the failure information.

**pathname** specifies the path name for the mount point.

**File tag-specific option**

**-c ccсид|text|notext**
Specifies the file tag that will be implicitly set for untagged files in the mounted file system.

**ccsid**
Identifies the coded character set identifier to be implicitly set for the untagged file. ccсид is specified as a decimal value from 0 to 65535. However, when text is specified, the value must be
between 0 and 65535. Other than this, the value is not checked as being valid and the corresponding code page is not checked as being installed.

For more information about file tagging, see z/OS UNIX System Services Planning.

text
Specifies that each untagged file is implicitly marked as containing pure text data that can be converted.

notext
Specifies that none of the untagged files in the file system are automatically converted during file reading and writing.

Examples
1. The output of `mount -q` can be used for the input of `unmount`. For example:
   
   ```
   mount -q /ict/hfsfir
   ```

   can be used as input:

   ```
   unmount $(mount -q /ict/hfsdir)
   ```

2. To mount an HFS file system over `/u/wjs` with a sync interval of 120 seconds:
   
   ```
   mount -f omvs.hfs.user.wjs -o 'SYNC(120)' /u/wjs
   ```

3. To display a list of path names for all mount points under `/u`:
   
   ```
   mount -q /u
   ```

Usage notes
1. Systems exploiting shared file system will have I/O to an OMVS couple data set. Because of these I/O operations to the CDS, each mount request requires additional system overhead. You will need to consider the effect that this will have on your recovery time if a large number of mounts are required on any system participating in shared file system.

2. The `-a` `unmount` is not available to automounted file systems.

3. The file system name is treated as uppercase when the file system type is not specified (`-t` option).

File system recovery and mount

File system recovery in a shared file system environment takes into consideration file system specifications such as `-a yes|no|unmount` and whether the file system is mounted read-only or read/write.

Generally, when an owning system fails, ownership over its `-a yes` mounted file system is moved to another system and the file is usable. However, if a file system is mounted read/write and the owning system fails, then all file system operations for files in that file system will fail. This is because data integrity is lost when the file system owner fails. All files should be closed (BPX1CLO) and reopened (BPX1OPN) when the file system is recovered. (The BPX1CLO and BPX1OPN callable services are discussed in z/OS UNIX System Services Programming: Assembler Callable Services Reference.)

For file systems that are mounted read-only, specific I/O operations that were in progress at the time the file system owner failed might need to be submitted again. Otherwise, the file system is usable.

In some situations, even though a file system is mounted with the `-a yes` option, ownership of the file system might not be immediately moved to another system. This may occur, for example, when a physical
I/O path from another system to the volume where the file system resides is not available. As a result, the file system becomes "unowned" (the system will issue message BPXF213E when this occurs). This is true if the file system is mounted either read/write or read-only. The file system still exists in the file system hierarchy so that any dependent file systems that are owned by another system are still usable.

However, all file operations for the unowned file system will fail until a new owner is established. The shared file system support will continue to attempt recovery of -a yes mounted file systems on all systems in the sysplex that are enabled for shared file system. Should a subsequent recovery attempt succeed, the file system transitions from the unowned to the active state.

Applications that use files in unowned file systems must close (BPX1CLO) those files and reopen (BPX1OPN) them after the file system is recovered.

File systems that are mounted with the -a no option will become unowned when the file system owner exits the sysplex. The file system will remain unowned until the original owning system restarts or until the unowned file system is unmounted. Because the file system still exists in the file system hierarchy, the file system mount point is still in use.

An unowned file system is a mounted file system that does not have an owner. The file system still exists in the file system hierarchy. As such, you can recover or unmount an unowned file system.

File systems that are associated with a "never move" PFS will be unmounted during dead system recovery. For example, TFS is a "never move" PFS and will be unmounted, as well as any file systems mounted on it, when the owning system leaves the sysplex.

As stated in “Usage notes” on page 471, -a unmount is not available to automounted file systems. However, during dead system recovery processing for an automounted file system (whose owner is the dead system), the file system is unmounted if it is not being referenced by any other system in the sysplex.

**Exit values**

0

Successful completion.

**Related information**

chmount, unmount

### mv - Rename or move a file or directory

**Format**

```
mv [-fiMUv][-F format|B|T|X][-P params][-W seqparms=params][-Z][[-0 u | c=]codeset] file1 file2
mv [-ACfiMUv][-F format|B|T|X][-S suffix][-Z][-0 u | c=]codeset] file ... file ... directory
mv -R[[-fi][-Z][-0 u | c=]codeset] directory1 directory2
```

**Description**

mv renames files or moves them to a different directory. If you specify multiple files, the target (that is, the last path name on the command line) must be a directory. mv moves the files into that directory and gives them names that match the final components of the source path names. When you specify a single source file and the target is not a directory, mv moves the source to the new name, by a simple rename if possible.

You can also use mv to move files to and from MVS data sets. If you specify more than one file to be moved, the target (last path name on command line) must be either a directory or a partitioned data set. If the target is an MVS partitioned data set, the source cannot be a UNIX directory.
mv does not support the moving to or from generation data groups (GDGs). To use those MVS data sets, user must specify the real data set name. mv also does not support copying to a temporary PDSE.

When moving records, the string " \n" is moved the same way as the string "\n": both are read back as "\n", where "\n" indicates that z/OS XL C++ will write a record containing a single blank to the file (the default behavior of z/OS XL C/C++). All other blanks in your output are read back as blanks, and any empty (zero-length) records are ignored on input. However, if the environment variable _EDC_ZERO_RECLEN is set to Y before calling mv, an empty record is treated as a single newline and is not ignored. Also, if _EDC_ZERO_RECLEN is set to Y, a single newline is written to the file as an empty record, and a single blank will be represented by " \n".

A file can be moved by any user who has write permission to the directory containing the file, unless that directory has its sticky bit turned on. If the file is in a directory whose sticky bit is turned on, only the file owner or a superuser can move the file.

You can move:

• One file to another file in the working directory.
• One file to a new file on another directory.
• A set of directories and files to another place in your file system.
• A UNIX file to an MVS data set.
• An MVS data set to a file system.
• An MVS data set to an MVS data set.

Options

-A
Specifies that all suffixes (from the first period until the end of the target) be truncated. -A has precedence over -M and -C options. -S will be turned off if -A is the last option specified.

-B
Specifies that the data to be moved contains binary data. When you specify -B, mv operates without any consideration for <newline> characters or special characteristics of DBCS data (this type of behavior is typical when moving across a UNIX system). -B is mutually exclusive with -F, -X, and -T. You will get an error if you specify more than one of these options.

-C
Specifies truncating the file names to 8 characters to meet the restriction in the MVS data set member.

-F format
Specifies whether the file is to be treated as binary, text, or record file format when moved; for text files, specifies the end-of-line delimeter. Also sets the file format to format only if the source is an MVS data set and the target is a UNIX file. Only cp sets the file format for UNIX to UNIX operations. For text files, when moving from UNIX to MVS, the end-of-line delimeter will be stripped. When moving from MVS to UNIX, the end-of-line delimeter will be added. (Code page IBM-1047 is used to check for end-of-line delimiters). Record file formats are treated as if they were binary files.

-F is mutually exclusive with -B, -X, and -T. If you specify one of these options with -F, you will get an error. If -F is specified more than once, the last - specified is used.

For format, you can specify:

not.
   Not specified.
bin.
   Binary data
rec.
   Record. (File data consists of records with prefixes. The record prefix contains the length of the record that follows. From the shell command perspective, files with this format are treated as if they were binary files.)
Or the following text data delimiters:

- `nl`
  Newline character.

- `cr`
  Carriage return.

- `lf`
  Line feed.

- `crlf`
  Carriage return followed by line feed.

- `lfcr`
  Line feed followed by carriage return.

- `crnl`
  Carriage return followed by a new line character.

- `-f`
  Does not ask if you want to overwrite an existing UNIX destination file; it automatically behaves as if you answered yes. If you specify both `-f` and `-i`, `/mv` uses the option that appears last on the command line.

- `-i`
  When moving to a UNIX target, always prompts before overwriting an existing file, but does not overwrite the file if you do not have permission. If you specify both `-f` and `-i`, `/mv` uses the option that appears last on the command line.

- `-M`
  Specifies that some characters of the file name are converted when moving between a UNIX file and a data set member. Characters are converted as follows:
  - `_` (underscore) in UNIX is converted to `@` in MVS data set members and vice versa.
  - `.` (period) in UNIX is converted to `#` in MVS dataset members and vice versa.
  - `-` (dash) in UNIX is converted to `$` in MVS data set members and vice versa.

- `-P params`
  Specifies the parameters that are needed to create a sequential data set if one does not already exist. You can specify the `RECFM`, `LRECL`, `BLKSIZE`, and `SPACE` in the format that the `fopen()` function uses.

  SPACE=<(units, (primary, secondary))> where the following values are supported for units:
  - Any positive integer indicating `BLKSIZE`.
  - `CYL` (mixed case).
  - `TRK` (mixed case). For example:

  ```
  SPACE=(500, (100, 500)) units, primary, secondary
  SPACE=(500, 100) units and primary only
  ```

  For information about how to specify these parameters, see `z/OS XL C/C++ Programming Guide`.

**Note:**

1. The `fopen()` argument `LRECL` specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records.
2. `BLKSIZE` specifies the maximum length, in bytes, of a physical block of records.
3. `RECFM` refers to the record format of a data set and `SPACE` indicates the space attributes for MVS data sets.
-R (UNIX to UNIX only)
Moves a directory and all its contents (files, subdirectories, files in subdirectories, and so on). For example:

```
mv -R dir1 dir2
```

moves the entire contents of dir1 to dir2/dir1. `/mv` creates any directories that it needs.

-x (UNIX to UNIX only)
Is identical to -R.

-S d=suffix|a=suffix
- d=suffix
Removes the specified suffix from a file.
- a=suffix
Appends the specified suffix to a file.

-S has precedence over -M and -C. It also turns off the -A option (if -S is the last specified option).

-T
Specifies that the data to be moved contains text data. See “Usage notes” on page 481 for details on how to treat text data. This option looks for IBM-1047 end-of-line delimiters, and is mutually exclusive with -F, -X, and -B. You will get an error if you specify more than one of these options.

-T is ignored when moving across UNIX file systems.

-U
Keeps file names in uppercase when moving from MVS data set members to UNIX files. The default is to make the file names lowercase.

-v
Verbose

-w seqparms=params
Specifies the parameters that are needed to create a sequential data set if one does not already exist. You can specify the RECFM, LRECL, BLKSIZE, and SPACE in the format that the fopen() function uses.

```
SPACE=(units,(primary,secondary)) where the following values are supported for units:
- Any positive integer indicating BLKSIZE.
- CYL (mixed case).
- TRK (mixed case). For example:

```
```
SPACE=(500,(100,500)) units, primary, secondary
SPACE=(500,100) units and primary only
```
```
For information about how to specify these parameters, see z/OS XL C/C++ Programming Guide.

Note: The fopen() arguments: LRECL specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records. BLKSIZE specifies the maximum length, in bytes, of a physical block of records. RECFM refers to the record format of a data set and SPACE indicates the space attributes for MVS data sets.

This option is the same as -P params.

-X
Specifies that the data to be moved is an executable. Cannot be used in conjunction with -F, -T, or -B.

-Z
Specifies that error messages are not to be displayed when setting ACLs or the file tag on the target. The return code will be zero. `/mv` will try to preserve the ACLs, if possible. The ACLs are not preserved if a file system does not support ACLs, or if you are moving files to MVS. For more information about file tagging, see Automatic conversion and file tagging behavior for `mv`. 
mv

-0 u | c=\texttt{codeset}  
Allow automatic conversion on source and target files.

-0 u

If the target exists and is not empty or already tagged, \texttt{mv} will not change the target's tag in order for the target to be a candidate for automatic conversion.

For new targets and existing, untagged, empty files, this option has no effect and \texttt{mv} behaves the same as the default. For a description of the default behavior, see Automatic conversion and file tagging behavior for \texttt{mv}.

When using \texttt{mv} to move from a UNIX file to an MVS data set, if the source is a tagged text file, then it might be a candidate for automatic conversion.

When using \texttt{mv} to move executables from or to MVS, automatic conversion is disabled for both source and target.

-0 \texttt{c=\texttt{codeset}}

For a detailed description of the behavior of this option on \texttt{mv}, see Automatic conversion and file tagging behavior for \texttt{mv}.

\texttt{codeset} can be a code set name that is known to the system or the numeric coded character set identifier (CCSID). If a code set name exists, the numeric CCSID associated with that name is used. Note that the command \texttt{iconv -l} lists existing CCSIDs along with their corresponding code set names.

To prevent the corruption of text files, \texttt{mv} will fail if it cannot set the tag to text or code set.

\textbf{Attention:} If automatic conversion is not set properly or the source is not tagged properly, the target might end up with a tag code set that does not match the file content.

If you do not specify \texttt{-F | B | T} or \texttt{X}, \texttt{mv} will first check the format of the MVS data set indicated and then try to determine the type of file.

\textbf{Automatic conversion and file tagging behavior for \texttt{mv}}

The following tables describe the behavior of file tagging and automatic conversion for various source and target scenarios depending on whether the \texttt{-O} option is specified on the \texttt{mv} command.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
 & Default (without \texttt{-O option}) & With \texttt{-O u option} & With \texttt{-O c=\texttt{codeset}} option \\
\hline
\textbf{File tagging} & If the target file system supports setting file tags... & If the target file system does not support setting file tags (such as NFS) ... & The target's txtflag is set to TEXT and its code set is set to \texttt{codeset}. \\
\hline
 & Target file is tagged the same as the source file. & \begin{itemize}
  \item An existing target's tag is unchanged.
  \item A new target is created with a tag according to the file system's attributes (The MOUNT parameter can specify TAG).
\end{itemize} & \begin{itemize}
  \item The target's tag is not changed.
  \item The source or target file is a candidate for automatic conversion when its txtflag is tagged TEXT.
\end{itemize} \\
\hline
\textbf{Automatic conversion} & Disabled for source and target files & Allowed for source and target files & \\
\hline
\end{tabular}
\end{table}
Table 20: Automatic conversion and file tagging behavior for mv: Moving files to MVS data sets.

<table>
<thead>
<tr>
<th></th>
<th>Default (without -O option)</th>
<th>With -O u option</th>
<th>With -O c=code option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If the SOURCE is text or binary:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File tagging</td>
<td>Not applicable for target data set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic conversion</td>
<td>Disabled for source file</td>
<td>Allowed for source file</td>
<td>Disabled for source file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The source file is a candidate for automatic conversion when its txtflag is tagged TEXT.</td>
<td></td>
</tr>
<tr>
<td><strong>If the SOURCE is executable:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File tagging</td>
<td>Not applicable for target data set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic conversion</td>
<td>Disabled for source file</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Limits and requirements**

General requirements are as follows:

1. To specify an MVS data set name, precede the name with double slashes (/ /). For example, to specify the fully qualified data set names 'turbo.gammaLib' and 'turbo.gammaLib(pgm1)', write:

   "//turbo.gammaLib"
   "//turbo.gammaLib(pgm1)"

   The same goes for data set names that are not fully qualified:

   //turbo

2. For PDS (partitioned data set) or PDSE (partitioned data set extended), to avoid parsing by the shell, the name should be quoted or minimally, the parenthesis should be escaped. For example, to specify 'turbo(pgm1)', you can use quotation marks:

   "//turbo(pgm1)"

   or escape the parenthesis:

   //turbo\(pgm1\)

   As indicated, a fully qualified name must be single-quoted (as is done within TSO). To prevent the single quotes from being interpreted by the shell, they must be escaped or the name must be placed within regular quotation marks. See the 'turbo.gammaLib' examples.

3. If you specify a UNIX file as source and the MVS data set (target) does not exist, a sequential data set will be created. If the partitioned data set exists, the UNIX file will be moved to the partitioned data set member.

4. If source is an MVS data set and target is a UNIX directory, the UNIX directory must exist.

5. You cannot have a UNIX directory, partitioned data set, or sequential data set as source if the target is a partitioned data set.

6. To move all members from a partitioned data set, you can specify the partitioned data set as source and a UNIX directory as target.

**MVS data set naming limitations:**

- Data set names can only contain uppercase alphabetic characters (A-Z). Lowercase characters will be converted to uppercase during any moves to MVS data sets.
• Data set names can contain numeric characters 0-9 and special characters @, #, and $.
• Data set names cannot begin with a numeric character.
• A data set member name cannot be more than 8 characters. If a file name is longer than 8 characters or uses characters that are not allowed in an MVS data set name, the file is not moved. You can use the -C option to truncate names to 8 characters.

Limitations: UNIX to MVS data set:
1. If you specify a sequential MVS data set that is in undefined record format, the file is moved as binary.
2. If you specify a PDSE that is in undefined record format, the first file successfully moved determines in what format files will be moved. Note that PDSE does not allow mixture. So if the first successfully moved file is an executable, the PDSE will have program objects only and all other files will fail. However, if the first file is data, then all files are moved as binary.
3. If you specify a PDS that is in undefined record format, UNIX executables are saved as PDS load modules. All other files are moved as binary.
4. If you specify an MVS data set that is either in variable length or fixed record length and you have not set the file format, text files are moved as text, binaries as binary, and executables as binary. (IBM-1047 end-of-line delimiters are detected in the data)
5. If you set the file format, the set value is used to determine if data is binary, text, or record file format.

Limitations: MVS data set to UNIX:
1. If an UNIX file does not exist, one is created using 666 mode value, whether the data to be copied is binary or text:

   666 mode value: owner(rw-) group(rw-) other(rw-)

   If the data to be copied is a shell script or executable residing in a PDS or PDSE with undefined record format, the UNIX file is created using 777 mode value:

   777 mode value: owner(rwx) group(rwx) other(rwx)

2. If a UNIX file exists and the file format is set, mv moves the file as that format. Otherwise,
   • Load modules (PDS) are stored as UNIX executables and program objects (PDSE) are moved since they are the same as executables;
   • Data within data sets of undefined record format are moved as binary if the data is not a program object or load module;
   • Data found within data sets of fixed length or variable record length are moved as text. (IBM-1047 end-of-line delimiters are detected in the data)

Limitations: MVS to MVS:
1. Options -A, -C, -f, and -S are ignored.
2. If target and source are in undefined record format (and neither is a sequential data set), mv will attempt to move the data as a load module. If that fails, then mv will move the data as binary.
3. If target and source are in undefined record format and either is a sequential data set, mv moves the data as binary.
4. If the source has a fixed or variable record length and the target is in undefined record format, mv moves the data as binary.
5. If the source is in undefined record format and the target has a fixed or variable record length, mv moves the data as binary.
6. If both source and target are in fixed or variable record length, mv moves the data as text.

Limitations: Moving executables into a PDS:
1. A PDS cannot store load modules that incorporate program management features.
2. c89, by default, produces objects using the highest level of program management.
3. If you plan on moving a load module to a PDS, you can use a prelinker, which produces output compatible with linkage editor. Linkage editor generated output can always be stored in a PDS.

Table 21 on page 479 is a quick reference for determining the correct use of options with `mv`.

<table>
<thead>
<tr>
<th>Source/target</th>
<th>Options allowed</th>
<th>Options ignored</th>
<th>Options failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX file/UNIX file</td>
<td>Ffi</td>
<td>ABCMPSTUX</td>
<td></td>
</tr>
<tr>
<td>UNIX file/Sequential data set</td>
<td>BFiPT</td>
<td>ACfMSU</td>
<td>X</td>
</tr>
<tr>
<td>UNIX file/PDS or PDSE member</td>
<td>BFiTX</td>
<td>ACfMPSU</td>
<td></td>
</tr>
<tr>
<td>Sequential data set/UNIX file</td>
<td>BFfTU</td>
<td>ACMPS</td>
<td>X</td>
</tr>
<tr>
<td>Sequential data set/ sequential data set</td>
<td>BFiPT</td>
<td>ACfMSU</td>
<td>X</td>
</tr>
<tr>
<td>Sequential data set/PDS or PDSE member</td>
<td>BFiT</td>
<td>ACfMPSU</td>
<td>X</td>
</tr>
<tr>
<td>PDS or PDSE member/UNIX file</td>
<td>BFfTX</td>
<td>ACMPS</td>
<td></td>
</tr>
<tr>
<td>PDS or PDSE member/ sequential data set</td>
<td>BFiPT</td>
<td>ACfMSU</td>
<td>X</td>
</tr>
<tr>
<td>PDS or PDSE member/PDS or PDSE member</td>
<td>BFiT</td>
<td>ACfMPSU</td>
<td></td>
</tr>
<tr>
<td>UNIX file/UNIX directory</td>
<td>Fi</td>
<td>ABCFMPSTUX</td>
<td></td>
</tr>
<tr>
<td>PDSE or PDSE member/UNIX directory</td>
<td>BFfMSTUX</td>
<td>ACP</td>
<td></td>
</tr>
<tr>
<td>UNIX file/partitioned data set</td>
<td>ABCFfMSTUX</td>
<td>fPU</td>
<td></td>
</tr>
<tr>
<td>PDS or PDSE member/ partitioned data set</td>
<td>BFfTX</td>
<td>ACfMPSU</td>
<td></td>
</tr>
<tr>
<td>UNIX directory/UNIX directory</td>
<td>fi</td>
<td>ABCFMPSTUX</td>
<td></td>
</tr>
<tr>
<td>Partitioned data set/UNIX directory</td>
<td>ABCfMSTUX</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>

The tables that follow indicate the types of moves allowed using `/mv`.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX file, sequential data set, or partitioned data set member</td>
<td>UNIX file, sequential data set, or partitioned data set member</td>
<td>Yes</td>
</tr>
<tr>
<td>UNIX directory (dir)</td>
<td>UNIX directory (dir2 exists)</td>
<td>Yes (Results will be found in dir2/ dir1/ ..).</td>
</tr>
<tr>
<td>UNIX directory (dir)</td>
<td>UNIX directory (dir2 does not exist)</td>
<td>Yes (Results will be found in dir2/...).</td>
</tr>
</tbody>
</table>
### Table 22: Moves allowed for mv: File to File (continued)

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioned data set</td>
<td>UNIX directory (dir). Results in each member of data set are moved to dir.</td>
<td>Yes</td>
</tr>
<tr>
<td>UNIX directory</td>
<td>Partitioned data set</td>
<td>No</td>
</tr>
<tr>
<td>Partitioned data set</td>
<td>Partitioned data set</td>
<td>No</td>
</tr>
<tr>
<td>UNIX file, UNIX directory, or partitioned data set member</td>
<td>UNIX directory</td>
<td>Yes</td>
</tr>
<tr>
<td>Partitioned data set member</td>
<td>Partitioned data set (must exist)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Table 23: Moves allowed for mv: File... (multiple files) to directory

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any combination of UNIX file or partitioned data set member</td>
<td>UNIX directory or partitioned data set</td>
<td>Yes</td>
</tr>
<tr>
<td>Any combination of UNIX directory, partitioned data set, sequential data set</td>
<td>Partitioned data set</td>
<td>No</td>
</tr>
<tr>
<td>Sequential data set</td>
<td>UNIX directory</td>
<td>No</td>
</tr>
<tr>
<td>Any combination of UNIX directory, UNIX file, partitioned data set, partitioned data set member</td>
<td>UNIX directory</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Examples

1. To specify `-P params` for a non-existing sequential target:

   ```
   mv -P "RECFM=U,space=(500,100)" file "//'turbo.gammalib'"
   
   This `mv` command is equivalent to:
   
   ```
   mv -W "seqparms='RECFM=U,space=(500,100)'" file "//'turbo.gammalib'"
   ```

2. To move file `f1` to a fully qualified sequential data set `turbo.gammalib` and treat it as a binary:

   ```
   mv -F bin f1 "//turbo.gammalib"
   ```

3. To move all members from a fully qualified PDS `turbo.gammalib` to an existing UNIX directory `dir`:

   ```
   mv "//turbo.gammalib" dir
   ```

4. To drop `.c` suffixes before moving all files in UNIX directory `dir` to an existing PDS `turbo.gammalib`:

   ```
   mv -S d=.c dir/* "//turbo.gammalib"
   ```
Usage notes

You cannot move a PDS or PDS/E data set member to a UNIX file with the `mv` command if the PDS or PDS/E data set has already been opened by another user or job. For example, if the data set `posix.dest.pdse` is being viewed by ISPF browse, the following `mv` command will fail:

```
mv "//'posix.test.pdse(member)'" test_file
```

**UNIX to MVS:**

1. To move from UNIX to a partitioned data set, you must allocate the data set before doing the `mv`.
2. If an MVS data set does not exist, `mv` will allocate a new sequential data set of variable record format.
3. For text files, all `<newline>` characters are stripped during the move. Each line in the file ending with a `<newline>` character is moved into a record of the MVS data set. If text file format is specified or exists for the source file, that file format will be used for end-of-line delimeter instead of `<newline>`. Note that `mv` looks for IBM-1047 end-of-line delimiters in data.

You cannot move a text file to an MVS data set that has an undefined record format:

- For an MVS data set in fixed record format, any line moved longer than the record size will cause `mv` to fail with a displayed error message and error code. If the line is shorter than the record size, the record is padded with blanks.
- For an MVS data set in variable record format: Any line moved longer than the largest record size will cause `mv` to fail with a displayed error message and error code. The record length is set to the length of the line.

4. For binary files, all moved data is preserved:

- For an MVS data set in fixed record format, data is cut into chunks of size equal to the record length. Each chunk is put into one record. The last record is padded with blanks.
- For an MVS data set in variable record format, data is cut into chunks of size equal to the largest record length. Each chunk is put into one record. The length of the last record is equal to length of the data left.
- For an MVS data set in undefined record format, data is cut into chunks of size equal to the block size. Each chunk is put into one record. The length of the last record is equal to the length of the data left.

5. For load modules, the partitioned data set specified must be in undefined record format otherwise the executable will not be moved.

6. If more than one file name is the same, the file is overwritten on each subsequent move.

7. If a UNIX file name contains characters that are not allowed in an MVS data set, it will not be moved. If the UNIX file name has more than 8 characters, it cannot be moved to an MVS data set member. (See the -ACMS options for converting file names.)

8. You are not allowed to move files into data sets with spanned records.

9. PDSE cannot have a mixture of program objects and data members. PDS allows mixing, but it is not recommended.

10. Special files such as external links and FIFO will not be moved to MVS data sets. However, you can move character special files to MVS data sets.

11. If a file is a symbolic link, `/mv` will move the resolved file, not the link itself.

12. UNIX file attributes are lost when moving to MVS. If you want to preserve file attributes, you should use the `pax` utility.

**MVS to UNIX:**

1. If the target UNIX file exists, the new data overwrites the existing data. The mode of the file is unchanged (except the S_ISUID and S_ISGID bits are turned off).
2. If the specified UNIX file does not exist, it will be created using a 666 mode value whether the data is binary or text (this is subject to umask). If the data to be moved is a shell script or executable, the UNIX file will be created with a 777 mode value (also subject to umask).

3. For an MVS data set in variable record format RECFM(VB) or undefined record format RECFM(U), trailing blanks are preserved when moving from MVS to UNIX. For an MVS data set in fixed record format, trailing blanks are not preserved when moving from MVS to UNIX.

4. When you move MVS data sets to text files in the z/OS UNIX file system, a <newline> character is appended to the end of each record. If trailing blanks exist in the record, the <newline> character is appended after the trailing blanks. If the file format option is specified or the target file has the file format set, that file format is used as the end-of-line delimiter instead of a <newline> character.

5. When you move MVS data sets to UNIX binary files, the <newline> character is not appended to the record.

6. You cannot use /mv to move data sets with spanned record lengths.

7. Due to an z/OS XL C/C++ run-time restriction, when moving a file from a file system to an MVS sequential data set with the same name and case, you need to prefix the file in the file system with "/". For example:

```
mv ./SMPL.DATA "//''SMPL.DATA''"
```

**Localization**

/mv uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


**Environment variables**

/mv uses the following environment variable when moving records to or from MVS data sets:

- `_EDC_ZERO_RECLLEN`

  If set to Y before calling /mv, an empty record is treated as a single newline character and is not ignored. Also, a single newline is written to the file as an empty record, and a single blank will be represented by " \n " . If you do not set this environment variable when moving records, then the string " \n " is moved the same way as the string " \n ": both are read and written as " \n " . where " \n " indicates that z/OS XL C/C++ will write a record containing a single blank to the file (the default behavior of z/OS XL C/C++ ). All other blanks in the output are read back as blanks, and any empty (zero-length) records are ignored on input.

**Exit values**

- **0**
  Successful completion

- **1**
  Failure due to any of the following:
  - The argument had a trailing / but was not a directory.
  - Inability to find file.
  - Inability to open input file for reading.
• Inability to create or open output file for output.
• Read error on an input file.
• Write error on an output file.
• Input and output files identical.
• Inability to unlink input file.
• Inability to rename input file.
• Unrecoverable error when using the \( -r \) option, such as:
  – Inability to access a file.
  – Inability to read a directory.
  – Inability to remove a directory.
  – Inability to create a directory.
  – A target that is not a directory.
  – Source and destination directories identical.

2
Failure due to any of the following:
• Incorrect command-line option.
• Too few arguments on the command line.
• A target that should be a directory but isn't.
• No space left on target device.
• Out of memory to hold the data to be moved.
• Inability to create a directory to hold a target file.

Messages
Possible error messages include:

cannot allocate target string
\( mv \) has no space to hold the name of the target file. Try to free some memory to give \( mv \) more space.

filename?
You are attempting to move a file, but there is already a file with the target name and the file is read-only. To write over the existing file, type y and press <Enter>. If you do not want to write over the existing file, type n and press <Enter>.

source name and target name are identical
The source and the target are actually the same file (for example, because of links). In this case, \( mv \) does nothing.

unreadable directory name
\( mv \) cannot read the specified directory; for example, because you do not have appropriate permissions.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The -R and -r options are extensions of the POSIX standard.

Related information
\( cp, cpio, zm \)
newgrp - Change to a new group

Format

newgrp [-l] [group]
newgrp [-] [group]

tcsh shell: newgrp [-] group

Description

Use newgrp to change to a new group. You stay logged in and your working directory does not change, but access permissions are calculated according to your new real and effective group IDs. If an error occurs, your session might be ended, and you must log in again.

After the group IDs are changed, a new shell is initialized within the existing process, effectively overlaying the current shell from which newgrp was invoked. The new shell is determined from the initial program value of the OMVS segment of your user profile.

newgrp does not change the value of exported shell variables, and all others are either set to their default or are unset.

If you did not specify any arguments on the command line, newgrp changes to the default group specified for your user ID in the system user database. It also sets the list of supplementary groups to that set in the systems group database.

If you specify a group, newgrp changes your real and effective group ID to that group. You are permitted to change to that group only if your user ID is a member of that group, as specified in the system group database.

group can be a group name from the security facility group database, or it can be a numeric group ID. If a numeric group exists as a group name in the group database, the group ID number that is associated with that group is used.

On systems where the supplementary group list also contains the new effective group ID or where the previous effective group ID was actually in the supplementary group list:

• If the supplementary group list also contains the new effective group ID, newgrp changes the effective group ID.
• If the supplementary group list does not contain the new effective group ID, newgrp adds it to the list (if there is room).

On systems where the supplementary group list does not normally contain the effective group ID or where the old effective group ID was not in the supplementary group list:

• If the supplementary group list contains the new effective group ID, newgrp removes it from the list.
• If the supplementary group list does not contain the old effective group ID, newgrp adds it to the list (if there is room).

newgrp in the tcsh shell, as in the z/OS shell, allows you to change to a new group.

Options

-  
  Starts the new shell session as a login session. This implies that it can run any shell profile code.
  
-  
  Is the obsolete version of -l.
Localization

`newgrp` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


Usage notes

1. The RACF profile `FILE.GROUPOWNER.SETGID` in the UNIXPRIV class sets the group of a new file from the process creating the file instead of from the Directory group information.

2. `newgrp` allows you to change your default group in case RACF list of group checking is not enabled.

3. `newgrp` is not supported from an address space running multiple processes because it would cause all processes in the address space to have their security environment changed unexpectedly. If you are using the OMVS interface, you must be using the NOSHAREAS parameter before you issue the `newgrp` command. Also, if you are running in an environment with the `_BPX_SHAREAS` environment variable set to YES, you must unset it and start a new shell before issuing `newgrp`. For example:

   ```
   unset _BPX_SHAREAS; sh
   ```

Exit values

If `newgrp` succeeds, its exit status is that of the shell. Otherwise, the exit status is:

- `>0`

  Failure because `newgrp` could not obtain the proper user or group information or because it could not run the shell, and it ends the current shell.

Portability

POSIX.2 User Portability Extension, UNIX systems.

Related information

`export`, `fc`, `sh`, `tcsh`

nice - Run a command at a different priority

Format

- `nice [-n number] command-line`
- `nice [-number] command-line`
- `tcsh shell: nice [+number][command]`

Description

`nice` runs a command at a different priority than usual. Normally, `nice` lowers the current priority by 10. The `command-line` must invoke a single utility command, without using compound commands, pipelines, command substitution, and other special structures.
In the tcsh shell, nice sets the scheduling priority for the tcsh shell to `number`, or, without `number`, to 4. With `command`, nice runs `command` at the appropriate priority. The greater the number, the less cpu the process gets. The super-user may specify negative priority by using:

```
nice -number ...```

`command` is always executed in a subshell, and the restrictions placed on commands in simple if statements apply. See “tcsh: Invoke a C shell” on page 671.

**Options**

**-n number**

Lowers the current priority by `number`. On systems supporting higher priorities, a user with appropriate privileges can use nice to increase priority by specifying a negative value for `number`. For example,

```
nice -n -3 command```

runs the command with an increased priority of 3.

**--number**

Is an obsolete version of the `-n number`.

**Localization**

nice uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

If nice invokes the `command-line`, it exits with the exit status returned by `command-line` otherwise its exit status is one of the following:

**1-125**

An error occurred in the `nice` utility.

**126**

nice could not invoke `command-line`.

**127**

nice could not find the utility specified in `command-line`.

**Portability**

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.

**Related information**

nohup, renice, tcsh
nl - Number lines in a file command

Format

```
```

Description

`nl` is a filter that numbers lines in a single file. If you do not specify `file` on the command line, the standard input is used.

The input is displayed as a stream of text lines, possibly divided into logical pages by separators. In turn, each page consists of a header, body, or footer, in that order. Any missing part is assumed to be empty. Using the default page delimiter character of \ and :, lines consisting entirely of the following combinations are logical page part delimiters and are not numbered.

<table>
<thead>
<tr>
<th>Input line</th>
<th>Starts</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\:\:\</code></td>
<td>Page header</td>
</tr>
<tr>
<td><code>\:\</code></td>
<td>Page body</td>
</tr>
<tr>
<td><code>\</code></td>
<td>Page footer</td>
</tr>
</tbody>
</table>

Options

**-btype**

Specifies the numbering type for each page body. The numbering type is one of the following options:

- `a` 
  Numbers all lines

- `n` 
  Does not number any lines

- `pregexp` 
  Numbers only those lines that contain the basic regular expression `regexp`. See Appendix C, “Regular expressions (regexp),” on page 945 for more information about `regexp`.

- `t` 
  Numbers only those lines that are not empty. An empty line consists of only a newline character. The default body numbering type is `t`.

**-dxy**

Changes the default delimiter characters to characters `x` and `y`. If only `x` is specified, only the first delimiter character is changed. The default delimiter characters are \ and :.

**-ftype**

Specifies the page footer numbering type (see the -b option). The default type is `n`.

**-htype**

Specifies the page header numbering type (see the -b option). The default type is `n`. (The lines are not numbered.)

**-in**

Sets the line increment to `n` rather than the default value of `l`.

**-ln**

When the page numbering type is (all), blank lines are treated specially. Every `n`th consecutive blank line is numbered. If you do not specify this option, `n` defaults to 1 and every blank line is numbered.

**-nfmt**

Specifies the line numbering format, which must be one of the following:
nl

n  Right-aligned line number, padded to width (see the -w option) on the left with spaces (the default format).

rz  Right-aligned line number, padded on left with zeroes.

ln  Left-aligned line number, padded on right with spaces.

-p  Specifies continuous page numbering across page boundaries. By default, nl restarts numbering (as in the next option) at each new page.

-ssep  The string sep is printed to separate the line number from the text of the line being numbered. When this option is not specified, this separator is a single tab character.

-vn  Starts numbering for each new page at n. If you do not specify this option, page numbering starts at 1.

-wn  Sets the width of the line number in the output to n. If you do not specify n, the default is 6.

Examples
The following command numbers every second consecutive blank line, using page delimiters of -!:

```bash
nl -l2 -ha -ba -fa -n rz -v10 -i10 -d~! file
```

Localization

nl uses the following localization environment variable:

- LC_SYNTAX
- NLSPATH


Messages

0  Successful completion

1  Failure due to any of the following:
- Incorrect command-line argument.
- More than one file name was specified.
- Unable to open the file.
- Incorrect regular expression in -b, -f, -h.
- Incorrect numbering type.
- Badly formed number in a command-line option.

Portability

POSIX.2, X/Open Portability Guide.

Related information

awk, pr

See Appendix C, “Regular expressions (regexp),” on page 945 for more information about regexp.
nm - Display symbol table of object, library, or executable files

Format

nm [-AaefgMnoPprsuv] [-t format] file ...

Description

nm displays the symbol table that is associated with an object, archive library of objects, or executable files.

By default, nm lists the symbols in file in alphabetical order by name and provides the following information about each:

• File or object name (if you specified -A).
• Symbol name.
• Symbol type. Not all of these symbol types are available on all systems. For instance, not all systems support the ability to determine different segment information.

A
  Absolute symbol, global
a
  Absolute symbol, local
B
  Uninitialized data (bss), global
b
  Uninitialized data (bss), local
D
  Initialized data (bbs), global
d
  Initialized data (bbs), local
F
  File name
l
  Line number entry (see the -a option)
N
  No defined type, global. This is an unspecified type, which is compared to the undefined type. U.
n
  No defined type, local. This is an unspecified type, which is compared to the undefined type. U.
S
  Section symbol, global
s
  Section symbol, local
T
  Text symbol, global
t
  Text symbol, local (static)
U
  Undefined symbol

• Symbol value.
• Symbol size, if applicable.
Options

- A
  Prefixes each line with the file name or archive member.

- a
  Displays all symbols, including line number entries on systems that support them.

- e
  Displays only global (external) and static symbols.

- f
  Displays full output. This is the default because output is not suppressed.

- g
  Displays only global symbols.

- M
  Inserts three columns in the output before each symbol name. The format of these columns is as follows:

  rmode amode compiler_options

  The rmode and amode column will display one of the following:

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>24 bit mode</td>
</tr>
<tr>
<td>31</td>
<td>31 bit mode</td>
</tr>
<tr>
<td>64</td>
<td>64 bit mode</td>
</tr>
<tr>
<td>ANY</td>
<td>ANY mode</td>
</tr>
<tr>
<td>MIN</td>
<td>MIN mode</td>
</tr>
<tr>
<td>---</td>
<td>Undetermined or not/applicable</td>
</tr>
</tbody>
</table>

  The compiler options field shows a character for each compiler option that is determined to be in effect or a dash if none is in effect:

  I
  Symbol is compiled with IPA. Note that IPA is not seen when running nm against an executable because that information is no longer available.

  X
  Symbol is compiled with XPLINK.

- n
  Is equivalent to -v.

- o
  Displays output in octal (same as -t o).

- P
  Displays output in a portable POSIX-compliant format, with blanks separating the output fields.

  - If you specified -A and file is not a library, the format is:

    file: name type value size.

  - If you specified -A and file is a library, the format is:

    file [object_file] : name type value size

    where object_file is the object file in the library that contains the symbol that is being described.

  - If you did not specify -A, the format is:

    name type value size

  - If you did not specify the -t option, nm displays value and size in hexadecimal.
If you did not specify -A and the command line contains more than one file, or file is a library, nm displays a line preceding the list of symbols for each specified file or each object file in a specified library. If file is a library, this line has the following format:

```
file[object_file]:
```

If file is not a library, the format is:

```
file:
```

-p
Does not sort output.

-r
Reverses sort order.

-s
Includes symbol size for each symbol.

-t format
Defines the numeric value formatting base. The format is one of d, o, or x, for decimal, octal, or hexadecimal, respectively. If this option is not used, numbers are displayed in decimal.

-u
Displays only undefined symbols.

-v
Sorts output by value.

-x
Displays information in hexadecimal (same as -t x).

Localization

nm uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLECT
- LC_CTYPE
- LC_MESSAGES
- LC_TIME

See Appendix F, "Localization," on page 965 for more information.

Exit values

0
Successful completion.

1
Failure due to any of the following reasons:

- Invalid command-line option.
- Missing file name.
- Unknown symbol table type.
- Invalid library file.
- End-of-file found in library.
- Bad record in the library.
- Out of memory.
If a file does not contain a symbol table, `nm` displays a warning and goes to the next file, but this is not considered an error.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

The -A, -e, -f, -n, -o, -p, -r, -s, and -x options are not part of the POSIX standard.

The -A, -n, -p, -r, -s, and -t options are not part of the X/Open X/Open standard.

**Related information**

`ar`, `size`, `strip`

---

**nohup - Start a process that is immune to hangups**

**Format**

```
nohup command-line
```

tcsh shell: `nohup command`

**Description**

`nohup` invokes a utility program using the given `command-line`. The utility runs normally; however, it ignores the SIGHUP signal.

If the standard output is a terminal, `nohup` appends the utility's output to a file named `nohup.out` in the working directory. This file is created if it does not already exist; if it cannot be created in the working directory, it is created in your home directory.

If the standard error stream is a terminal, `nohup` redirects the utility's error output to the same file as the standard output.

`nohup` simply runs a program from an executable file. `command-line` cannot contain such special shell constructs as compound commands or pipelines; however, you can use `nohup` to invoke a version of the shell to run such a command line, as in:

```
nohup sh -c 'command*'
```

where `command` can contain such constructs.

In the tcsh shell, with `command`, `nohup` runs `command` such that it will ignore hang up signals. Commands can set their own response to hang ups, overriding `nohup`. Without an argument (allowed only in a shell script), `nohup` causes the tcsh shell to ignore hang ups for the remainder of the script. See “tcsh: Invoke a C shell” on page 671.

**Localization**

`nohup` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH

Exit values

126
    nohup found the utility program but could not invoke it.

127
    An error occurred before nohup invoked the utility, or nohup could not find the utility program.
Otherwise, the exit status is the exit status of the utility program that is invoked.

Portability
POSIX.2, X/Open Portability Guide, UNIX Systems

Related information
exec, hup, nice, sh, tcsh

obrowse - Browse a z/OS UNIX file

Format
obrowse -r xx [file...]

Description
Use obrowse to browse a file in the z/OS UNIX file system. This command uses the TSO/E OBROWSE command and must be run in the foreground. The 3270 passthrough mode is used to invoke the TSO/E OBROWSE command under OMVS.

You can specify any number of files; the TSO/E OBROWSE command is invoked once for each file. If you do not specify a file name, the main entry panel is displayed. From that panel, you can enter the directory name and file name of an existing file you want to browse. If you are browsing fixed-length records, you must also indicate the record length.

The file name can be absolute or relative. Avoid using single quotation marks or parentheses within the file name.

Options
-r xx
    Sets the record length to be browsed for fixed length text files. xx is length. If -r xx is specified, the file is processed as fixed length records. This lets you convert a variable length file to fixed length for viewing.

Environment variables
BPXWISHISPF
    By default, starting in V1R11, the ISPF browse dialog service is used when browsing z/OS UNIX files. Specify BPXWISHISPF=NO if you want obrowse to use the original dialog service.

Usage notes
1. You cannot use obrowse if you used rlogin or telnet to access the shell.
2. obrowse passes the effective UID of its process to the TSO session. If the EUID does not match the EUID of the TSO process, the OBROWSE TSO command will attempt to set the effective UID of the TSO process to that of the shell command prior to loading the file.
Exit values

0
The TSO/E OBROWSE command was invoked once for each file specified.

1
Failure because obrowse could not access at least one file because single quotation marks or parentheses were used in the file name.

2
Failure because obrowse was not able to set 3270 passthrough mode.

od - Dump a file in a specified format

Format

od [-v] [-A addr_fmt] [-j num [bkm]] [-j num] [-T] [-t type_string] [file ... ]
od [-bcDdhOoSsTvXx] [file] [[+offset[. ]][b]]

Description

od (octal dump) dumps a file to the standard output in a format specified by command-line options. The default format is octal words. You can use combinations of options to generate multiple formats with the requested representation of each byte vertically aligned. The file seek address (in octal) precedes each line of new data.

od recognizes two syntaxes. The first one conforms to POSIX. If you choose the first form, od displays files from the list file one at a time. If no file appears on the command line, od reads the standard input.

For a summary of the UNIX03 changes to this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Options

The first form of od accepts the following options:

-v
Displays all lines. Typically, od does not display multiple lines that differ only in the address. It displays the first line with a single * under it. to show that any subsequent lines are the same.

-A addr_fmt
Specifies the format that od uses to display the address field. addr_fmt can be d (decimal), o (octal), x (hexadecimal), or n (do not display address). The default is -A o.

-j num
Skips num bytes from the beginning of the file. If you precede num with 0X or 0x, od interprets it as hexadecimal. If you precede it with 0, od interprets it as octal; otherwise, od assumes that it is decimal. You can also append b, k, or m to num to indicate 512-byte blocks, kilobytes, or megabytes instead of bytes. If num is hexadecimal, any appended b is considered to be the final hexadecimal digit rather than 512-byte block.

Be careful with this option when working with double-byte characters. If byte num+1 (the starting byte, after skipping num bytes) is not the first byte of a character, od proceeds as though it is, resulting in a misinterpretation of that and subsequent characters. This misinterpretation continues until od encounters a <newline>. Then it is once again synchronized with the first byte of a double-byte character.

-N num
Processes a maximum of num bytes. Be careful with this option when working with double-byte characters. If od is processing a double-byte character when it encounters the numth byte and this byte is not the last byte of the character, od displays ??? instead of the character.
-T
Enables automatic conversion for tagged files. This option is mutually exclusive with -t a. For more information about converting files between code pages and file tagging, see z/OS UNIX System Services Planning.

-t type_string
Specifies the output format. type_string can contain the following format characters:

a
Named characters from the ISO 646 character set. Data is interpreted as if it was coded in the ISO 646 character set.

c
Characters. od displays nonprintable characters as backslash sequences and displays printable double-byte characters properly.
A printable double-byte character is displayed in the first byte position, and the remaining positions to the end of the character display ** to indicate the double-byte character. Nonprintable double-byte characters are displayed using a 3-digit octal number to represent each byte.
Also, incorrect double-byte sequences are displayed with ??? for each incorrect byte.

d
Signed decimal. A one-digit number may follow d telling od how many bytes to use. This must correspond to the size of a char (1-byte character), a short (2 byte short), an int (4 byte integer), a long (4 bytes long, which is currently the same as integer on z/OS), or a long long (8 byte integer). The default size is the size of an int. A symbolic size character can follow d, rather than the number of bytes. These have the following meaning:

C
Corresponds to number of bytes in a char

S
Corresponds to number of bytes in a short int

I
Corresponds to the number of bytes in an int

L
Corresponds to the number of bytes in a long int

LL
Corresponds to the number of bytes in a long long int

f
Hexadecimal floating-point. A one-digit number can follow f, telling od how many bytes to use. This must correspond to the size of a float, double, or long double. The default size is the size of a double. A symbolic size character can follow f, rather than the number of bytes. These have the following meaning:

F
Corresponds to size of float

D
Corresponds to size of double

L
Corresponds to size of long double

F
IEEE binary floating-point. A one-digit number can follow F, telling od how many bytes to use. This must correspond to the size of a float, double, or long double. The default size is the size of a double. A symbolic size character can follow F, rather than the number of bytes. These have the following meaning:

F
Corresponds to size of float
D  Corresponds to size of double
L  Corresponds to size of long double

Octal. A one-digit number can follow o, telling od how many bytes to use. This must correspond to the size of a char (1 byte character), a short (2 byte short), an int (4 byte integer), a long (4 byte long, which is currently the same as integer on z/OS), or a long long (8 byte integer). The default size is the size of an int. A symbolic size character can follow o, rather than the number of bytes. These have the following meaning:

C  Corresponds to number of bytes in a char
S  Corresponds to number of bytes in a short int
I  Corresponds to the number of bytes in an int
L  Corresponds to the number of bytes in a long int
LL Corresponds to the number of bytes in a long long int

U  Unsigned decimal. A one-digit number can follow u, telling od how many bytes to use. This must correspond to the size of a char (1 byte character), a short (2 byte short), an int (4 byte integer), a long (4 byte long, which is currently the same as integer on z/OS), or a long long (8 byte integer). The default size is the size of an int. A symbolic size character can follow u, rather than the number of bytes. These have the following meaning:

C  Corresponds to number of bytes in a char
S  Corresponds to number of bytes in a short int
I  Corresponds to the number of bytes in an int
L  Corresponds to the number of bytes in a long int
LL Corresponds to the number of bytes in a long long int

X  Hexadecimal. A one-digit number can follow x, telling od how many bytes to use. This must correspond to the size of a char (1 byte character), a short (2 byte short), an int (4 byte integer), a long (4 byte long, which is currently the same as integer on z/OS), or a long long (8 byte integer). The default size is the size of an int. A symbolic size character can follow x, rather than the number of bytes. These have the following meaning:

C  Corresponds to number of bytes in a char
S  Corresponds to number of bytes in a short int
I  Corresponds to the number of bytes in an int
L  Corresponds to the number of bytes in a long int
LL
Corresponds to the number of bytes in a long long int

Multiple format characters can appear in one type_string and multiple -t options can appear on the command line. If there is no -t option, the default is -t oS.

-t a is mutually exclusive with the -T option.

The second form of od is the historical (Berkeley Software Distribution) implementation of the command. If you use this form, you can specify only a single input file. If you do not give a file argument, od reads the standard input. You can supply an offset, but you must precede it with a plus sign (+) to distinguish it from a file name if no file is given. Giving an offset causes a seek to a position in the file where output begins. If the offset ends in a period (.), od considers it to be decimal; otherwise, od considers it octal. If you follow the offset with a b, od multiplies it by the block size of 512 bytes. The format of the offset determines the format of the address; that is, if it is interpreted as decimal, the addresses are displayed in decimal.

The od command does not work on a file whose file name starts with either a digit or a plus (+) sign, unless the -A, -N, -j, or -t options are used.

The second form of od accepts the following options:

-b
  Bytes in octal format
-c
  Bytes as characters
-D
  Unsigned decimal longs (4 bytes)
-d
  Unsigned decimal words (2 bytes)
-h
  Bytes in hexadecimal format
-O
  Unsigned octal longs
-o
  Unsigned octal words
-S
  Signed decimal longs
-s
  Signed decimal words
-T
  Enables automatic conversion for tagged files. This option is mutually exclusive with -t a. For more information about converting files between code pages and file tagging, see z/OS UNIX System Services Planning.
-v
  Displays all lines. Typically, od does not display multiple lines that differ only in the address. It displays the first line with a single * under it to show that any subsequent lines are the same.
-X
  Unsigned hexadecimal longs
-x
  Unsigned hexadecimal words

Localization
od uses the following localization environment variables:
- LANG
- LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_NUMERIC
• LC_SYNTAX
• NLSPATH


**Exit values**

0  
Successful completion

1  
Failure due to any of the following:
• Inability to open the input file
• Badly formed offset
• Seek or read error on the input file

2  
Failure due to any of the following:
• Incorrect command-line argument
• The wrong number of command-line arguments
• Incorrect format character
• Incorrect size modifier for format character

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

The options to operate on longs (-OSXD) and the hexadecimal byte (-h) are extensions to the POSIX standard.

The –T option is an extension to the POSIX standard.

**Related information**

dd

**oedit - Edit files in a z/OS UNIX file system**

**Format**

`oedit [-r xx] [file...]`

**Description**

Use `oedit` to edit a file in the z/OS UNIX file system. This command uses the TSO/E OEDIT command and must be run in the foreground. The 3270 passthrough mode is used to invoke the TSO/E OEDIT command under OMVS.

You can specify any number of files; the TSO/E OEDIT command is invoked once for each file. If you do not specify a file name, the Edit Entry panel is displayed. From that panel, you can enter the directory name and file name of an existing file, or you can specify a directory name and file name for a new file. The Edit Entry panel also lets you specify an edit profile and an initial edit macro.
The file name can be absolute or relative. Avoid using single quotation marks or parentheses within the file name. Avoid using spaces or single quotation marks within path names.

Options

-\( \text{r } xx \)

Set the record length to be edited for fixed-length text files. \( xx \) is the record length.

If -\( r \) \( xx \) is specified, the file is processed as variable length but loaded into the editor as fixed-length records and saved as fixed-length records. This lets you convert a variable length file to fixed length. If any lines are longer than the specified record length, the edit session will not load the file and will issue the customary message that a line is too long.

Usage notes

1. ASCII files must be tagged as ISO8859-1 in order for oedit to automatically translate the file. Do not enter the OEDIT session and type SOURCE ASCII.

2. oedit attempts to load the file into a VB255 session. If this is an ISPF that supports wide edit (such as ISPF 4.1) and any line exceeds 235 characters, the width for the new session is the length of the longest line plus 25% to allow for some expansion.

3. The COPY command cannot copy in files that have records wider than the edit session.

4. oedit attempts to open an existing file as read/write. If this fails, it attempts opening the file read-only to allow the user to view the file. Changes that are made in this mode cannot be saved to the file. If changes are made, the edit session must be ended using the ISPF CANCEL primary command. However, you can use the ISPF CREATE and REPLACE primary commands to save all or part of the changed file to another file before you cancel the edit session.

5. oedit passes the effective UID of its process to the TSO session. If the EUID does not match the EUID of the TSO process, the OEDIT TSO command attempts to set the effective UID of the TSO process to that of the shell command before loading the file.

6. You cannot use oedit if you used rlogin or telnet to access the z/OS shell.

7. The TSO region size must be large enough to hold the size of the file to be edited.

8. Two ISPF variables are available to edit macros:
   - HFSCWD, which contains the path name for the directory in which the file that is being edited resides.
   - HFSNAME, which contains the name of the file that is being edited.

Environment variables

BPXWISHISPF

By default, starting in V1R11, the ISPF edit dialog service is used for editing z/OS UNIX files. Specify BPXWISHISPF=NO if you want oedit to use the original dialog service.

BPXWPERM

Specifies the default open permissions that are used by oedit. Permissions are specified in octal format. The supplied permissions are not validated and the number is used as the file mode on an open() call. If the file already exists, the permissions are not changed. If the environment variable is not set, oedit works as before, using 0700 as the default permissions.

Exit values

0

The TSO/E OEDIT command was invoked for each specified file.

1

Failure because oedit could not access at least one file because single quotation marks or parentheses were used in the file name.
Failure because `oedit` could not set the 3270 passthrough mode.

**pack - Compress files by Huffman coding**

**Format**

```
pack [-][Bf] [-o file] file ...
```

The `pack` utility is fully supported for compatibility with older UNIX systems. However, the `compress` utility should be used instead because it might provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

**Description**

`pack` compresses files using a Huffman minimal redundancy code on a byte basis. Each file is compressed in place; the resulting file has a `.z` extension appended to the file name, but keeps the same owner and permissions. For example, `abc` is compressed into `abc.z`. The times of last access and last modification are also preserved.

Packed files can be identified by `file` and uncompressed by `unpack` (whichunpacks the file in place) or `pcat` (which unpacks to the standard output).

Normally `pack` reports the degree of compression achieved in each file (the report is printed on standard output). This number can be negative for small files with little redundancy if the `-f` option is used.

`pack` does not pack files if:

- The file appears to have already been packed.
- The file name is too long; an error occurs if `.z` is appended.
- The file has links or is a directory.
- The packed file would be larger than the existing file (this includes empty files).
- The destination file already exists, or there is an error in processing.

**Options**

- Displays more detail on size, overhead, and entropy (information rate). If this option is used several times on the command line. It acts as a toggle, inverting the detailed-report flag at each mention.

- **B**
  Disables the automatic conversion of tagged files.

- **f**
  Forces compression when it typically would not occur. Without this option, `pack` does not compress a file if its size is not reduced by compression, the file is already compressed, or the file has more than one link.

- **o file**
  Specifies a different output file so that compressed output is written to `file` rather than overwriting the original input file. Several input and output files can be specified. For example,

  ```
  pack -o out1 in1 -o out2 in2
  ```

  packs file `in1` into `out1` and file `in2` into `out2`. The input files are not changed.

**Localization**

`pack` uses the following localization environment variables:

- `LANG`
Environment variables

pack uses the following environment variable:

_TEXT_CONV
Contains text conversion information for the command. The text conversion information is not used when the -B option is specified. For more information about text conversion, see Appendix L, "Controlling text conversion for z/OS UNIX shell commands," on page 993.

Exit values

0
  Successful completion.
1
  An error occurred due to one of the following:
  • A problem related to manipulating (opening, closing, renaming) the file, or a single file could not be packed properly.
  • Could not turn off automatic conversion.

n
  Indicates that n files could not be packed properly. For example, if three out of six files could not be packed properly, the exit status is 3.

file: no saving
The file is too small or uniform to benefit from packing. The file can still be packed using the -f option.

file: already packed
The file appears to be a packed file. It can still be packed by specifying the -f option.

file: has links
The file has more than one link. You can override it with the -f option.

file: directory
pack cannot modify directories.

file: empty
The file is empty.

file: can't pack in place
The file is too large to pack in place. Use the -o option to specify an output file.

Interrupt
If you press BREAK while pack is running, it does not stop immediately; if it did, it would leave you with a corrupted file. Thus pack just displays this message to show that the BREAK has been received and it stops as soon as it is safe to do so.

Other messages, such as those about inaccessibility of files, are self-explanatory. The exit status is the number of file arguments that could not be processed.

Portability
X/Open Portability Guide, UNIX System V.

The -B and -o options are extensions of the POSIX standard.
passwd - Change user passwords or password phrases

**Format**

```
passwd [-u userid]
```

**Description**

`passwd` changes the login password or password phrase for the user ID specified. If `userid` is omitted, or if the `-u` option is omitted, the login name that is associated with the current terminal is used. You are prompted for the new password or password phrase, which might be truncated to the length defined as the maximum length for the passwords.

Users can change the password or password phrase for another user if they know the user ID and current password or password phrase.

**Examples**

1. To change your password or password phrase, issue:

   ```
   passwd
   ```

   You are prompted for the old password or password phrase and the new password or password phrase.

2. To change the password or password phrase for user ID Steve, issue:

   ```
   passwd -u steve
   ```

   You are prompted for the old password or password phrase and the new password or password phrase.

**Exit values**

0  The password or password phrase was changed.

1  Failure due to any of the following reasons:
   - The specified user does not exist.
   - The current password or password phrase is incorrect.
   - The new password or password phrase does not meet the requirements of the installation exit.

2  The new password or password phrase was not entered the same way twice.

4  Error obtaining user login name.

paste - Merge corresponding or subsequent lines of a file

**Format**

```
paste [-Bs] [-d list] [-W option[option] ... file ...
```
Description

paste concatenates lines of all the specified input files onto the standard output. If you specify - (dash) instead of a file, paste uses the standard input. Typically, an output line consists of the corresponding lines from all the input files. paste replaces the newline character at the end of each input line (except the one from the last file on the command line) with a tab character, or characters specified by the -d option.

Options

-B
Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-d list
Specifies a list of characters to be used one at a time instead of the tab character to replace the newline at the end of input lines. In a double-byte locale, list can contain double-byte characters. paste uses list circularly; when it exhausts the characters in list, it returns to the first character in the list. If you also specify the -s option, paste returns to the first character of list after each file is processed. Otherwise, it returns to the first character after each line of output.

list can contain any of the following standard C escapes such as \n, \t, \r, \b, \, and \0, where \0 indicates that no separator is to be used.

-s
Concatenates all lines from each input file together on the single output line. If the -s option is not specified and the end of the file is detected on any (but not all) of the input files, paste behaves as though empty lines have been read from those files.

-W option[.option]...
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

filecodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). The command iconv -1 lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -s option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

pgmcodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). The command iconv -1 lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion.
If `filecodeset` or `pgmcodeset` is specified, then automatic conversion is disabled for this command invocation and the `-B` option is ignored if it is also specified. See `z/OS UNIX System Services Planning` for more information about automatic conversion.

The only supported values for `pgmcodeset` are IBM-1047 and 1047.

**Examples**

1. To display the `ls` output in three tab-separated columns:
   ```
   ls | paste - - -
   ```

2. To concatenate lines in two text files containing UTF-8 characters, assuming that
   - The text files are untagged and you do not want to tag them or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying untagged public text files or read-only text files)
   then issue:
   ```
   paste -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File01 myUtf8File02
   ```

3. To concatenate lines in three text files containing EBCDIC characters, assuming that automatic conversion has been enabled but the text files are incorrectly tagged as ASCII:
   ```
   paste -B myMisTaggedFile01 myMisTaggedFile02 myMisTaggedFile03
   ```

4. If file A contains:
   ```
   a
   b
   c
   ```
   and file X contains
   ```
   x
   y
   z
   ```
   then the command:
   ```
   paste A X
   ```
   produces:
   ```
   a      x
   b      y
   c      z
   ```
   and the command:
   ```
   paste -s A X
   ```
   produces:
   ```
   a      b      c
   x      y      z
   ```

**Localization**

`paste` uses the following localization environment variables:

- `LANG`
Environment variables

paste uses the following environment variable:

_TEXT_CONV
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

Exit values

0
Successful completion

1
Failure due to any of the following:
• Too many files specified
• Inability to open a file
• The code set is not valid
• Could not turn off automatic conversion
• Could not perform requested text conversion

2
Failure due to any of the following reasons:
• Incorrect command-line option
• Missing input files

Messages

Possible error messages include:

Too many files at name
You specified more files than paste can handle. The name given in the error message is the name of the first file that paste could not open. The number of files that paste can open depends on the number of files that other processes have open.

Portability

POSIX.2, X/Open Portability Guide, UNIX System V.
The -B and -W options are extensions of the POSIX standard.

Related information

cut
patch

patch - Change a file using diff output

Format

Description
patch reads a patchfile that contains output from diff describing changes from an old text file to a new text file. patch then applies those changes to another text file. Typically, you use patch if you are keeping parallel versions of a file. When you make a set of changes to one file, you can use patch to incorporate those same changes in other versions of the file.

Options
-B prefix
Saves a copy of the original file in a backup file. The backup file name is the name of the original file preceded by the string prefix. If there is already a file with this name, patch overwrites it. When applying more than one patch to the same file, patch copies only the original for the first patch. When you also specify -o outfile, patch does not create prefixfile, but if outfile already exists, it creates prefixoutfile.

-b
Saves a copy of the original file in a backup file. The backup file name is the name of the original file plus the suffix orig. If there is already a file with that name, patch overwrites it. When applying more than one patch to the same file, patch only creates file orig. When you also specify -o outfile, patch does not create file-.orig, but if outfile already exists, it creates outfile .orig.

-c
Interprets the patchfile as a context diff file (the output of diff when -c or -C is specified). You cannot use this option with -e or -n.

-D symbol
Marks changes with the C preprocessor construct:

```
    #ifdef symbol
    #endif
```

When you compile the resulting (patched) file, you get the original file if symbol is not defined, and the changed file if symbol is defined.

-d dir
Changes the current directory to dir before processing the patch.

-e
Interprets the patchfile as an ed script (the output of diff when -e is specified). You cannot use this option with -c or -n.

-F n
Specifies the number of lines of a context diff to ignore when searching for a place to install a patch.

-f
Forces processing without requesting additional information from the user.

-i patchfile
Reads the patchfile information from the file patchfile. If you do not specify patchfile, patch reads the information from the standard input.

-l
Matches any sequences of blanks in the patchfile to any sequence of blanks in the input file. In other words, patch considers two lines equivalent if the only difference between the two is their spacing.
-N
Ignores any patches that have already been applied. By default, patch rejects already-applied patches.

-n
Interprets the patchfile as normal diff output. You cannot use this option with c or -e.

-o outfile
Writes patched output to outfile instead of to the original file. When you specify more than one patch to a single file, patch applies the patches to intermediate versions of the file created by previous patches, resulting in multiple, concatenated versions of the file being written to outfile.

-p n
Deletes n components from the beginning of all path names found in the patch file. If a path name is an absolute path name (that is, starts with a slash), patch treats the leading slash as the first component of the path, and patch -p 1 deletes the leading slash. Specifying -p 0 tells patch to use the full path names given in the patchfile. If you do not specify this option, patch only uses the basename (the final path component).

-R
Reverses the sense of the patch script. In other words, patch behaves as if the patch script shows the changes that make the new version into the old version. You cannot use -R if the patchfile is in ed script format.

With -R, patch attempts to reverse each change recorded in the script before applying the change. patch saves rejected differences in reversed format (which means that you can check the rejections to see if patch made the reversals correctly).

-r rejectfile
Records rejects in the file rejectfile, instead of the default reject file name.

-s
Tells patch to remain silent until an error occurs. Normally, patch writes information about the results of the patching process to standard error (stderr).

-v
Displays the version number of patch and then exits.

If you do not specify either the -b or -B option, patch attempts to change the original file directly. If you do not specify -c, -e, or -n, patch looks at the format of the diff output and tries to determine which type of output the patch file contains.

If you do not specify a file to be patched and the patchfile is not in context format, patch prompts you for the name of the file you want to patch.

If the patchfile is in context format, patch tries to determine the file name on its own. The first two lines of a context patch file give the names of the old and new files that diff compared. If only one of the files exists, patch patches that file; if neither exists or both do, patch checks for a line starting with a string Index: before asking you for the name of the file to patch. If both files exist but one of them is empty, the empty file will automatically be patched.

After patch has determined the file to patch, it checks for a source control system (SCCS) subdirectory in the current directory; if one exists, it tries to obtain an editable version of that file by performing the source code control system (SCCS) command get –e. If patch cannot determine the file to patch, it prompts you for the name of the file to use.

With a context format patchfile, patch can recognize when line numbers given in the patchfile do not match line numbers in the file being patched. Thus, it can patch a file with line counts that do not match the old file that was used by diff. To do this, it takes these steps:

1. For each section to be changed, patch starts with the line number found in the patch file, plus or minus any adjustment that must be made for the previous section.
2. If the line at this location does not match the line in the patch file, patch scans forward and backward for a line that does match. If it finds a matching line, patch proceeds to make the required changes.
patch also remembers the adjustment it had to make to find the matching line, and uses this adjustment in the next section to be changed.

3. If patch cannot find a line matching the one in the patch file, it tries to find the line using the lines given as context. It ignores the first and last two lines of the context and goes searching again. If it finds a match this time, it makes the change and remembers the adjustment.

4. If a search ignoring the first and last lines of the context fails, patch searches one more time, ignoring the first two and last two lines of the context. If it finds a match, it makes the changes and remembers the adjustment.

5. If patch still cannot find a match, it writes the unmatching portion to the reject file. It then tries to process the next section of changes. Thus, the reject file contains the sections that patch is not able to change. Line numbers on sections in the reject file may be different than those in the patchfile, because patch adjusts them using the adjustment that patch calculated for preceding sections.

To some extent, patch tries the same process if the patch file is in normal format rather than context format. Because the patch file does not contain the context information, patch has less to work with and probably creates more rejects. patch always writes the rejectfile in context format, regardless of the format of the patchfile.

By default, the reject file has the same name as the original file, plus the suffix .rej. You can use -r to specify a different reject file on the command line. If the reject file already exists, patch overwrites it.

If you do not specify -R, patch starts out with the assumption that the patch file could be normal or reversed. Therefore if the first change is rejected, patch tries the reverse change to see if that one works. If the reverse change is also rejected, patch continues with other changes in the file, trying both forward changes and reverses until one of them works. If the one that works is a forward change, patch attempts only forward changes for the rest of the file. If the one that works is a reverse change, patch issues a message to this effect and ask if it should treat all the changes as reverse ones. However, if the -R option is specified on the command line, it is assumed to hold for all changes in the patch file.

The patch file can contain output from several diff comparisons. patch treats each collection of changes as a separate patch file, and with each, patch may prompt you for the name of the file you want to patch.

Localization

patch uses the following localization variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_TIME
- LC_SYNTAX
- NLSPATH


Exit values

0
    Successful completion.

1
    There were one or more rejects.

>1
    An error occurred.
Portability
POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
The -B, -F, -f, -s, and -v options are not extensions to the POSIX standard.

Related information
diff, ed

pathchk - Check a path name

Format
pathchk [-p] pathname ...

Description
pathchk checks one or more path names (specified by pathname) for validity and portability (based on the underlying file system). A path name is valid if you can use it to create or access a file without causing a syntax error. A path name is portable if the file system does not truncate the name when it tries to use it. pathchk writes an error message indicating the error that was detected and the erroneous path name if any path name is longer than PATH_MAX bytes or contains any of the following:

• A component longer than NAME_MAX bytes.
• Any component in a directory that is not searchable.
• Any character in any component that is not valid.

Options
-p
Instead of using the previous criteria, writes an error message if pathname:

• Is longer than _POSIX_PATH_MAX bytes.
• Contains any component longer than _POSIX_NAME_MAX bytes.
• Contains any character in any component that is not in the portable file name character set.

Localization
pathchk uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Exit values
0
All path names passed the check
1
An error occurred.
2
Unknown command-line option.
Portability

POSIX.2, X/Open Portability Guide.

**pax - Interchange portable archives**

**Format**

```plaintext
pax [-cdEnvz][-H][-L][-f archive] [-o type] [-s substitute] ... [pattern ...]
pax -r [-cdEiknuqvxz] [-H][-L] [-f archive] [-o options ...] [-p string ...] [-s substitute ...] [-v volpat] [pattern ...]
pax -w [-dEituvXz] [-H][-L] [-w seaparms=parms] [-b blocksize] [[-a][-f archive]] [-o options ...] [-s substitute ...] [-v volpat] [-x format] [pathname ...]
pax -r -w [-CdDEikLIMntuvX] [-H][-L][-o options ...] [-p string ...] [-s substitute ...] [pathname ...] directory
```

The `pax` interchange format (`-x pax`), which is a standard UNIX format, stores all file attributes that extended USTAR (`-o saveeext`) or os390 format (`-x os390`) does. It can also save and restore file attributes that cannot be handled by any other format such as: files greater than 8 GB, UID, and GID values greater than 2097151 large time values (see the usage notes), and z/OS-specific attributes like user audit and auditor audit flags and file format. The `pax` interchange format is supported on z/OS release 8 and later. `pax` interchange format archives can be extracted on previous systems; however, there will be loss of information for archived files that have attributes that cannot be stored in USTAR format. If you are creating archives that might be extracted on previous z/OS systems, it is recommended that USTAR (default), extended USTAR (-o saveeext) or os390 (-x os390) format be used. If you are creating archives that will be extracted on z/OS release 8 systems and later, the `pax` format (`-x pax`) is the recommended format. For more information about preserving extended attributes with the `pax` format, see `-x pax`.

**Description**

`pax` reads, writes, or lists an archive file, or copies directory hierarchies. An archive file can be a UNIX file or a z/OS data set or a z/OS data set member. A file that is stored inside an archive is called a component file. Similarly, a directory that is stored inside an archive is called a component directory.

Included with each component file and directory is recorded information such as owner and group name, permission bits, file attributes, and modification time.

If you are using `pax` keep these restrictions in mind.

- `pax` supports the use of generation data groups (GDGs) and tape data sets. To use those MVS data sets, the user must be in read or write mode. If the user is in copy or append mode, the `pax` command does not support the use of those MVS data sets.
- MVS data sets cannot be specified for component files.
- If you are writing to a MVS data set, the `-f archive` parameter must be used to identify the data set.

You can therefore use a single archive file to transfer a directory structure from one machine to another, or to back up or restore groups of files and directories.

- `pax`

As shown in “Format” on page 510, `pax` performs one of the four archive functions based on the usage of the `-r` and `-w` options:

**list**

If you do not specify `-r` or `-w`, you are in list mode. In this mode, `pax` uses the standard output to display the table of contents of an existing archive file. The `-v` (verbose) and `-E` options can be used to show the file attributes (to include file tags and ACLs) and extended attributes of each component. By default, `pax` displays all component files and directories that are contained in the archive. One or more patterns can be used to display information about specific components.
read
If you specify -r but not -w, you are in read mode. In this mode, pax reads an archive file as input and extracts components from the archive. By default, pax selects all components. Patterns can also be used to identify specific components to extract. If the archive contains several components with the same name, pax extracts each of them with later components overwriting files that are created by earlier components with the same name. The -k, -n, or -u options can be used to control the extraction of files when multiple files with the same name exist in the archive or on the file system.

pax can read input archives in cpio, tar, OS390, and GNU tar format. For the special entry types in the GNU tar format, pax only supports the long pathname or long linkname. Other special entry types in the GNU tar, such as archiving the sparse file, are not supported.

When extracted, if a component does not have a fully qualified path name beginning with the root (/) directory, its path is assumed to be relative to the current working directory. The -s or -i options can be used to dynamically change the path name of extracted components. Ownership, permissions, file attributes (such as file tags and ACLs), and extended attributes of the extracted files are discussed under the -p option.

write
If you specify -w but not -r, you are in write mode. In this mode, pax creates an archive file that contains the specified path name as components. If a path name is a directory, pax writes to the archive file all the files and subdirectories in that directory. If you do not specify any path name, pax reads the standard input to get a list of path name to select; the input should give one path name per line.

The -d, -X, and -L options can be used to restrict path name to the current directory or device, or to follow symbolic links.

The -a (with -w) option can be used to append to an existing archive.

copy
If you specify both -r and -w, you are in copy mode. In this mode, pax reads the specified path names and copies them to the target directory. In this case, the given directory must exist and you must be able to write to that directory. If a path name is a directory, pax copies all the files and subdirectories in that directory as well as the directory itself. If you do not specify any path name, pax reads the standard input to get a list of path names to copy; the input should give one path name per line. pax is only carried out in the pax (-x pax) format.

The name of the archive file can be specified with the -f archive option. If -f is not used, pax will read from standard input for the list and read (-r) functions and will write to standard output for the write (-w) function.

pax can read input archives in cpio, tar, OS390 and GNU tar format. It can also write these formats except the GNU tar format; see the -x option. For the special entry types in GNU tar format, pax only supports reading the long pathname or long linkname. Other special entry types in the GNU tar format, such as archiving the sparse file, are not supported.

Patterns
Command-line patterns are similar to the wildcard constructs described in the sh command. You can use them to select specific components when reading or listing an archive.

Slash characters in a path name must be explicitly matched by using one or more slashes in the pattern; it cannot be matched by the asterisk (*) or question mark (?) special characters or by a bracket expression. For example, the pattern "*.c" will only match files in the archive with name that are not preceded by a slash. The pattern "/*.*.c" will match files in the archive preceded by a single slash.

Tip: To prevent the shell from first expanding them, put quotation marks around patterns. For example, if the pattern ".h" is not quoted, the shell will first resolve it into the list of files in the current directory ending with ".h". If there are none, the shell will replace "h" with an empty list and pax will then list every component in the archive because no pattern is specified. If one or more ".h" files are returned by the
shell, **pax** will list only those components in the archive matching the `.h` files found in the current directory.

**pax** does not support patterns when writing or copying. However, wildcards can be used in specifying the path name with the write or copy function because the shell will first expand them before passing the results to **pax**.

The `-c` option can be used to select files that do not match the pattern.

---

**Options for the pax command**

The following options might appear on **pax** command lines. Some of them are appropriate to only some forms of the command, as shown in “Format” on page 510.

- **-a**
  
  Appends specified files or directories to the end of the contents of an existing archive. If the archive does not exist, **pax** creates it.

  **Restrictions**: Compressed archives and archives in MVS partitioned data sets cannot be appended. Also, archives in OS390 format cannot be appended to archives in non-OS390 format, and archives in non-OS390 format cannot be appended to archives in OS390 format.

- **-b** *blocksize*
  
  Specifies the block size in an output operation. Each output operation writes *blocksize* bytes, where *blocksize* is an integer appropriate to the output device. If b follows the *blocksize* number, the block size is the given number of 512-byte blocks. If k follows the blocksize number, the block size is the given number of 1024-byte blocks. The default blocksize is 10 K for tar archives and 5 K for cpio archives. For utox, pax, and os390, the default block size is 32,256 bytes.

  The output size will always be in multiples of the block size. Therefore, the minimum output archive size will be equal to the block size.

  **Rule**: The block size must be at least 512 bytes for reading.

- **-c**
  
  Causes **pax** to continue after an error on the source file system is encountered. **pax** will print an error message and return a nonzero value after the command ends. Errors on the target file system (such as out of space or write errors) will still cause the **pax** command to end as it always has.

  The socket-type file is supported with the `-c` option in the copy mode.

- **-c**
  
  Selects all those files that do not match any of the patterns that are given on the command line; this is the opposite of the usual behavior. If a pattern is not given, then no files will match.

- **-D**
  
  Files will not be created sparse in the target directory tree. Sparse files are those that do not use real disk storage for pages of file data that contain only zeros, which save on disk space. When those files are opened and read, the file system returns zeros for those portions of the files that do not have real disk storage. The default for **pax** is to copy all files as sparse, whether the original file was sparse, if sparse files are supported on the target file system.

  **Restriction**: The `-D` option is only for **pax** copy mode.

- **-D**
  
  Does not traverse directories. A pattern matching a directory extracts only the directory itself. When an archive is created, a directory name stores only the directory itself.

- **-E**
  
  Same as verbose `-v` output, but additionally displays extended attributes. See “Output” on page 523 for more information. `-o` E is equivalent to **pax** `-E`. 
-f archive
Specifies the name of the archive file instead of using the standard input for list mode, read mode (-r operations), and the standard output for write mode (-w). The archive file that you specify can be an MVS data set. For more information, see Appendix K, “Specifying MVS data set names in the shell environment,” on page 991.

Tip: Avoid writing to an archive that is in the directory tree or the set of files that is being archived. Doing so causes pax to write the archive to itself and results in unpredictable results during the write or later during a read.

-H
Follows symbolic links that are specified on the command line only. When you specify this option, pax copies the file pointed to by a symbolic link to an archive. The exception is if a symbolic link on the command line points to another symbolic link. A chain of symbolic links is followed to the end. Symbolic links that are encountered during tree traversal are not followed; the symbolic link itself is archived. The default behavior is to archive the symbolic link itself.

Specifying more than one of the mutually exclusive options -H and -L is not considered an error and the last option that is specified determines the behavior of the utility.

-i
Renames files as pax works. With extractions, pax displays the name of the component it is about to extract and gives you the chance to specify a name for the extracted file. With write operations, pax displays the name of the file or directory it is about to record in the archive, and lets you specify a different name to be assigned to the component. If you enter . as the name, pax processes the file or directory with no change to the name. If you just press <Enter>, pax skips the file (does not extract or archive it). pax ends if you enter end-of-file.

If you also specify -s, pax makes the given substitution before displaying the name of the component.

-k
Prevents the overwriting of existing files.

-L
Follows symbolic links. When you specify this option, pax copies the file to which a symbolic link points to the archive. Normally, only the symbolic link is copied.

Specifying more than one of the mutually exclusive options -H and -L is not considered an error and the last option that is specified determines the behavior of the utility.

-\n
Is applicable only when you are in copy mode, such as when you are using the -rw format to copy files to another directory. If you specify -L, pax creates links to the original files whenever possible, rather than copying them.

-M
Creates empty directories within the target directory tree for each active mount point encountered within the source directory tree. pax identifies mount points by checking whether a subdirectory in the source tree is on the same device as the parent current directory. This behavior is like the current pax -X option (write out only those files and directories that are on the same device as their parent directory) except instead of skipping the subdirectory entirely a corresponding empty directory is created in the target directory tree. Any contents in the subdirectory on the source directory tree are ignored.

Restriction: The -M option is only for pax copy mode.

-n
Treats the pattern arguments as ordinary path names. You can use this option only when you specify -r but not -w. pax extracts only the first component with a given path name, even if the archive contains several components with the same name. pax checks the given path names against the archive before applying any renaming from the -i, or -s options. pax extracts writes an error message for each specified file that cannot be found in the archive.
-o options
Provides information for modifying the algorithm for writing and extracting files.

The following set of options controls the use of z/OS extended USTAR support for the USTAR, OS390 format and pax format to preserve, restore, and display z/OS-specific information such as external links, extended attributes, file tag information, ACLs, and other information (long link names, for example) not otherwise supported by the USTAR format. When you use the OS390 and pax format in extract mode, those z/OS-specific attributes are saved by default. For more information about extended USTAR support, see “z/OS-extended USTAR support” on page 532.

- o keyword[[:]=value][,keyword[[:]=value], ...]
The value of options shall consist of one or more keywords or keyword/value pairs.

Multiple keywords or keyword/value pairs that are specified to a single -o option can be separated by a comma or a space unless the environment variable _UNIX03=NO is used, then this must be a comma-separated list. Some keywords apply only to certain file formats, as indicated with each description. Use of keywords that are inapplicable to the file format being processed will be ignored by pax.

If _UNIX03=YES is not used, then keywords can be preceded with white space and the value field consists of zero or more characters. Within value, any literal comma must be preceded with a backslash (\). A comma as the final character, or a comma followed solely by white space at the end of options is ignored.

Multiple - o options can be specified. If keywords given to these multiple -o options conflict, the keywords and values that appear later in command-line sequences take precedence.

If the -x pax format is specified, any of the keywords and values that are defined in “Extended header keywords” on page 525 and are in the following list can be used in - o options in either of two modes:

  keyword=value
When used in write or copy mode, these keyword-value pairs are written into the global extended header records of the new archive. When used in read or list mode, these keyword-value pairs act as if they were present in the global extended header records of the archive being read. In both cases, the given value is applied to all files that do not have a value that is assigned in their individual extended header records for the specified keyword.

  keyword:=value
When used in write or copy mode, these keyword-value pairs are written into the extended header records of each file in the new archive. When used in read or list mode, these keyword-value pairs act as if they were present in the extended header records of each file in the archive that is being read. In both cases, the given value overrides any value for the specified keyword found in the global or file-specific extended header records.

For example:

```
pax -f -o "gname:=mygroup" <archive>
```

the group name is forced to a new value for all files read from the archive.

The following keywords are supported:

A
Displays data for the extended access control list (ACL). For more information about ACLs, see z/OS UNIX System Services Planning and “pax support for access control list (ACL)” on page 534.

Specifying pax - o A does not automatically turn on the verbose table of contents format. You must also specify - v to display the file permission bit settings that are associated with the file.

atime=value
See “Extended header keywords” on page 525.

charset=value
See “Extended header keywords” on page 525.
**comment=value**

See “Extended header keywords” on page 525.

**delete=pattern**

(Applicable only to the -x pax format.) When used in write or copy mode, pax omits any keywords matching the string pattern from the extended header records. When used in read or list mode, pax ignores any keywords matching pattern in the extended header records. For example:

```
-o delete=realtime.*
```

suppresses information that is related to the realtime keyword. When multiple \(-o delete=pattern\) options are specified, the patterns are additive; all keywords matching the specified string patterns are omitted from extended header records that pax produces. Patterns follow the same rules given in “File name generation” on page 604.

**E**

Shows extended attributes when the archive table of contents is being displayed. Automatically turns on \(-v\). This option is synonymous with the pax \(-E\) option.

**exthdr.name=string**

(Applicable only to the -x pax format.) This keyword allows user control over the name that is written into the USTAR header blocks for the extended header produced under the circumstances described in “pax header block” on page 975. The name is the contents of string, after the following character substitutions have been made:

<table>
<thead>
<tr>
<th>string includes:</th>
<th>Replaced by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>%d</td>
<td>The directory name of the file, equivalent to the result of the dirname utility on the translated path name.</td>
</tr>
<tr>
<td>%f</td>
<td>The file name of the file, equivalent to the result of the basename utility on the translated path name.</td>
</tr>
<tr>
<td>%p</td>
<td>The process ID of the pax process.</td>
</tr>
<tr>
<td>%%</td>
<td>A % character.</td>
</tr>
</tbody>
</table>

Any other % characters in string produce the character itself. For instance, %s prints the character 's'.

If no \(-o exthdr.name=string\) is specified, pax uses the following default value:

```
%d/PaxHeaders.%p/%f
```

**from=codeset**

\(\text{from}=\text{codeset}\) is typically used with \(\text{to}=\text{codeset}\).

Converts data from one code set to another while reading or writing an archive. This is functionally equivalent to using the \texttt{iconv} utility to convert each file before or after archiving. This option has the format where \textit{keyword is from and value is the code set. codeset can be a code set name known to the system or the numeric coded character set identifier (CCSID). If a code set name is specified, the numeric CCSID associated with that name is used. Note that the command \texttt{iconv -l} lists existing CCSIDs along with their corresponding code set names.

Two common code set names and their values are ISO8859-1 (ASCII) and IBM-1047 (EBCDIC). For example, to convert from ASCII to EBCDIC, use:

```
-o from=ISO8859-1,to=IBM-1047
```

From EBCDIC to ASCII, use:

```
-o from=IBM-1047,to=ISO8859-1
```
For a complete list of code sets, refer to z/OS XL C/C++ Programming Guide.

You can omit either the to or from keyword.

- When writing an archive, if you specify from, but omit to, then pax assumes that you want to write a portable archive and will convert the data to ISO/IEC 8859-1. If you specify to, but omit from, then pax will convert the data from IBM-1047.
- When reading an archive, if you specify from, but omit to, then pax will convert the data to IBM-1047. If you specify to, but omit from, then pax will convert the data from ISO/IEC 8859-1.

If your input contains a character that is not valid in the source code set, pax displays a warning and continues, leaving the character untranslated. If the source code set contains a character that is not in the destination code set, pax converts the character to an underscore (_).

By default, no code set conversion of file contents is done. When making code set conversions, pax assumes that all files are text files, because only text files are portable.

**fromfiletag**
For use with -o from=,to=.

Use of -o fromfiletag indicates that if a component file has a coded character set assigned to it, then that coded character set is used as the from=codeset, which overrides the value specified on -o from=,to=.

**gid=value**
See “Extended header keywords” on page 525.

**globexthdr.name=string**
(Applicable only to the -x pax format.) When used in write or copy mode with the appropriate options, pax creates global extended header records with USTAR header blocks that will be treated as regular files by previous versions of pax. This keyword allows user control over the name that is written into the USTAR header blocks for global extended header records. The name is the contents of string, after the following character substitutions have been made:

<table>
<thead>
<tr>
<th>string includes:</th>
<th>Replaced by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>%n</td>
<td>An integer that represents the sequence number of the global extended header record in the archive, starting at 1.</td>
</tr>
<tr>
<td>%p</td>
<td>The process ID of the pax process in extract mode.</td>
</tr>
<tr>
<td>%%</td>
<td>A % character.</td>
</tr>
</tbody>
</table>

Any other % characters in string produce the character itself. For instance, %s prints the character ‘s’.

If no -o globexthdr.name=string is specified, pax uses the following default value:

```
$TMPDIR/GlobalHead.%p.%n
```

where $TMPDIR represents the value of the TMPDIR environment variable. In extract mode, if TMPDIR is not set, pax uses /tmp.

**gname=value**
See “Extended header keywords” on page 525.

**invalid=action**
(Applicable only to the -x pax format.) This keyword allows user control over the action pax extracts takes after encountering values in an extended header record that, in read or copy mode, are not valid in the destination hierarchy or, in list mode, cannot be written in the code set and current locale. The following are values for the invalid keyword that are recognized by pax extracts

- **Replace**
  - Replace with the next code set.

- **Ignore**
  - Ignore the character.

- **Map**
  - Map to another character.

- **Replace**
  - Replace with the next code set.

- **Ignore**
  - Ignore the character.

- **Map**
  - Map to another character.
- In read or copy mode, a file name or link name that contains character encodings that are not valid in the destination hierarchy. For example, the name might contain embedded NULL characters.
- In read or copy mode, a file name or link name that is longer than the maximum allowed in the destination hierarchy for either a path name component or the entire path name.
- In list mode, any character string value (file name, link name, user name, and so on) that cannot be written in the code set and current locale.

**bypass**
In read or copy mode, pax extract will bypass the file, causing no change to the destination hierarchy. In list mode, pax writes all requested valid values for the file, but will not write invalid values.

**rename**
In read or copy mode, pax extracts acts as if the -i option were in effect for each file with invalid file name or link name values, allowing the user to provide a replacement name interactively. In list mode, pax extracts behaves identically to the bypass action.

**UTF-8**
When used in read, copy, or list mode and a file name, link name, owner name, or any other field in an extended header record cannot be translated from the pax UTF-8 code set format to the code set and current locale, pax extracts uses the actual UTF-8 encoding for the name.

**write**
In read or copy mode, pax extracts writes the file, translating the name, regardless of whether this may overwrite an existing file with a valid name. In list mode, pax extracts will behave identically to the bypass action.

If no -o invalid= option is specified, pax extracts acts as if -o invalid= bypass were specified. Any overwriting of existing files that might be allowed by the -o invalid= actions is subject to permission (-p) and modification time (-u) restrictions, and is suppressed if the -k option is also specified.

**linkdata**
(Applicable only to the pax extracts format.) In write mode, pax extracts writes the contents of a file to the archive even when that file is merely a hard link to a file whose contents have already been written to the archive.

**linkpath=value**
See “Extended header keywords” on page 525.

**listopt=format**
This keyword specifies the output format of the table of contents that is produced when the v option is specified in list mode. To avoid ambiguity, listopt= format must be only or final keyword in the -o options argument. All characters in the remainder of the -o options argument are considered part of the format string. When multiple -o listopt=format options are specified, the format strings are considered a single, concatenated string, evaluated in command line order.

To ensure proper data display, use the proper conversion specifier character for the field that is being displayed for numeric data. For example, the size field on z/OS systems is often a long data type. Attempting to display the size field using a conversion specifier for a smaller data type, for example %d, will result in a zero being displayed instead of the contents of the size field. You can use the lld conversion specifier character for the long data type. Note that lld is lowercase LL.

**mtime=value**
See “Extended header keywords” on page 525.

**noext**
See -o saveext | noext.

**path=value**
See “Extended header keywords” on page 525.

**realtime.any=value**
See “Extended header keywords” on page 525.
security.any=value
See “Extended header keywords” on page 525.

saveext | noext
For USTAR and OS390-formatted archives, controls whether extended USTAR support is enabled (saveext) or disabled (noext). noext is the default behavior for USTAR format when writing an archive.

The saveext option is the default behavior for OS390 format when writing an archive. It is the default behavior when extracting or listing files from the archive. It is also the default to save extended attributes and external links. To list attributes such as ACLs or file tags, -o A and -o T option must be used. This option has no effect for non-USTAR format. For more information about extended USTAR support, see “z/OS-extended USTAR support” on page 532.

Table 26: Defaults for USTAR

<table>
<thead>
<tr>
<th>Action</th>
<th>USTAR default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writing, copying</td>
<td>-noext</td>
</tr>
<tr>
<td>Extracting, listing</td>
<td>-saveext</td>
</tr>
</tbody>
</table>

saveext
During archive writing, saveext causes pax extracts to preserve extended USTAR information. During archive listing, saveext causes pax extracts to display extended USTAR information and long pathname or long linkname in GNU tar format. During archive reading, saveext enables pax to restore extended USTAR information information and long pathname or long linkname in GNU tar format. To restore certain information, the user must also have the appropriate privileges and have specified the corresponding options. For example, in order to restore extended attributes, pax must be specified and to restore ACLs -pA must be specified. The external links and extended attributes are saved by default for USTAR and OS390 format. The file attributes requiring special headers, such as long links, file tags, and ACLs, need the -o saveext to be specified for USTAR (OS390 uses -o saveext by default). The environment variable _OS390_USTAR=Y can also be used to turn on the support. For more information about extended USTAR support, see “z/OS-extended USTAR support” on page 532.

noext
When creating archives, does not preserve extended USTAR information. When reading or listing an archive, ignore any extended USTAR support (such as extended attributes, long links, external links, file tags, and ACLs) encoded within the archive and the long pathname or long linkname in GNU tar archives. If an archive contains z/OS special header files, these are displayed or restored (or both) as regular files. Special header files are described in “z/OS-extended USTAR support” on page 532.

Restriction: The pax extracts (-x pax) format does not recognize the noext option.

setfiletag
For use with -o from=, to=.

Using -o setfiletag tags component files that have not already been tagged.

If a file is untagged (TXTFLAG = OFF, CCSID = 0), then it is automatically stored with TXTFLAG = ON and with CCSID = to the target code set.

For files that are already not untagged, -o setfiletag does not change the default behavior. The target code set and TXTFLAG values remain as is. For example, a file that is tagged as mixed will have TXTFLAG = OFF and CCSID <> . UNIX will not automatically force TXTFLAG = ON because it does not want to override the user’s reason for making the file mixed.

size=value
See “Extended header keywords” on page 525.
times
(Applicable only to the -x pax format.) When used in write or copy mode, pax extracts include atime and mtime extended header records for each file.

T
Displays file tag information. Similar to ls -T and chtag output. Does not automatically turn on verbose (-v) in the same way that ls -T does not automatically turn on its -L (long listing) option. When used without -v, only the file tag information and file names are displayed.

For example, when used without -v:

```
/tmp> pax -o T -f asciitagged.pax
m ISO8859-1 T=off text_am
t ISO8859-1 T=on  text_at
- untagged  T=off text_au
```

This option can be used with -v or -o E to display additional verbose output.

For example, to display additional verbose output:

```
/tmp> pax -o T -vf asciitagged.pax
m ISO8859-1 T=off -rw-r--r-- 1 SteveS  Kings 9 Apr 30 22:31 text_am
nt ISO8859-1 T=on  -rw r--r-- 1 SteveS  Kings 9 Apr 30 22:31 text_at
- untagged  T=off -rw-r--r-- 1 SteveS  Kings 9 Apr 30 22:06 text_au
```

to=codeset
to=codeset is typically used with from=codeset.

Converts data to another code set while reading or writing an archive. This is functionally equivalent to using the iconv utility to convert each file before or after archiving. This option has the format where keyword is to and value is a code set. codeset can be a code set name known to the system or the numeric coded character set identifier (CCSID). If a code set name is specified, the numeric CCSID associated with that name is used. Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names.

Two common code set names and their values are ISO8859-1 (ASCII) and IBM-1047 (EBCDIC). For example, to convert from ASCII to EBCDIC, use:

```
-ofto=ISO8859-1,to=IBM-1047
```

From EBCDIC to ASCII, use:

```
-ofto=IBM-1047,to=ISO8859-1
```

For a complete list of code sets, refer to z/OS XL C/C++ Programming Guide.

You can omit either the to or from keyword.

• When writing an archive, if you specify from, but omit to, then pax assumes that you want to write a portable archive and will convert the data to ISO/IEC 8859-1. If you specify to, but omit from, then pax will convert the data from IBM-1047.
• When reading an archive, if you specify from, but omit to, then pax will convert the data to IBM-1047. If you specify to, but omit from, then pax will convert the data from ISO/IEC 8859-1.

If your input contains a character that is not valid in the source code set, pax displays a warning and continues, leaving the character untranslated. If the source code set contains a character that is not in the destination code set, pax converts the character to an underscore (_). By default, no code set conversion of file contents is done. pax assumes that all files are text files, since only text files are portable.

uid=value
See “Extended header keywords” on page 525.
uname=value
   See “Extended header keywords” on page 525.

ZOS.any=value
   See “Extended header keywords” on page 525.

-p string
Specifies which file characteristics to restore. By default, pax only restores the access time (if it is stored in the archive) and modification time of each component file, and the access permissions (mode) as modified by the current umask. That is, they are restored entirely when the umask is 000. Currently only pax format archives are capable of storing the access time. Other archive formats use the modification time as the access time.

To store the access time in a pax format archive, you must specify -o times when the archive is created or you can manually specify a value for a common access time for all the files in the archive with the -o option used with the atime keyword on archive creation or extraction. The file tag information, external links, and links whose target exceed 100 characters are also restored by default. Only file attributes that are available in the archive that is being read can be restored. See the -x option, the -o saveext|noext option, and the file format descriptions in Appendix H, “File formats,” on page 969 to understand the limitations of the archive formats.

string can consist of any combination of the following characters:

A
   Restores ACL data.

a
   Does not preserve file access times.

e
   Preserves the user ID, group ID, file mode, access time, modification time, extended attributes, and ACL entries. Prior to z/OS 1.8, audit flags and file format (line end) attributes were not restored because they are not available in any archive format. The extended attributes are the apsl flags that are set by the extattr command. A pax format archive can be used to store the audit flags and file format, and -p e will restore them when available.

m
   Does not preserve file modification times.

o
   Preserves the user ID and group ID.

p
   Preserves the file mode: access permissions (without modification by umask), set-user-ID bit, set-group-ID bit, and sticky bit.

   pax restores access permissions by default. If _UNIX03=YES extracted files will have permissions of 0666 (modified by umask) unless -p p or -p e are used.

W
   Preserves user-requested audit attributes and auditor-requested audit attributes and the file format. The invoking user ID must have the AUDITOR attribute set in the system security product to successfully set auditor-requested audit attributes.

x
   Preserves extended attributes. The extended attributes are the apsl flags that are set by the extattr command.

   If neither the e nor the o specification character is specified, or the user ID and group ID are not preserved for any reason, pax shall not set the set-user-ID and set-group-ID bits of the file mode.

-q
   For read mode only, pax assumes that all created files are text files and extracts them to the local text file format. On systems with fixed length records, this might mean appending blanks as padding.
   On UNIX and POSIX-compliant systems, pax removes all carriage return characters (\r) and retains only the newline (\n) characters.
-r
Reads an archive file from standard input.

-s substitute
Modifies path name using a substitution command substitute. This is similar to the substitution command of the ed text editor. The full option has the form:

```
-s#bregexp#string#[gp]
```
where bregexp is a basic regular expression and string is a string that pax is to insert in place of matches for the regular expression. string can contain an ampersand & (standing for the string matching bregexp), or \1, \2, and so on (with the meanings defined in regexp), for subexpression matching.

The # is used as the delimiter character separating bregexp and string. You can use any non-null character instead. There cannot be any space between -s and the delimiter character.

Normally, -s replaces only the first match for bregexp. A g following the string replaces all matches in the line.

A p following the string prints all successful substitutions on the standard error stream. pax displays a substitution in the format:

```
oldname >> newname
```
There might be more than one -s option on the command line. In this case, pax tries the substitutions in the order given. pax stops trying to make these substitutions as soon as it makes its first successful substitution. If the null string replaces a file name, pax ignores that file name on both input and output.

-T
After pax reads files that are being archived, it resets the access time to that prior to pax’s access.

-u
Compares component dates to dates of existing files with the same name. When pax extracts components with -r (read mode), it extracts a file only if its modification date is more recent than the modification date on an existing file of the same name. In other words, it doesn't overwrite an existing file if the existing file is newer than the one in the archive.

Similarly, when copying files with -rw (copy mode), pax does not overwrite an existing file if the existing file is newer than the one being copied.

In a command that uses -w but not -r (write mode), -u checks to see if the file that is being added has the same name as a file already in the archive. If so, and if the file being added is newer than the one in the archive, pax leaves the old file in the archive and appends the new one at the end. In this case, -u automatically implies -a, which means that pax adds new files to the end of the archive.

-v volpat
Provides automatic multivolume support. pax writes output to files the names of which are formatted with volpat. It replaces any occurrence of # in volpat with the current volume number. When you invoke pax with this option, it asks for the first number in the archive set, and waits for you to type the number and a carriage return before proceeding with the operation. pax issues the same sort of message when a write error or read error occurs on the archive; the reasoning is that this kind of error means that pax has reached the end of the volume and is to go on to a new one. An interrupt at this point ends pax.

-v
Lists path name on the standard error stream just before beginning to process the files or directories, but after any -i, or -s options have had their effect. In list mode (neither -r nor -w is specified), pax displays a verbose table of contents; this verbose format shows information about the components in the same format that is used by the ls command. See “Output” on page 523 for more information.

-w seqparms=parms
Specifies the parameters that are needed to create a sequential data set if one does not exist. You can specify the RECFM, LRECL, BLKSIZE, and SPACE in the format that fopen() function uses.
SPACE=(units, (primary, secondary)) where the following values are supported for units:

- Any positive integer that indicates BLKSIZE
- CYL (mixed case)
- TRK (mixed case)

Space can be specified as follows:

- SPACE=(500, (100, 500)) units, primary, secondary
- SPACE=(500, 100) units and primary only

The fopen() arguments: LRECL specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records. BLKSIZE specifies the maximum length, in bytes, of a physical block of records. RECFM refers to the record format of a data set and SPACE indicates the space attributes for MVS data sets. For example:

```
pax -W "seqparms='RECFM=U, space=(500,100)"" -wf "//target.dataset" source
```

For information about specifying these parameters, see *z/OS XL C/C++ Programming Guide*.

- **-w**
  Writes files to standard output in the specified archive format.

- **-x**
  Writes out only those files that are on the same device as their parent directory. However, it will not copy a directory that is used as a mount point. The user must either unmount the file system from that mount point or copy the directory manually.

- **-x format**
  Specifies a file format for an output archive. The *format* argument can be:

  - **cpio**
    The ASCII format that is used by the *cpio* command.

  - **cpio8**
    The binary format that is used by *cpio*.

  - **os390**
    The OS390 format, which has all the support for saving and restoring extended USTAR support such as special headers, external links, and long links. This format is only supported on z/OS systems.

  - **pax**
    The *pax* interchange format which, like *os390* format (-x os390) and extended USTAR (-o saveext), saves or restores file attributes that cannot be stored in the USTAR header format such as ACLs, external links, long link names, long path names, file tags, and extended attributes.

    Additionally, the *pax* interchange format can save and restore file attributes that cannot be handled by any other format such as files greater than 8 GB, uid and gid values greater than 2097151, large time values (see the usage notes), and z/OS-specific attributes like user audit and auditor audit flags and file format. To restore certain information, the user must also have the appropriate privileges and have specified the corresponding options. See -p for options for restoring file attributes.

    In copy mode, *pax* behaves as if it were using the *pax* interchange format.

  - **tar**
    The historical format of *tar* files.

  - **ustar**
    The USTAR format that is used by the *tar* command. It is the default for *format*.

    To preserve information about extended attributes, external links, and link names greater than 100 characters, *ustar* with either the _OS390_ USTAR=Y environment variable or the -o saveext option can be used. You can also use the -x os390 or -x pax option to store these attributes.
**Tip:** If you are creating archives that might be extracted on previous z/OS systems, use the USTAR, extended USTAR (-o saveext) or os390 format. If the archives will only be extracted on z/OS release 8 systems and later, the `pax` format (-x pax) is the recommended format.

-z
For write or read mode, performs Lempel-Ziv compression. -z cannot be used when appending (-a) to an existing archive.

**Tip:** If you are creating archive files by using the -f option, suffix the archive name with a .Z to identify it as a compressed file and to facilitate it being processed by uncompress (if needed).

For reads, -z is functionally equivalent to first uncompressing the archive using the uncompress utility and then reading it. This option is not required when reading a compressed archive. *pax* will automatically detect that the archive is compressed. It might be useful, however, to use -z to confirm that the archive is compressed; you will receive an error message if you specify -z on an archive that is not compressed.

**Output**
When the -v or -E option is used in list mode, *pax* produces a verbose table of contents for the archive. The output for -v is similar to the output from the `ls -L` command with the following exceptions:

- The notation:

  ```
  pathname == linkname
  ```

  indicates that `linkname` is a hard link of `pathname`.
- For symbolic and external links, *pax* output always shows a file size of 0.

The output from the -E option has the same format as -v and additionally displays a column that shows the extended attributes:

- `a`
  The program runs APF-authorized if linked AC=1.
- `p`
  The program is considered program-controlled.
- `s`
  The program runs in a shared address space.
- `l`
  The program is loaded from the shared library region. (l is a lowercase L, not an uppercase i.)
- `-`
  The attribute is not set.

The format of the `pax -E` output is variable in length and is extended as necessary to display additional file characteristics that are not supported by *pax* -v (ls -L).

**Usage notes**
1. On the z/OS UNIX system, superuser privileges or read access to the appropriate FACILITY class resources are required to create character special files, restore user and group names, and to set certain extended attributes (read access to the corresponding FACILITY class resources).
2. The POSIX 1003.1 standard defines formats for `pax`, `tar`, and `cpio` archives that limit the UIDs and GIDs that can be stored to the maximum values as shown in "pax - Interchange portable archives" on page 510.
Table 27: Maximum UID and GID values for tar, USTAR, cpio, and pax

<table>
<thead>
<tr>
<th>Format</th>
<th>Maximum UID and GID values</th>
</tr>
</thead>
<tbody>
<tr>
<td>tar, USTAR</td>
<td>2,097,151</td>
</tr>
<tr>
<td>cpio</td>
<td>262,143</td>
</tr>
<tr>
<td>pax</td>
<td>2,147,483,647</td>
</tr>
</tbody>
</table>

Larger values are not properly restored for **tar** and **cpio** formatted archives. For USTAR formatted archives, because the user and group names are also stored in the archive, the correct UID and GID are restored only if the name is defined on the target system.

3. In historical UNIX standard formats, for **pax** and **tar** archives, the length of a link file target is limited to 100 characters or less. If hard links exist, the target is the first occurrence of the hard link that is saved in the archive. Subsequent hard links refer to the first instance. You can use the extended USTAR support that is provided by **pax** and **tar** to save the long hard links when the archives are created, and to restore them when the archives are read. As of z/OS V1R8, the **pax** format as defined by the current UNIX standard provides a means to save and restore long link names that can be transferred between other (including non-z/OS) UNIX operating systems. For more information, see `-x pax`.

4. In historical UNIX standard formats, the size of a file that can be stored in a **pax** and **tar** archive was limited to 8 gigabytes. As of z/OS V1R8, the **pax** format archive allows files greater than 8 gigabytes to be archived. For more information, see `-x pax`.

5. If you are transferring archives between z/OS systems and other UNIX systems, consider the following notes:
   a. Archive files must always be transferred in binary mode, even if the archives contain only text files. Common ways of transferring files include FTP, the cp shell command, and the OPUT and OGET TSO/E commands.
   b. You might need to convert text files from EBCDIC to ASCII (or some other character set). The pax `-o` option can be used to convert text files while an archive is being created or being restored. You can use the `iconv` utility to convert files before they are stored in the archive or after you restore them from an archive.

6. Automatic conversion on files with file tag information is disabled when files are read during the creation of an archive. It is also disabled during writes when files are extracted from archives. That is, the settings of system and environment variables that turn automatic conversion on and off have no effect on the reading and writing of files by **pax**.

7. The POSIX 1003.1 standard defines formats for cpio, cpio, tar, and USTAR archives that limit the modification time to a maximum value of `FFFFFFFE` (hexadecimal) for the cpio format and `7777777777` (octal) for all other formats. Modification times that are larger than the maximum value are not properly saved. Modification times beyond 03:14:07 UTC on January 19, 2038 but less than or equal to the maximum value, are not properly saved or restored on z/OS V2R2 and earlier systems. They also might not be properly saved or restored on non-z/OS systems. However, on systems later than z/OS V2R2, such values are properly saved and restored. Due to these restrictions, the pax interchange format is the preferred format for processing time values beyond 03:14:07 UTC on January 19, 2038.

**Note:**
- **pax** does not support times beyond 03:14:07 UTC on January 19, 2038 on z/OS V2R2 and earlier systems regardless of the archive format used.
- The OS390 format has the same maximum value as the USTAR format.

**File tagging**

When the `-o from=,to=` option is used to perform conversion, the default behavior for storing the file tag information is as follows:
-w (write)
For files that are tagged, the CCSID preserved in the archive is set to the CCSID of the to=codeset argument. Files that are untagged (TXTFLAG = OFF and CCSID = 0) will not have file tag information stored. The -o setfiletag option can be used to force the tagging of files that are not already tagged.

When a file in the archive is tagged with a different CCSID than the from=codeset, an error message is generated. However, pax continues processing. Because this situation indicates a probable corruption of data, upon completion, pax issues a nonzero return code. To avoid this situation, the -o fromfiletag option can be used. It causes pax to use the CCSID of the file instead of the one specified on the -o from=,to= option.

-r (read)
For files that are tagged, the TXTFLAG value is restored to the value preserved in the archive (ON or OFF), but the CCSID of the target file is altered to the to=codeset CCSID. For example, a file that is tagged as mixed will have TXTFLAG = OFF and CCSID ≠ 0. z/OS UNIX will not automatically force TXTFLAG = ON because it does not want to override the user's reason for making the file mixed.

The default behavior for files in the archive that are untagged will not change, and the target file will also be set to untagged. The -o setfiletag option can be used to force the tagging of files that do not have filetag information associated with them in the archive.

If the target file exists, its filetag information is ignored.

When a file in the archive is tagged with a different CCSID than the from=codeset, an error message is generated. However, pax continues processing. Because this situation indicates a probable corruption of data, upon completion, pax issues a nonzero return code. The -o fromfiletag option can be used to avoid this situation. It causes pax to use the CCSID of the file rather than the one specified on the -o from=,to= option.

-w (copy)
If the source files are tagged, then the target file will have its CCSID set to the CCSID of the to=codeset CCSID. If the target exists, then its TXTFLAG values are ignored. The source file is used to determine the TXTFLAG setting of the target and overrides whatever the TXTFLAG settings are of the target.

Like -r and -w, when the CCSID of the source file is different from the from=codeset CCSID, a warning message is generated and, upon completion, pax issues a nonzero return code. The -o fromfiletag option can be used to avoid this situation. It causes pax to use the CCSID of the file rather than the one specified on the -o from=,to= option.

Extended header keywords
The following extended header keywords are applicable only in the -x pax format.

atime
The file access time for the following files, equivalent to the value of the st_atime member of the stat64 structure for a file. The access time shall be restored if the process has the appropriate privilege required to do so.

charset
The name of the character set used to encode the data in the following files. The entries in Table 28 on page 526 are defined to refer to known standards and the charset value used to represent each:

The encoding is included in an extended header for information only; when pax is used as described, it does not convert the file data into any other encoding. The BINARY entry indicates unencoded binary data.
<table>
<thead>
<tr>
<th>Value</th>
<th>Formal standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO-IR 10646 2000 UTF-8</td>
<td>ISO/IEC 10646, UTF-8 encoding</td>
</tr>
<tr>
<td>BINARY</td>
<td>None</td>
</tr>
</tbody>
</table>

**comment**

A series of characters that are used as a comment. All characters in the value field are ignored by pax.

**gid**

The group ID of the group that owns the file, expressed as a decimal number using digits from ISO/IEC 646. This record overrides the gid field in the following header blocks.

When used in write or copy mode, pax includes a GID extended header record for each file whose group ID is greater than 2097151 (octal 7777777).

**gname**

The group of the following files, formatted as a group name in the group database. This record overrides the gid and gname fields in the following header blocks, and any gid extended header record.

In read, copy, or list mode, pax translates the name from the UTF-8 encoding in the header record to the character set appropriate for the group database on the receiving system. If any of the UTF-8 characters cannot be translated, and if the `-o invalid=UTF-8` option is not specified, the results are undefined as if `-o invalid=bypass` were specified.

In write or copy mode, pax includes a gname extended header record for each file whose group name cannot be represented entirely with the letters and digits of the portable character set.

**linkpath**

The path name of a link that is being created to another file, of any type, previously archived. This record overrides the linkname field in the following USTAR header blocks.

The following USTAR header block determines the type of link created, whether hard or symbolic. In the latter case, the linkpath value is the contents of the symbolic link. pax converts the name of the
link (contents of the symbolic link) from the UTF-8 encoding to the character set appropriate for the local file system.

In write or copy mode, **pax** includes a linkpath extended header record for each link whose path name cannot be represented entirely with the members of the portable character set other than NULL.

**mtime**
The file modification time of the following files, equivalent to the value of the st_mtime member of the stat64 structure for a file. This record overrides the mtime field in the following header blocks. The modification time is restored if the process has the appropriate privilege to do so. If **pax** is used in write or copy mode, it automatically includes an mtime extended header record for each file with a mtime value greater than 8589934591 (octal 77777777777).

**path**
The path name of the following files. This record overrides the name and prefix fields in the following header blocks. **pax** converts the path name of the file from the UTF-8 encoding to the character set appropriate for the local file system.

In write or copy mode, **pax** includes a path extended header record for each file whose path name cannot be represented entirely with the members of the portable character set other than NULL.

**realtime.any**
The keywords that are prefixed by **realtime**. are reserved for future POSIX realtime standardization. **pax** recognizes but silently ignores them.

**security.any**
The keywords that are prefixed by **security**. are reserved for future POSIX security standardization. **pax** recognizes but silently ignores them.

**size**
The size of the file in octets, expressed as a decimal number using digits from ISO/IEC 646. This record overrides the size field in the following header blocks.

If **pax** is used in write or copy mode, it automatically includes a size of extended header record for each file with a size value greater than 8589934591 (octal 77777777777). As with other keywords, the user can manually set this value by using `-o size=value` or `-o size=value`. However, it is strongly recommended this not be done. Creating a global or extended size record for the size extended record keyword can cause failures or data corruption when used in read or write mode. size extended records are ignored by **pax** in copy mode.

**uid**
The user ID of the user that owns the file, expressed as a decimal number using digits from ISO/IEC 646. This record overrides the uid field in the following header blocks.

When used in write or copy mode, **pax** includes a uid extended header record for each file whose owner ID is greater than 2097151 (octal 7777777).

**uname**
The owner of the following files, formatted as a user name in the user database. This record overrides the uid and uname fields in the following header blocks, and any uid extended header record.

In read, copy, or list mode, **pax** converts the name from the UTF-8 encoding in the header record to the character set appropriate for the user database on the receiving system. If any of the UTF-8 characters cannot be converted, and if the `-o invalid=UTF-8` option is not specified, the results are as if `-o invalid=bypass` were specified.

In write or copy mode, **pax** includes a uname extended header record for each file whose user name cannot be represented entirely with the letters and digits of the portable character set.

**ZOS.acls**
The extended access control lists (extended ACLs) of the following files.
When used in write or copy mode, **pax** includes a ZOS.acls record for each file that has extended ACLs set. Values of the ZOS.acls keyword have the following format:

```plaintext
[d[efault]: | f[default]:]u[ser]:uid:perm
[d[efault]: | f[default]:]g[roup]:gid:perm
```

where:

**d[efault]**
- If specified, extended ACL refers to directory default ACL.

**f[efault]**
- If specified, extended ACL refers to file default ACL.

**u[ser]**
- Extended ACL refers to a particular numeric user ID (UID) or user name.

**g[roup]**
- Extended ACL refers to a particular numeric group ID (GID) or group name.

**uid**
- User name or numeric user ID (UID).

**gid**
- Group name, or numeric group ID (GID).

**perm**
- Permissions that are specified either in absolute form (string rwx with - as a placeholder or octal form.

Syntax examples:

```plaintext
-o ZOS.acls=user:billy:r-x
-o ZOS.acls=g:cartoons:r
```

In the next example, the multiple entries in the value are comma-separated. However, because these literal commas are in a -o value, they must be preceded by a backslash since commas are used to delimit keyword-value pairs regardless of whether the value is enclosed in quotation marks.

```plaintext
-o ZOS.acls=user:user1:r-x,group:thegang:r--
\,d:user:user1:r-x\,d:group:thegang:r--\,d:user:user2:r-x
```

**ZOS.taginfo**

The value for the ZOS.taginfo keyword is composed of a text flag (txtflag) and a coded character set and allows the user to modify the taginfo that is associated with the file.

The txtflag indicates whether a file contains uniformly encoded or non-uniformly encoded text data. Values for txtflag are 0 (indicating txtflag is OFF) or 1 (indicating txtflag is ON). If the txtflag is 1 (ON), it indicates that the specified file contains pure (uniformly encoded) text data. For files that contain binary, mixed, or unknown data, the txtflag is 0 (OFF).

The coded character set represents the code set in which text data is encoded. The code set can be used for uniformly encoded text files or files that contain mixed text/binary data. It can be a code set name known to the system or the numeric coded character set identifier (CCSID). If a code set name exists, the numeric CCSID associated with that name is used.

When used in write or copy mode, **pax** includes a ZOS.taginfo extended header record for each file for which txtflag is 1 (ON) or the file is not untagged.

The command `iconv -l` lists existing CCSIDs along with their corresponding code set names. Values of the ZOS.taginfo keyword have the following format:

```plaintext
0 [ccsid]
1 ccsid
```

Syntax examples:
Indicates the user-requested audit attributes of the specified files or directories. Audit attributes determine whether accesses to a file are audited by the System Authorization Facility (SAF) interface.

When used in write or copy mode, pax includes a ZOS.useraudit record for each file which the user-requested audit attributes are anything other than auditing read, write, and execute failures on the file.

The value of the ZOS.useraudit keyword is a sequence of three characters, each of which can be one of the following four characters. The character in the first position represents the audit properties for read operations on the corresponding file, the second represents audit properties for write operations on the corresponding file, and the third character represents audit properties for execute operations on the corresponding file.

- Do not audit.
- Audit failures.
- Audit successes.
- Audit both successes and failures.

Syntax examples:

```bash
-o ZOS.useraudit=ffa
-o ZOS.useraudit=ssa
-o ZOS.useraudit=sf-
```

Indicates the auditor-requested audit attributes of the specified files or directories. Audit attributes determine whether accesses to a file are audited by the System Authorization Facility (SAF) interface.

When used in write or copy mode, pax includes a ZOS.auditoraudit record for each file which the auditor-requested audit attributes are set on the file.

The value of the ZOS.auditoraudit keyword is a sequence of three characters, each of which can be one of the following four characters. The character in the first position represents the audit properties for read operations on the corresponding file, the second represents audit properties for write operations on the corresponding file, and the third character represents audit properties for execute operations on the corresponding file.

- Do not audit.
- Audit failures.
- Audit successes.
- Audit both successes and failures.

Syntax examples:

```bash
-o ZOS.auditoraudit=ffa
-o ZOS.auditoraudit=ssa
-o ZOS.auditoraudit=sf-
```
**ZOS.filefmt**

Specifies whether a file is binary or text and for text files, specifies the end-of-line delimiter. For format you can specify:

- **not**
  - Not specified.
- **bin**
  - Binary data.
- **rec**
  - Record. (File data consists of records with prefixes. The record prefix contains the length of the record that follows. From the shell command perspective, files with this format are treated as if they were binary files.)

Or the following text data delimiters:

- **nl**
  - Newline character.
- **cr**
  - Carriage return.
- **lf**
  - Line feed.
- **crlf**
  - Carriage return followed by line feed.
- **lfcr**
  - Line feed followed by carriage return.
- **crnl**
  - Carriage return followed by a newline character.

**Restriction:** The rec value for ZOS.filefmt is only available on z/OS V1R12 and later. Therefore, if an object with the rec file format is restored on an earlier release, the rec file format information will be lost.

**ZOS.extattr**

The value of this keyword is a 4-character string that specifies the extended attributes for files to allow executable files to be marked so they run APF-authorized, as a program controlled executable, or not in a shared address space.

- The first character of the value specifies whether the program runs APF-authorized and is either 'a' or '-'.
- The second character of the value specifies whether the program is considered program controlled and is either 'p' or '-'.
- The third character of the value specifies whether the program runs in a shared address space and is either 's' or '-'.
- The fourth character of the value specifies whether the program file is loaded from the shared library region and is either 'l' or '-'.

- **a**
  - The program runs APF-authorized if linked AC=1.
- **p**
  - The program is considered program controlled.
- **s**
  - The program runs in a shared address space.
- **l**
  - The program file is loaded from the shared library region.
- **-**
  - The attribute is not set.
## Extended header keyword precedence

(Applicable only to the `-x pax` format.)

This topic describes the precedence in which the various header records and fields and command-line options are selected to apply to a file in the archive. When **pax** is used in read or list modes, it determines a file attribute in this sequence:

1. If `-o delete=keyword-prefix` is used, the affected attribute is determined from step “7” on page 531 if applicable, or ignored otherwise.

2. If `-o keyword:=` is used with no value specified, the affected attribute is ignored.

3. If `-o keyword:=value` is used, the affected attribute is assigned the value.

4. If a `keyword` exists in a file-specific extended header record, the affected attribute is assigned the value. When extended header records conflict, the last one given in the header takes precedence.

5. If `-o keyword=value` is used, the affected attribute is assigned the value.

6. If a `keyword` exists in a global extended header record, the affected attribute is assigned the value. When global extended header records conflict, the last one given in the global header takes precedence.

7. Otherwise, the attribute is determined from the USTAR header block.

### Specifications for list mode format

In list mode with the `-o listopt=format` option, the format argument is applied for each selected file. The **pax** utility appends a newline character to the listopt output for each selected file. The format argument is used as the format string with the following exceptions:

1. A `<space>` character in the format string, in any context other than a flag of a conversion specification, is treated as an ordinary character that is copied to the output.

2. In addition to the escape sequences `\|\|a, \|b, \|f, \|n, \|r, \|t, and \|v, the escape sequence `\ddd`, where `ddd` is a one-, two-, or three-digit octal number, is written as a byte with the numeric value specified by the octal number.

3. Output from the d or u conversion specifiers is not preceded or followed with s not specified by the format operand.

4. Output from the o conversion specifier is not preceded with zeros that are not specified by the format operand.

5. The sequence `(keyword)` can occur before a format conversion specifier. The conversion argument is defined by the value of `keyword`. The following keywords are supported:

   Any of the Field Name entries in USTAR Header Block and Octet-Oriented cpio Archive Entry. The implementation supports the cpio `c_` mode Field.

   Any keyword that is defined for the extended header in “Extended header keywords” on page 525.

   Any keyword that is provided as an implementation-defined extension within the extended header that is defined in “Extended header keywords” on page 525. For example, the sequence `"% (charset)s"` is the string value of the name of the character set in the extended header.
To ensure proper data display, be sure to use the proper conversion specifier character for the field that is being displayed for numeric data. For example, the size field on z/OS systems ID is often a long long data type. Attempting to display the size field using a conversion specifier for a smaller data type, for example %d, will result in a zero being displayed instead of the contents of the size field.

The result of the keyword conversion argument is the value from the applicable header field or extended header, without any trailing NULs. All keyword values used as conversion arguments are translated from the UTF-8 encoding to the character set appropriate for the local file system, user database, and so on, as applicable.

6. An additional conversion specifier character, T, is used to specify time formats. The T conversion specifier character can be preceded by the sequence (keyword=subformat), where subformat is a date format as defined by date operands. The default keyword is mtime and the default subformat is:

```
%b %e %H:%M %Y
```

7. An additional conversion specifier character, M, is used to specify the file mode string as defined in `ls` standard output. If (keyword) is omitted, the mode keyword is used. For example, `%M` writes the single character corresponding to the entry type field of the `ls -l` command.

8. An additional conversion specifier character, D, is used to specify the device for block or special files.
   - If the use of D is applicable, the format is devmajor,devminor.
   - If the use of D is not applicable and (keyword) is specified, then this conversion is equivalent to % (keyword)u.
   - If the use of D is not applicable and (keyword) is omitted, then this conversion is equivalent to <space>.

9. An additional conversion specifier character, F, is used to specify a path name. The F conversion character can be preceded by a sequence of comma-separated keywords:

```
(keyword[,keyword] ... )
```

The values for all the keywords that are non-null are concatenated, each separated by a /. The default is (path) if the keyword path is defined. Otherwise, the default is (prefix,name).

10. An additional conversion specifier character, L, is used to specify a symbolic link expansion. If the current file is a symbolic link, then %L expands to: "%s -> %s", value of keyword, contents of link. Otherwise, the %L conversion specification is the equivalent of %F.

11. If the conversion specifier character s for ZOS.taginfo is specified, pax uses the conversion specified character i for the text flag and character s for coded character set. If the conversion specified character for ZOS.taginfo is not s, pax uses the specified conversion specified character for the text flag and s for coded character set.

**z/OS-extended USTAR support**

OS390 archive format stores all the extended USTAR attributes by default (the -o options do not apply). By default, the z/OS implementation of `pax` and `tar` provide extended support with the USTAR format to preserve and restore z/OS-specific file attributes and other information not otherwise supported due to limitations with the USTAR format. The OS390 format also stores these by default. Examples of these include:

- External links
- Extended file attributes (such as program-controlled and APF-authorized). The extended attributes are the apsl flags that are set by the `extattr` command. Audit flags and file format attributes are not stored.

This support is only provided for archives that use the USTAR format. USTAR is the default format for pax when archives are created. For `tar`, the default format is the original `tar` format. The -U option, however, can be used to cause `tar` to use USTAR. When reading an archive, `tar` will automatically recognize the
USTAR format; no special option is required. (For more information about the USTAR format, see the section on tar archives in Appendix H, “File formats,” on page 969.)

The `pax` and `tar` commands also allow the storing and restoring of additional file attributes using explicit options and environment variable. The following attributes require special header support. OS390 format stores and restores these by default. Examples of these additional attributes include:

- Links whose targets exceed 100 characters.
- Access control lists (ACLs).
- File tag information.
- Files with names longer than 255 characters.

The extended USTAR support is provided by using two mechanisms: encoding the information within the USTAR header record and through the creation of special header files. (The same mechanism is used for the OS390 archive format.)

**Encoding information within the USTAR header record**

External link and extended attribute information is encoded within the standard USTAR header in a manner that is compliant with POSIX standards and should be tolerated by other non-z/OS versions of `pax` and `tar`. Because external links and extended attributes are specific to z/OS, however, they cannot be restored on other platforms.

**Special header files**

Hard links and symbolic links with targets greater than 100 characters cannot be preserved within the standard USTAR format (for a hard link, the target is the first occurrence of the hard link that is archived; subsequent hard links refer to the first instance). In order to preserve links with targets greater than 100 characters, special header files are created for each link and stored in the archive. The special headers are stored when one of the following is used: `-o saveext` option, environment variable `_OS390_USTAR=Y`, or `-x os390` option (OS390 format).

Each special header file contains information that is used by z/OS `pax` and `tar` to restore the link to its original state. Special header files are identified in the archive with type "S" (see the information about the format of tar archives in Appendix H, “File formats,” on page 969 for more information about file types).

Each special header file in the archive has the same name: `/tmp/OS390_USTAR_SUMMARY_timestamp` where `timestamp` is the creation time (represented in seconds since the epoch) of the first special header file. For example:

```
/tmp/OS390_USTAR_SUMMARY_919026474
```

When a special header file is required to preserve a file and the `_OS390_USTAR=Y` environment variable was used, an informative message along with a reason is displayed indicating that a special header file was created. When `-o saveext` or `-x os390` for `pax` or `-UX` or `-s` for `tar` is used, the informational message is not displayed.

When reading or listing an archive that contains special header files and when using the default extended USTAR support, `pax` and `tar` recognize type "S" files as special header files and display or restore the file described by the special header rather than the actual special header file. So, typically, the presence of special header files is not known to the user.

When the archive is complete, if one or more special header files were created, then a final special header summary file is created and added to the archive. This file is identified in the archive with type "T" and has the same name as the special header files. This file summarizes, via script commands and comments, the contents of all previously archived special header files. Its primary purpose is to provide help restoring files that are included via special header files to those with versions of `pax` or `tar` that do not implement extended USTAR support.
So, to allow users of non-z/OS systems to read the special header summary file, it is encoded in the ASCII ISO8859-1 code set. To view the special header file in EBCDIC code page IBM-1047, first convert the file by using the `iconv` command. For example:

```
iconv -f ISO8859-1 -t IBM-1047 /tmp/OS390_USTAR_SUMMARY_919026474 > summary_in_ebcdic
```

If extended USTAR support is disabled during reading or listing an archive by using the `pax -o noext` or the `tar -0` option, or if the archive is processed by either an earlier version of z/OS `pax` or `tar` that does not implement extended USTAR support or a non-z/OS system version of `pax` or `tar`, then the special header files is not recognized and are processed as unknown type regular files. During extraction, because all files have the same name, each extracted special header file will overlay the previous one with the special header summary file being the final one restored.

For an example of the special header summary file, see “USTAR archive format” on page 973.

### File tags and the use of `-o noext`

Because special headers are required to store file tag information, the storing and restoring of file tag information is disabled if the user specifies the `-o noext` option. The `-o noext` option is the default for writing an archive. To store information in the special headers, the `-o saveext` or `_OS390_USTAR=Y` environment variable must be used. When `-o noext` is used, each file is treated as if it were untagged. That is, if `-o noext` is specified, the stored or extracted file are set to untagged regardless of its previous file tag setting.

- `-o noext` disables all attributes that are stored with special headers, so this option cannot be used to selectively disable the storing or restoring of text flag information. You must use `chtag` to do that.

- `-o noext` does not affect the automatic conversion of files. If you use `pax` to read, write or copy files, automatic conversion is disabled whether `-o noext` is specified or not.

For more information about automatic conversion and file tagging, see [z/OS UNIX System Services Planning](https://www.ibm.com/support资).
Examples for the pax command

For archive listing (table of contents): If file2 and dir1 have extended ACL entries:

```
> pax -vf acldata.pax
-rwx------   1 STIERT   SHUT      294912 Nov  9 09:57 file1
-rwx------+  1 STIERT   SHUT      294912 Nov  9 09:57 file2
drwxr-xr-x+  2 STIERT   SHUT        8192 Mar 20  2000 dir1/
```

Writing (creating) an archive:

1. The following example creates an archive file that is named /tmp/files.pax from all the files in the current working directory. The -v option is used to display each file as it is being added:

   ```
pax -wvf /tmp/files.pax *
   or
   pax -wvf /tmp/files.pax .
   ```

   The difference between these two forms is that in the latter example (using .), names recorded in the archive are preceded by a "/" which will include and preserve the attributes of the current working directory and any hidden files in the current working directory.

2. Either of these commands creates a compressed version of the archive that was created in Example 1:

   ```
pax -wzvf /tmp/files.pax.Z *
   or
   ```

   In some instances, you can obtain a smaller, more compressed output archive by first creating the pax archive uncompressed, and then using the `compress` command on the archive. For example:

   ```
pax -wvf /tmp/files.pax *
   compress /tmp/files.pax
   ```

3. The following example creates an archive /tmp/dironly.pax containing only the files and directory names in the current directories (it does not include the contents of subdirectories):

   ```
pax -wdvf /tmp/dironly.pax. *
   ```

4. This example creates the archive /tmp/cfiles.pax containing all c files in the current directory:

   ```
pax -wvf /tmp/cfiles.pax *.c
   ```

5. This example creates the archive /tmp/allcfiles.pax containing all c files in the current directory and all subdirectories:

   ```
pax -wvf /tmp/allcfiles.pax $(find . -name "*.c")
   ```

6. This example creates the archive /tmp/ascii_src.pax using all .c and .h files in the current directory converted into ASCII:

   ```
pax -wv -o to=ISO8859-1 -f /tmp/ascii_src.pax *.ch
   ```

7. The following example creates the compressed archive /u/smith/oldfiles.pax.Z containing all files on the system that were not accessed within the last 10 days:

   ```
pax -wzvf /u/smith/oldfiles.pax.Z $(find / -type f -atime +10)
   ```
8. The following example creates the archive /tmp/basename.pax containing all files in the directory sub1 stored in the archive with sub1/ removed from each component name. The pound character # is used as the delimiter for the -s option:

```
pax -wv -s#sub1/# -f /tmp/basename.pax sub1/*
```

**Reading an archive:**

1. To print the file format, extended attributes and file tag information:

```
> pax -vf archive.pax -o listopt='%{ZOS.filefmt}s' -o listopt='%{ZOS.extattr}s' \
> -o listopt='%{ZOS.taginfo}s' args.c
nl--s-1 1047args.c
```

2. This example extracts all the components of the archive source.pax. The -v option is used to display each file or directory as it is extracted.

```
pax -rvf source.pax
```

3. To extract all files in source.pax and convert them from ASCII to EBCDIC:

```
pax -ofrom=ISO8859-1,to=IBM-1047 -rf source.pax
```

4. To extract all files in the archive source.pax ending with .h:

```
pax -rf source.pax `pax -f source.pax | grep h$`
```

This example uses command substitution to first read the archive and generate a list of all files in the archive that end with /.

5. This example extracts files into a directory that is different from the directory they are stored in within the archive. Assume the names of all files that are stored in the archive source.pax begin with the root directory (/). To extract them into /newroot/, use the following command:

```
pax -rvf source.pax -s#/#/newroot/#
```

The -v option is used to show the names of the files as they are extracted and is not required.

6. In the following examples, archive acldata.pax contains file1, file2, and dir1. file1 has no ACL data, file2 has an access ACL, and dir1 has a file default ACL, a directory default ACL, and an access ACL. If you only specify option -f, your output is:

```
> pax -f acldata.pax
file1
file2
dir1
```

If you also specify -o A, ACL information is displayed:

```
> pax -o A -f acldata.pax
file1
file2
user:WELLIE2:rw-
group:SYS1:rwx
dir1
```

Finally, if you add the verbose option, -v, you will see the file permission bit settings that are associated with the file:
Specifying `pax -o A` does not automatically turn on the verbose table of contents format. You must also specify `-v` to display the file permission bit settings that are associated with the file. To check whether a file has an ACL when for example, `file2` and `dir1` have ACLs:

```
pax -vf acldata.pax
```

For more information about access control lists, see `z/OS UNIX System Services Planning`. To store a file with an ACL using the OS390 archive format:

```
pax -o os390 -wf acldata.pax fileAcls
```

**Files**

`/tmp/OS390_USTAR_SUMMARY_`

`timestamp` is a z/OS-extended USTAR special header file. For more information, see “z/OS-extended USTAR support” on page 532.

**Environment variables**

`pax` uses the following environment variable:

```_UNIX03```

For more information about the effect of `_UNIX03` on this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

**Localization**

`pax` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_TIME
- LC_SYNTAX
- NLSPATH

For more information, see Appendix F, “Localization,” on page 965.
pax

Exit values for pax

0
  Successful completion.

1
  Failure due to any of the following reasons:
  • Incorrect option.
  • Incorrect command-line arguments.
  • Out of memory.
  • Compression error.
  • Failure on extraction.
  • Failure on creation.

If pax cannot extract a particular file when reading, or cannot find a particular file when writing, or finds a file with an unsupported file type when writing, it generates an error message and continues to process other files but returns a status of 1. If any other error occurs, pax ends immediately without attempting further processing.

If you see the following message after a write operation, it indicates that your directory or device containing the archive file is full. To continue, enter the name of a new directory; to end pax, type <Ctrl-C>.

If you want to go on, type device/filename when ready

If you see that message after a read operation, it means that pax could not find the archive file that you specified, or that it was damaged. In this case, type <Ctrl-C> to end the operation and then restart pax with the correct archive name.

Portability

POSIX.2, X/Open Portability Guide.

The -L, -q, -v, -E, -p x and -z options are extensions of the POSIX standard.

Related information

compress, cpio, ls, tar, uncompress

See Appendix C, “Regular expressions (regexp),” on page 945 for more information about regexp.

See the cpio and pax file formats in Appendix H, “File formats,” on page 969.

pcat - Unpack and display Huffman packed files

Format

pcat file ...

The pcat utility is fully supported for compatibility with older UNIX systems. However, it is recommended that the zcat utility be used instead because it may provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

Description

pcat uncompresses files that were compressed by pack using a Huffman minimal redundancy code. The uncompressed data is sent to the standard output. This is handy for packed text files, but inappropriate for binary files, because the standard output is treated as a text stream. Binary files can be decoded in place by unpack.
The names of compressed input files are expected to end in .z. If a specified input file name does not end in this suffix, pcat automatically adds the .z. For example, if the command line specifies file abc, pcat looks for abc.z.

Localization

pcat uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Exit values

0
  Successful completion

n
  Indicates that n files could not be unpacked properly. For example, if three out of six files could not be unpacked properly, the exit status is 3.

Related information

cat, file, pack, unpack

pg - Display files interactively

Format

pg [-cfnst] [-p prompt] [-s screen] [+line] [+pattern] [file ...]

The pg utility is fully supported for compatibility with older UNIX systems. However, it is recommended that the more utility be used instead because it may provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

Description

pg displays input files or piped output from another command, a screen at a time. If you do not specify any files, the standard input (stdin) is read. Any file named – specifies the stdin.

“Commands” on page 540 lists commands that can be entered at page and file breaks.

Options

-c
  Clears the screen before displaying each new window.

-e
  Eliminates the (EOF) : prompt at the end of each file.

-f
  Does not fold lines. Typically, lines longer than the screen width, as given by the environment variable COLUMNS are folded into multiple lines. This option may be useful for files containing device-specific escape sequences.
Executes interactive commands immediately after receiving the command character. This works for most commands. Typically, you must press <Enter> for interactive commands.

Set the prompt string that appears at the end of each screen of text to string. The default prompt is a colon (:). If string contains the characters %d, pg replaces those characters with the current page number as in [Page %d].

Displays all interactive command prompts in standout mode (most often reverse video) on the screen.

Does not save input in a temporary file. Typically, if any of the inputs is not directly seekable (as is the case for a serial device or pipe), pg reads input and saves it in a temporary file so that it can be reviewed. Because of this, you cannot scan backwards when viewing such input. This option is also recommended when reading a larger amount of data from a stream that cannot be accommodated on disk.

Sets the number of lines displayed in each screen to n lines. If you do not select this option, the number of lines displayed is one less than the number of lines on the screen as given by the environment variable LINES. “Commands” on page 540 discusses the w command.

Starts printing at line n of the first file. The default is to start printing at line 1.

Starts printing at the line containing the first occurrence of the extended regular expression pattern.

See Appendix C, “Regular expressions (regexp),” on page 945 for more information about regexp.

Commands

Depending on the options you specify, pg pauses between windows (screens) of text, at the end of each file and before starting any file after the first. At these pauses, pg prompts you to enter a command. To read the file, type the command ENTER (newline or Return) at each prompt.

An optional sign (+ or –) followed by an optional numeric address can precede the following commands. Addresses work in multiples of screen displays: for example, an address of +2 displays the second next screen. Typically, an unsigned address implies direct addressing (measured from the beginning of the file). A signed address implies relative addressing in the file; a command beginning with a + scans forward and one beginning with a – scans backward from the current position.

You can edit commands interactively with the standard erase and kill characters.

These are the interactive commands:

Prints a summary of the interactive commands.

Exits immediately from pg.

Executes the string command as if it were typed to the default command interpreter (as in ed). Whether or not you specified the –n option, you must end this command with a newline.

Without a specified address, displays the next window of text. With an address, displays the nth next window of text.

Scrolls a half screen of text. The address is measured in half screens and defaults to the next half screen.
[[±]n/l]  With no address, displays the next line of the file. With an address, it displays a screen starting at the addressed line.

$  Displays the last screen of text in the file.

<Ctrl-L>, .  Redisplays the current displayed window of text.

s file  Saves the entire contents of the current file in file. Whether or not you specified the --n option, you must end this command with a newline.

[n]/n  Displays the first screen of the next file. The address (n) is actually the nth next file, counting from the current file. If present, n must be unsigned.

[n]/p  Displays the first screen of the previous file. The address (n) is actually the nth previous file, counting from the current file. If present, n must be unsigned.

[n]w  Scrolls another window of text. The argument, n (which must be unsigned), sets the window size to n and displays the next window of text.

[i]/pattern/[tmb]  Searches forward within the current file for the ith next occurrence of a line matching the regular expression pattern (default i is 1, the next matching pattern). The search starts right after the current window and continues to the end of the file. Typically, the matching line is displayed at the top of the window, but this can be changed by an optional character at the end of the search command. The letter t is the default and displays the line at the top of the window, m displays it in the middle of the window, and b displays it in the bottom of the window. When no letter is present, pg uses the last letter entered (or .t if no letter has been entered). Whether or not you specified the --n option, you must end this command with a newline.

[i]?pattern?[tmb], [i]^ pattern^[tmb]  Is similar to the previous command, but searches backward instead of forward. The search starts just before the current window.

Examples

The following interactive commands illustrate the flexibility of pg. Suppose you enter the command:

```
pg --n *.*
```

and that there are a large number of source files in the current directory:

1  Redisplays the first screen of the current file.

-4  Goes back 4 windows in the current file and displays a screen of text.

p  Displays the first screen of the previous file.

10w  Sets the screen size to 10 lines.

/Fred/m  Finds the first line containing

Fred

searching forward from the current position in the file, and displays a screen with that line in the middle of the screen.
Localization

`pg` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_MESSAGES`
- `LC_SYNTAX`
- `NLSPATH`


Exit values

0  
Successful completion

1  
Failure due to any of the following:

- Unknown command-line option
- Insufficient memory
- Inability to create a temporary file
- Inability to access the terminal
- Missing `string` after a `-p` option

Files

`pg` uses the following file:

`$TMPDIR/pg*`

Temporary files to allow backward reading. You can specify a different temporary directory using the `TMPDIR` environment variable.

Environment variables

`pg` uses the following environment variables:

`COLUMNS`
Contains the width of the screen in columns.

`LINES`
Contains the number of lines on the screen.

`TMPDIR`
Contains the pathname of the directory where temporary files reside.

Portability

X/Open Portability Guide, UNIX System V.

This implementation does not handle double-byte characters.

The `-screen` and `--line` options are extensions to the XPG standard.

Related information

`alias`, `ed`, `head`, `more`, `sh`, `tail`, `vi`

See Appendix C, “Regular expressions (regexp),” on page 945 for more information about `regexp`. 
pr - Format a file in paginated form and send it to standard output

Format

pr [-adFfprtW] [-n | -c n | -m] [-e [char][gap]] [-H header-fmt] [-h header] [-i[char] . [gap]] [-l n] [-n[char] [n]] [-o n] [-s[char]] [-w n] [+n] [file ...]

Description

pr prints the specified files on standard output (stdout) in a paginated form. If you do not specify any files or if you specify a file name of –, pr reads the standard input. By default, pr formats the given files into single-column 66-line pages. Each page has a five-line header. By default, the third line contains the file's path name, the date it was last modified, and the current page number; the other lines are blank. A five-line trailer consists of blank lines.

If you specify multiple columns, pr places its output in columns of equal width separated by at least one space, truncating each line to fit in its column. Input lines can be ordered down the columns or across the page on output; or different columns can each represent different files.

Options

+n
Starts printing with the nth page of each file; that is, skips the first n–1 pages. The default for n is 1.

-n
Prints n columns of output. When you specify this option, pr behaves as though you had also specified the -e and -i options. When you specify both this option and -t, pr uses the minimum number of lines possible to display the output. Do not specify this option with the -m option.

-a
Orders input lines across the page on output, instead of down. You should use this option only with -n.

-c n
Displays n columns of output. When you specify this option, pr behaves as though you had also specified the -e and -i options. When you specify both this option and -t, pr uses the minimum number of lines possible to display the output. Do not specify this option with -m.

-d
Produces double-spaced output.

-e[char][gap]
Expands each occurrence of the input tab character to a string of spaces so that the following character has the next column position which is a positive multiple of gap, plus 1. If you do not specify gap, or if it is zero, pr assumes that gap has the value of 8. If you specify the nondigit character char, pr treats it as the input tab character. Otherwise, pr uses the standard tab character.

-F
Uses form feeds to separate pages. pr normally separates pages by sending a series of <newline> characters to fill the length of a page.

-f
Uses form feeds to separate pages. When output is to a terminal, pr sounds the bell and waits for you to type a carriage return before displaying the text. pr normally separates pages by sending a series of <newline> characters to fill the length of a page.

-H header_fmt
Lets you customize your header line by specifying a format with the string header_fmt. pr recognizes the following special formatting commands:

%c
Date and time

%F
The current file name, or header string given by -h
The current user name

The default header format is equivalent to the option: -H "%c %F Page %P"

-h header
Uses the header string instead of the file name on each succeeding page header.

-l[char][gap]
Replaces white space with tabs on output. char, if given, is the output tab character. The default is the tab character. pr sets tabs every gap positions; the default for gap is 8. If this tab character differs from the input tab character and the actual data contains this tab character, the result is liable to be quite a mess.

-l n
Sets the number of lines per page of output. The default is 66. The actual number of lines printed per page is this number less 5 for the header and 5 for the trailer. If n is less than 10 (the number of lines needed for the header and the trailer), pr displays neither the header nor the trailer.

-m
Prints each file in its own column down the page. This overrides the -a option, forcing the -n option to be the number of files given. When you also specify the -n option, it gives line numbers for the first column only.

-n[char][n]
Numbers the lines of each file. Each number takes up n positions; the default for n is 5. The character char separates the number from the line; this defaults to the tab character. If char is the same as the input tab character, pr follows the number with the spaces needed to get to the next tab stop. pr may in turn replace these spaces with the output tab character if you specified the -i option. For multicolumn output, pr adds line numbers to each column. The -m option gives the line number for the first column only.

-o n
Offsets each line of output by n character positions.

-p
Pauses before the beginning of each page if output is to a terminal device. pr sounds the bell and waits for a carriage return from the controlling workstation (not the input files).

-r
Suppresses error messages due to failures when opening files.

-s[char]
Prints each column at its correct length. The character char separates columns. The default value for char is the tab character. This character is never replaced by the output tab character. Normally pr pads each column with spaces or truncates it to the exact column width. Unless the -w option is also used, -s resets the page width to 512 column positions.

-t
Does not print the headers and trailers, and quits after the last line of the file—it does not display any extra lines.

-W
Folds lines at the column width when you do not specify the -s option; pr treats each separate part of the line as a separate line.
-w n
Sets the width of the page to n column positions. If you do not specify this option, the default page width is 72 (if you did not specify the -s option) or 512 (if you did specify -s). This page width does not normally apply to single-column output; however, single-column output with the -W option does use this width.

Files
pr uses the following file:
/dev/tty
    For prompting.

Environment variables
pr uses the following environment variable:
TZ
    Contains the local time zone. pr uses this value when displaying times in header lines.

Localization
pr uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_TIME
• NLSPATH

Exit values
0
    Successful completion
1
    Failure due to any of the following:
        • Insufficient memory
        • Insufficient line width
        • Write error on stdout
2
    Syntax error or unknown command-line option

Messages
Possible error messages include:

Missing header
    You specified -h or -H but did not supply a header or header_fmt string.

Width is insufficient
    The line is not wide enough to hold the given number of columns with the given column width; or a column is not wide enough to hold the minimum amount of data.

Portability
POSIX.2, X/Open Portability Guide.
The -c, -H, -p, and -W options are extensions of the POSIX standard.

In a double-byte environment, remember that column positions are always based on the width of characters. A double-byte character may take up two columns of output (called a **thick character**), but a single-byte character will only take up one column of output (called a **thin character**). Specify column widths according to the expected thickness of characters.

For example, with a column width of 10, then ten thin characters or five thick characters are displayed.

**Related information**
cat, expand, fold, unexpand

For information about setting the local time zone, see Appendix I, “Format of the TZ environment variable,” on page 987.

---

**print - Return arguments from the shell**

**Format**

```
print [-npRrs] [-u[descriptor]] [argument ...]
```

**Description**

Calling `print` without options or with only the - option displays each `argument` to the standard output using the same escape conventions as `echo`. In this case, `print` and `echo` work the same way; see `echo`.

**Options**

The options accepted by `print` increase its utility beyond that of `echo`.

- **-n**
  Does not automatically add a new line to the end of the output.

- **-p**
  Sends output to a coprocess.

- **-R**
  Is similar to -r, except that `print` treats all subsequent options (except -n) as arguments rather than as options.

- **-r**
  Ignores escape conventions.

- **-s**
  Appends the output to the command history file rather than sending it to standard output.

- **-u[descriptor]**
  Redirects the output to the file corresponding to the single digit file `descriptor`. The default file descriptor is 1.

**Usage notes**

`print` is a built-in shell command.

**Localization**

`print` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
• NLSPATH

Exit values
0
   Successful completion
1
   Failure due to any of the following:
   • Incorrect descriptor specified with -u
   • Nonexistent coprocess
2
   Failure due to an incorrect command-line option

Messages
Possible error messages include:

Cannot print on file descriptor ...
   You tried to print on a file descriptor that was not opened for writing.

History not available
   You specified the -s option to write into a history file, but you are not now using a history file.

Portability
print is an extension to POSIX.2 and XPG.

Related information
echo, fc, read, sh

printenv - Display the values of environment variables

Format
printenv [name]
tcsh shell: printenv [name]

Description
The printenv command displays the values of environment variables. If the name argument is specified, only the value associated with name is printed. If it is not specified, printenv displays the current environment variables, one name=value pair per line.

If a name argument is specified but is not defined in the environment variable, printenv returns exit status 1; otherwise it returns status 0.

In the tcsh shell, printenv prints the names and values of all environment variables or, with name, the value of the environment variable named. For more information, see “tcsh: Invoke a C shell” on page 671.

Options
There are no options.
Examples
To find the current setting of the HOME environment variable, enter:

```
printenv HOME
```

Usage notes
1. Only one name argument can be specified.
2. `printenv SOMENAME` is equivalent to `echo $SOMENAME` for exported variables.
3. `printenv` without any arguments is functionally equivalent to `env` without any arguments.

Exit values
0  
Successful completion

1  
Failure due to one of the following:
  • More than one environment variable was specified
  • An option was specified (`printenv` has no options)

Portability
`printenv` is compatible with the AIX® `printenv` utility.

Related information
`env`, `tcsh`

`printf` - Write formatted output

Format
```
printf format [argument ...]
```

Description
`printf` writes the `argument` operands to standard output, formatted according to the `format` operand.

`format` is a format string that is composed of conversion specifications that convert and add the next `argument` to the output. `format` can contain backslash-escape sequences. These conversions are similar to those used by the American National Standard C standard. Conversion specifications have the form:

```
%[flag][width][precision][char]
```

where `flag` is one of the following options:

```
-
```
Left-justifies the field; default is right justification.

```
+
```
Always prefixes a signed value with a sign (+ or -).

```
space
```
Reserves a character position at the start of the string for the minus sign (for negative numbers) or a space (for positive numbers). If both space and - appear as flags, the space flag is ignored.
Prefixes octal values with 0 and hexadecimal values with 0x or 0X. For floating-point values, this causes the decimal point always to be displayed even if no characters follow it.

0

Pads numeric values with leading zeros. If both 0 and - appear as flags, the 0 flag is ignored.

width is the minimum field width of the output field. If the converted value is shorter than the minimum width, printf pads it with spaces or zeros.

In a string, precision is the maximum number of bytes to be printed from the string; in a number, the precision is the number of digits to be printed to right of the decimal point in a floating-point value. width or precision can be specified as *, in which case the value is read from the next argument, which must be an integer. For example:

```
printf "%*.10d\n" 20 10 200
```

is equivalent to:

```
printf "%20.10d\n" 200
```

The conversion character char is one of the following:

b

A string that may contain a backslash-escape sequence.

c

Single character of an integer value; the first character of a string.

d

Decimal integer.

e,E

Floating point (scientific notation).

f,F

Floating point.

g,G

The shorter of e and f (suppresses nonsignificant zeros).

i

Decimal integer.

o

Unsigned octal integer.

s

String.

u

Unsigned decimal integer.

x,X

Unsigned hexadecimal integer.

When there are more arguments than positions in format, the format string is applied again to the remaining arguments. When there are fewer arguments than there are positions in the format string, printf fills the remaining positions with null strings (character fields) or zeros (numeric fields).

Caution

The POSIX.2POSIX.2 printf facility (like the C language printf on which it is based), does not accommodate double-byte characters gracefully when using %c conversion, or either of %b or %s conversions with a specified precision. Use these features cautiously when you have double-byte characters in the character set.

In a double-byte environment, normal backslash-escape characters are handled correctly (printf shifts state as required) but octal and hexadecimal escape characters do not change state. This behavior is
significant in a shift-lock environment. For example, if an octal escape character contains the shift-in character, it is the user’s responsibility to ensure that there is also a shift-out character. Further, an octal or hexadecimal backslash escape character that comes immediately after a double-byte character may or may not be processed in the shifted state.

For more information about double-byte character environments, see “Using the double-byte character set (DBCS)” on page 7.

**Localization**

`printf` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_NUMERIC`
- `LC_SYNTAX`
- `NLSPATH`


**Exit values**

- 0
  
  Successful completion.

- >0
  
  The number of failures due to any of the following reasons:
  
  - Missing format specifications.
  - Arguments that were supplied for a `format` string that does not accept them (that is, that has no `%s`).
  - Incorrect integer argument.
  - Incorrect floating-point argument.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX System V.

The `%F` format and the handling of `*` as a width or precision argument are extensions of the POSIX standard.

**Related information**

- `echo`, `printf`

---

**ps - Return the status of a process**

**Format**

```bash
```

**Description**

`ps` displays status information about processes, and optionally, the threads running under each process.
By default, for each process that is associated with the user's terminal, `ps` displays the process ID (PID), TTY, processor time used (TIME), and name of the command (COMM).

The `-a`, `-A`, and `-e` options can be used to show information that is associated with all available or accessible processes on the system. However, these options can only show information for those processes the user has appropriate privileges to access. The `-g`, `-G`, `-p`, `-s`, `-t`, `-u`, and `-U` options can be used to select specific processes by process ID, terminal ID, and user name.

The `-f`, `-j`, and `-l` options can be used to display additional status fields using predefined formats. The `-o` format option allows the user to select specific status fields and to define the format in which these fields are displayed.

`ps` displays information for each thread that is associated with a process when the `-m` and `-o THREAD` options are used. Output lines for thread information immediately follow the output line for the parent process. Because the default behavior of `ps` displays process status fields only, to provide meaningful thread output, the `-o` option is used to specify thread-specific status fields. There are some conditions, such as when the process is in a terminating or zombie state, where thread data cannot be captured. In these cases, a single thread output line is displayed showing a `?` in the thread output fields.

**Restriction:** The `ps` command has a limitation of 80 chars for the column `args` before z/OS V2R1. On V2R1 and later, the column `args` are updated to support 1024 chars.

**Options**

`ps` accepts several options. When a description says that `ps` lists all processes, it means all the processes on the system, if you have appropriate privileges.

The fields `pcpu`, `nice`, `pri`, `addr`, and `wchan` are unsupported and will always display a dash.

**-A**
Displays information about all available processes. You can specify `-A`, `-a`, and `-e` in any combination; however, `-a` overrides both `-A` and `-e`.

**-a**
Displays information about all processes that are associated with terminals. You can specify `-A`, `-a`, and `-e` in any combination; however, `-a` overrides both `-A` and `-e`.

**-c**
Displays more detailed information about processes for the `-f` and `-j` options. `-c` is accepted but not currently implemented.

**-d**
Displays information for all processes except group leaders.

**-e**
Displays information about all accessible processes. You can specify `-A`, `-a`, and `-e` in any combination; however, `-a` overrides both `-A` and `-e`.

**-f**
Displays information as if the user specified:

```
-o user=UID -opid,ppid,pcpu=C -ostime,tty=TTY -oatime,args=CMD
```

**-G grouplist**
Displays information about processes with real group ID numbers in `grouplist`. Separate numbers in `grouplist` with either blanks or commas.

**-g idlist**
Displays information about processes with process ID numbers in `idlist`. Separate the numbers in `idlist` with either blanks or commas.

**-j**
Displays information as if the user specified:

```
-o pid,sid,pgid=PGRP -o tty=TTY -oatime,args
```
Displays information as if the user had specified:

```
-o oflags,state,ruid=UID -opid,ppid,pcpu=C -opri,nice,addr,vsz=SZ
-owchan,tty=TTY -oatime,comm=CMD
```

Displays thread status information. Output lines for thread status immediately follow the output line for the parent process. Process-only status fields will contain dashes for thread output lines. Since the default behavior of `ps` is to display process-only status fields, to provide meaningful thread output, the `-o` option should be used to specify thread supported status fields. If `-o` THREAD is used, `-m` is assumed.

- `n name` Specifies the name of the executable file containing the kernel symbol table. This option is not supported and is ignored.

- `o format` Displays information according to the given `format` specifications. If `-o` is not used, the default format is the same as specifying:
  
  `o pid,tty=TTY -o atime,comm`

  See “Format specifications” on page 552.

- `p proclist` Displays information for processes with process ID numbers in `proclist`. Separate numbers in `proclist` with either blanks or commas.

- `s idlist` Displays information for processes with session ID numbers in `idlist`. Separate the numbers in `idlist` with commas.

- `t termlist` Displays information for processes with terminals in `termlist`. Denote terminals in `termlist` with either the file name of the device (for example, tty04). Or, if the file name begins with tty, you can specify the characters following tty. For example, tty04 and 04 both denote the same terminal. Terminals in `termlist` are separated by either blanks or commas.

- `U userlist` Displays information for processes with user IDs in `userlist`. Items in `userlist` can be user ID numbers or login names, and are separated by commas.

- `u userlist` Displays information for processes with user IDs in `userlist`. Items in `userlist` can be user ID numbers or login names, and are separated by commas.

**Format specifications**

Using the `-o` option, the user can define the status fields that are displayed and their column headings. If you do not specify the `-o` option, `ps` displays the information as though you specified:

```
-o pid,tty=TTY -o atime,comm
```

The `format` specification is a list of status field names that are separated with blanks or commas. However, if the list of names is separated by blanks, the list must be contained in single quotation marks. The following list shows the status field names that are recognized by `ps`.

Multiple `-o format specifications` can be provided and, in the case where user-specified column headings are defined, these specifications might be necessary.

The first line of `ps` output contains column headings for each status field. Each status field has a default heading which can be overridden by the user by specifying `-newheading` after the status field. When a
new heading is specified, it must be the last field given on the -o option. To specify additional fields, it is necessary to use additional -o statements.

For example, if you want to display the process ID (pid), real user name (ruser), and command name (comm), but change the heading for the real user name from the default of (RUSER) to WHO, use:

```
-o pid,ruser=WHO -o comm
```

An additional -o is required when comm is specified because the last argument must be user-specified headings (in this case ruser=WHO).

If you specify = with no heading, ps displays that column without a heading. If none of the columns have a heading, ps displays no heading line.

In a double-byte locale, user-defined headings might contain multibyte (double-byte) characters.

The following list shows the names that ps recognizes. The list is separated into three groups:

**process only**
These are fields that only display meaningful data for process output lines. For thread output lines, a dash is shown in these fields.

**thread only**
These are fields that only display meaningful data for thread output lines. For process output lines, a dash is shown in these fields.

**processes and threads**
These are fields that apply to both processes and threads. For example, state is meaningful because both processes and threads have a state that can be determined for them.

At the end of each description, the default column heading is inside square brackets.

The Process Only group:

**addr**
Displays the address of the process. This field is not supported and will display a dash. [ADDR]

**args**
Displays the command that is running, with all its arguments. [COMMAND]

**atime**
Displays the amount of processor time that the process has used since it began running. Time is displayed in one of the following abbreviated formats:

- `days d hours`
- `hours h minutes`
- `minutes : seconds`

depending on the amount of processor time used. [TIME]

**attr**
Displays the process attributes. [ATTR]

The following values might be displayed:

**B**
Shutdown blocking process; will prevent the shutdown from proceeding until it either de-registers as a blocking process or ends.

For more information about the B attribute, see [z/OS UNIX System Services Planning](#).

**P**
Permanent process; will survive across a shutdown.

For more information about the P attribute, see [z/OS UNIX System Services Planning](#).

**R**
Respawnable process; will be restarted when it ends.
For more information about the R attribute, see z/OS UNIX System Services Planning.

T
Tracing is active.
For more information about tracing, see “bpxtrace - Activate or deactivate traces for processes” on page 68.

comm
Displays the name of the command that is running without its arguments. This string is padded on the right if necessary. [COMMAND]

etime
Displays the amount of real time that has elapsed since the process began running. ps shows the time in the form:

```
[dd-]hh:mm:ss
```

where dd is the number of days, hh is the number of hours, mm is the number of minutes, and ss is the number of seconds. [ELAPSED]

gid
Displays the effective group ID of the process. [EGID]
group
Displays the effective group ID of the process, as a group name if possible and as a decimal group ID if not. [GROUP]

jobname
Displays the job name. [JOBNAME]
nice
Displays the nice value (urgency) of the process as a decimal value. This field is not supported and will display a dash. [NI]

pcpu
Displays a percentage value giving the ratio of processor time used to processor time available. This field is not supported and will display a dash. [%CPU]

pgid
Displays the process group ID as a decimal value. [PGID]

pid
Displays the process ID as a decimal value. Decimal pids are reported with default actions. [XPID]

ppid
Displays the parent process ID as a decimal value. [PPID]

pri
Displays the process priority. This field is not supported and will display a dash. [PRI]

rgid
Displays the real group ID of the process. [GID]

rgroup
Displays the real group ID of the process, as a group name if possible and as a decimal group ID if not. [RGROUP]

ruid
Displays the real user ID of the process. [UID]

ruser
Displays the real user ID of the process, as a user name if possible and as a decimal user ID otherwise. [RUSER]

sid
Displays the session ID of the process. [SID]

stime
Displays the start time of the process. [STIME]
thdcnt
Displays the total number of threads. [THCNT]

time
Displays the amount of processor time that the process has used since it began running. ps displays
this time in form similar to that used by etime. [TIME]
tty
Displays the name of the controlling terminal (if any). [TT]
uid
Displays the effective user ID of the process. [EUID]
user
Displays the effective user ID of the process, as a user name if possible and as a decimal user ID
otherwise. [USER]
vsz
Displays the amount of (virtual) memory that the process is using, as a decimal number of kilobytes.
[VSZ]

vsz64
Displays the virtual storage used above the 2-gigabyte bar [VSZ64].

wchan
Displays the channel upon which the process is waiting. This field is not supported and will display a
dash. [WCHAN]
xasid
Displays the address space ID as a hexadecimal value (Note: a non-hexadecimal ASID is not
supported). [ASID]
xpgid
Displays the process group ID as a hexadecimal value. [XPGID]

xpidd
Displays the process ID as a hexadecimal value. [XPID]

xppidd
Displays the parent process ID as a hexadecimal value. [XPPID]

xsidd
Displays the session ID as a hexadecimal value. [XSID]

The Thread Only group:

lpidd
Displays the latch pid waited for. [1pid]

lsyscall
Displays the last five syscalls. This is a 20 character string consisting of five 4-character syscalls with no delimiting characters between them. From left-to-right the syscalls are ordered from most recent to oldest. In the following example of lsyscall output, 1WAT is the most recent syscall: 1WAT1SPM1SPM1TSP. [LASTSYSC]

semnum
Displays the semaphore number of the semaphore the thread is in a wait state for. (Note: a semaphore number is only available when the thread is in a semaphore wait state (state field value equals d), otherwise, a dash will be displayed). [SNUM]

semval
Displays the semaphore value of the semaphore the thread is in a wait state for. (Note: a semaphore value is only available when the thread is in a semaphore wait state (state field value equals D), otherwise, a dash will be displayed). [SVAL]

sigmask
Displays the signal pending mask as a hexadecimal value. [SIGMASK]

syscall
Displays the current syscall (for example, 1fork for fork). [SYSC]

tagdata
Displays the tag that is assigned to the thread using pthread_tag_np(). If a tag was not assigned, a dash is displayed. [TAGDATA]

wtime
Displays waiting time in one of the following abbreviated formats:

- days d hours
- hours h minutes
- minutes : seconds

depending on the amount of waiting time to display. [TIME]

xtcbaddr
Displays the tcb address as a hexadecimal value. A non-hexadecimal tcb address is not supported. [TCBADDR]

xstid
Displays the short thread ID as a hexadecimal value. This is the low order word (the sequential value) of the thread ID. A non-hexadecimal short thread ID is not supported. [STID]

xtid
Displays thread ID as a hexadecimal value. A non-hexadecimal thread ID is not supported. [TID]

The Processes and Threads group:
flags
Displays the state field values using a hexadecimal representation. flags is the four-byte value determined when a bit is set to one for each corresponding state that is active. Below is the state-to-state bit mapping for the currently defined state values:

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>01111111</td>
<td>11111111</td>
<td>11111111</td>
<td>11010000</td>
</tr>
<tr>
<td>state</td>
<td>ABCDEFG</td>
<td>JK</td>
<td>NO</td>
<td>RS UVWX YZ</td>
</tr>
</tbody>
</table>

For example, if a thread or process had a state field value of `1W`, then the following bits would be set:

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>00000000</td>
<td>00000000</td>
<td>00000010</td>
<td>00010000</td>
</tr>
<tr>
<td>state</td>
<td>state</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which when represented as a hexadecimal value would be `210`. [F]

state
Displays the process state. [STATE] Various values can be printed in this field:

1. A single task using assembler callable services.
4. Communication system kernel wait.
5. Semaphore operation wait.
6. Quiesce frozen.
7. File system kernel wait.
8. MVS Pause wait.
9. One or more pthread created tasks (implies M as well).
10. Swapped out.
11. Pthread created.
12. Other kernel wait (for example, pause or sigsuspend).
13. Canceled, parent has performed wait, and still session or process group leader.
15. Medium weight thread.
**P**  Ptrace kernel wait.

**R**  Running (not kernel wait).

**S**  Sleeping.

**T**  Stopped.

**U**  Initial process thread.

**V**  Thread is detached.

**W**  Waiting for a child (wait or waitpid function is running).

**X**  Creating a new process (fork function is running).

**Y**  MVS wait.

**Z**  Canceled and parent has not performed wait (Z for zombie).

**THREAD**

THREAD [THREAD] is a synonym for specifying the following fields:

```
-m -o ruser=UID -o pid,ppid,xstid,state=STATE -o atime,syscall,args=CMD
```

The following is an example of how this output will appear:

```
<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>PPID</th>
<th>STID</th>
<th>STATE</th>
<th>TIME</th>
<th>SYSC</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>WELLIE8</td>
<td>67108867</td>
<td>15099496</td>
<td>-</td>
<td>1W</td>
<td>0:25</td>
<td>-</td>
<td>sh</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>W</td>
<td>0:17</td>
<td>1WAT</td>
<td>-</td>
</tr>
<tr>
<td>WELLIE8</td>
<td>1073741830</td>
<td>67108867</td>
<td>-</td>
<td>1Y</td>
<td>0:00</td>
<td>-</td>
<td>./ps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>Y</td>
<td>0:00</td>
<td>1GTH</td>
<td>-</td>
</tr>
</tbody>
</table>
```

**Environment variables**

**ps** uses the following environment variable:

**COLUMNS**

Contains the maximum number of columns to display on one line.

**Localization**

**ps** uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_CTYPE**
- **LC_MESSAGES**
- **LC_TIME**
- **NLSPATH**

Exit values

0
  Successful completion.

1
  Failure due to the inability to open the process table.

2
  Failure due to any of the following reasons:
  • Unknown command-line option.
  • Missing format string after -o.
  • Missing lists after other options.
  • Too many arguments on the command line.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide.
The -c, -d, -e, -f, -g, -j, -l, -m, -n, -s, and -u options are extensions of the POSIX standard.

Related information

jobs, kill

pwd - Return the working directory name

Format

pwd

Description

pwd displays the absolute path name of the working directory to standard output.

If the current working directory is a symbolic link to another directory, the path name that is displayed depends on the setting of the shell’s logical flag. See set for more information.

Usage notes

pwd is a built-in shell command and is also a separate utility.

Localization

pwd uses the following localization environment variables:
  • LANG
  • LC_ALL
  • LC_MESSAGES
  • NLSPATH

Exit values

0
  Successful completion.

1
  Inability to determine the working directory.
Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Related information
set, sh

r - Process a command history list

Format
r [old=new] [specifier]

Description
r is an alias for fc -s. Like fc -s, r reenters exactly one command without going through an editor. If a command specifier is given, r selects the command to reenter; otherwise, r uses the last command entered. To perform a simple substitution on the command before reentry, use a parameter of the form old=new. The string new replaces the first occurrence of string old. r displays the (possibly modified) command before reentering it.

See “fc - Process a command history list” on page 302 for more information.

Related information
fc, history, sh

read - Read a line from standard input

Format
read [-prs] [-u[dl]] [variable?prompt] [variable ...]

Description
When you call read without options, it reads one line from the standard input, breaks the line into fields, and assigns the fields to each variable in order.

To determine where to break the line into fields, read uses the built-in variable IFS (which stands for internal field separator). Encountering any of the characters in IFS means the end of one field and the beginning of the next. The default value of IFS is blank, tab, and newline.

In general, a single IFS character marks the end of one field and the beginning of the next. For example, if IFS is colon (:), read considers the input a : b to have three fields: a, an empty field, and b. However, if IFS contains blanks, tabs or escaped newlines, read considers a sequence of multiple blanks, tabs, or escaped newlines to be a single field separator. For example, a   b has two fields, even though there are several blanks between the a and b.

The nth variable in the command line is assigned the nth field. If there are more input fields than there are variables, the last variable is assigned all the unassigned fields. If there are more variables than fields, the extra variables are assigned the null string (""").

By default, unless the read option is specified, <backslash> acts as an escape character. An unescaped <backslash> preserves the literal value of the following character, except for a <newline>. If a <newline> follows the <backslash>, the read utility interprets this as line continuation. The <backslash> and <newline> are removed before splitting the input into fields. All other unescaped <backslash> characters are removed after the input is split into fields.
The environment variable `REPLY` is assigned the input when no variables are given. The exit status of `read` is 0, unless it encounters the end of the file.

**Options**

- **-p**
  - Receives input from a coprocess.

- **-r**
  - Treats input as raw data, ignoring escape conventions. For example, `read -r` does not interpret a final backslash (\) as a line continuation character, but as part of the input.

- **-s**
  - Adds input to the command history file as well as to the variables specified with `variable`.

- **-u[d]**
  - Reads input from the single-digit file descriptor d, rather than from the standard input. The default file descriptor is 0.

When the first variable parameter has the form:

```
variable?prompt
```

it defines a prompt for input. If the shell is interactive, `read` sends the prompt to the file descriptor d if it is open for write and is a terminal device. The default file descriptor for the prompt is 2.

**Examples**

```
IFS=':'
while read name junk junk1 junk2 junk3
do
  echo $name
done </samples/comics.lst
```

-provides a list of comic names from the sample `comics.lst` file.

**Environment variables**

`read` uses the following environment variables:

**IFS**

Contains a string of characters to be used as internal field separators.

**PS2**

Contains the prompt string that an interactive shell uses when it reads a line that ends with a backslash and you did not specify the `-r` option, or if a here-document is not terminated after you enter a newline.

**REPLY**

Contains the input (including separators) if you did not specify any variables. The ability of omitting the variable from the command and using the environment variable `REPLY` is an extension.

**Localization**

`read` uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_MESSAGES**
- **NLSPATH**

readonly

Usage notes
read is a built-in shell command.

Exit values
0
   Successful completion
1
   Failure due to any of the following reasons:
   • End of file on input
   • Incorrect variable
   • Incorrect descriptor specified after \-u
   • Missing coprocess
2
   Incorrect command-line argument

Messages
Possible error messages include:

Cannot read on file descriptor ...
   You tried to read a file descriptor that was not opened for reading.

Portability
POSIX.2, X/Open Portability Guide.
The -p, -s, and -u options are extensions of the POSIX standard.

Related information
continue, fc, print, sh

readonly - Mark a variable as read-only

Format
readonly [-p] [name[=value] ...]

Description
readonly prevents subsequent changes in the value of any of the name arguments. Parameters of the form:

name=value

assign value to name as well as marking name read-only. If readonly is called without arguments, it lists, with appropriate quoting, the names you have set as read-only in the following format:

Variable="value"
Options

-p
Displays export name=value pairs that, when read by a shell, ensures the read-only status and values of variables. The shell formats the output so it is suitable for reentry to the shell as commands that achieve the same attribute-setting results.

Because it is not possible to change a read-only variable, you cannot source the output unless you go to a new shell.

Usage notes
readonly is a special built-in shell command.

Localization
readonly uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH


Exit values

0
Successful completion

1
An attempt to give read-only status to a variable that is already read-only

2
Failure due to incorrect command-line argument

Portability
POSIX.2, X/Open Portability Guide.

The behavior given for calling readonly with no arguments is an extension of the POSIX standard.

Related information
alias, sh, typeset

renice - Change priorities of a running process

Format

renice [-n increment] [-g|-p -u] ID ...
renice priority [-p] pid ... [-g pgrp ...] [-p pid ...] [-u user ...]
renice priority -g pgrp ... [-g pgrp ...] [-p pid ...] [-u user ...]
renice priority -u user ... [-g pgrp ....] [-p pid ....] [-u user ...]

Description

renice changes the priority of one or more running processes. Normal users can change only the priority of processes that have the same real or effective user ID as the real or effective user ID of the process that calls renice. Privileged users can set the priority of any process.
You can specify the new **priority** as a decimal integer, with higher values indicating more urgent priority. The range of priorities is site-specific, and you might require appropriate privileges for some priority values.

When you change the priority of a process group, the priority of all processes in that group are changed. If the string `--` appears in the arguments, `renice` does not interpret it as the end of command-line arguments. This is an exception to the usual POSIX syntax rules.

**Options**

- **-g**
  Treats all following IDs (or just `pgrps` in the obsolescent versions) as process group IDs.

- **-n increment**
  Adjusts the system scheduling priority of the specified processes by `increment`. Positive `increments` lower the priority while negative `increments` result in a higher priority.

  **Note:** Negative `increments` might require appropriate privileges.

- **-p**
  Treats all following IDs (or just `pids` in the obsolescent versions) as process IDs.

- **-u**
  Treats all following IDs (or just `users` in the obsolescent versions) as either user names or numeric user IDs.

**priority**

A number that indicates an absolute priority value (higher numbers reflect higher priorities).

If no `-p`, `-g`, or `-u` option appears on the command line, `renice` assumes `-p`.

**Localization**

`renice` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `NLSPATH`


**Exit values**

0

Successful completion

1

Failure due to one of the following:

- Incorrect command-line argument
- The wrong number of command-line arguments
- A **priority** that is outside the range
- An incorrect **priority** argument
- An incorrect **ID** argument
- Missing arguments following one of the options

2

Failure because the system does not recognize the ID in a `-u` option
Portability
POSIX.2 User Portability Extension, UNIX systems.
POSIX considers all but the first form of the renice command to be obsolescent.

Related information
nice

return - Return from a shell function or . (dot) script

Format
return [expression]

Description
return returns from a shell function or . (dot) script. The exit status is the value of expression. The default value of expression is the exit status of the last command run.

Usage notes
return is a special built-in shell command.

Localization
renice uses the following localization environment variables:
• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH

Exit values
The current function or script returns the value of expression. If no expression is given, the exit status is the exit status of the last command run.

Portability
POSIX.2, X/Open Portability Guide.

Related information
exit, sh

rlogind - Validate rlogin requests

Format
Description

Warning: The rlogin daemon is by nature unsecure because it requires user authentication that uses cleartext sockets. A remote attacker can exploit this vulnerability to sniff logins and passwords. Do not use this daemon if possible.

The rlogind program is the server for the remote login command rlogin commonly found on UNIX systems. It validates the remote login request and verifies the password or password phrase of the target user. It starts a z/OS shell for the user and handles conversion between ASCII and EBCDIC code pages as data flows between the workstation and the shell.

The rlogind program is given control via an execl() issued by the inetd daemon.

Rule: Always invoke rlogind from inetd through the /etc/inetd.conf file. Do not invoke it from the shell. inetd sets up certain files and sockets that are needed by rlogind. Invoking rlogind directly gives unpredictable results.

Options

-a
Checks the requester's Internet address against the local gethostbyname() file. This option has no effect because the rlogin program never uses the .rhosts file for authentication.

-d
Enables the debugging option. Informational messages on the rlogin process are written to the system log.

-l
The .rhosts file for authentication is not used. This option has no effect because the rlogin program never uses the .rhosts file for authentication.

-L
Allows the calling of a ruserok exit that lives in /usr/sbin. A return code zero allows bypassing of password or password phrase checking. The installation is responsible for providing the ruserok exit.

Note: IBM does not recommend using this capability. Using this capability might open security holes, allowing unauthorized users to access and modify files and MVS data sets. Even with the most rigorous checking in the ruserok exit, it is important to keep in mind the well-known IP spoofing attacks that make it impossible to accurately identify the remote user's identity.

Do not specify the -L flag. APARS will not be accepted for security problems that are caused by the use of this option.

When the -L flag is specified, /usr/sbin/ruserok is called, passing:

• The name of the program, /usr/sbin/ruserok.
• "hostname" or "hostname.domainname" of the client.
• A superuser flag, an integer set to 1 if the user wants to be superuser.
• Client user name, the user name on the client system.
• Server user name, the user name on the server's system.

If the ruserok program exits with a zero return value, the user is allowed to login. Otherwise, normal password or password phrase checking is done.

Note: If the FACILITY class is active, and BPX.DAEMON is defined, then the user names for both inetd and rlogind must be permitted to BPX.DAEMON. The ruserok program (as well as inetd and rlogind) must be marked program-controlled.

-m
Enables the multiprocessing support in the user's address space. Using the -m option uses fewer system resources and provides faster performance for the user.

If you do not specify -m, each rlogin request causes two MVS address spaces to be consumed. The first address space is the rlogind code, which provides the user connection to the socket, and the
second is the user’s shell. In this mode, all shell functions behave in a manner conformant to the standards.

If you specify -m, the rlogin process and the shell process share the address space by using z/OS UNIX System Services support for multiple processes in an address space. Using -m has the potential of doubling the number of users who are supported via rlogin.

**Note:** If you issued rlogind with the -m option, the shell process cannot execute a setuid program that replaces the shell. This causes functions like newgrp to fail. In this situation, you might want to create a secondary shell that runs in its own address space.

**-n**
Disables the transport-level keepalive messages. The messages are enabled by default.

**Usage notes**

1. The rlogind program normally converts all error and warning messages to ASCII and then sends them to the originating terminal.

   However, when the C runtime library writes error messages, the rlogind program cannot intercept them to convert the messages to ASCII. Therefore, these messages are written to the file /tmp/rlogind.stderr or /tmp/rlogind2.stderr.

   These two files must be predefined in /tmp, and owned by the superuser (UID 0). The files should have permissions of rw–rw–rw or rw––w––w–. In addition, the sticky bit must be set for the /tmp directory so that these files (and other files in /tmp) cannot be removed except by the files’ owners or the superuser.

2. rlogind is not affected by the locale information that is specified in locale-related environment variables.

**Related information**

inetd

---

**rm - Remove a directory entry**

**Format**

```plaintext
rm [-fiRrv] file ... file ...
```

**Description**

`rm` removes files if it is a valid path name. If you specify either . or .. as the final component of the path name for a file, `rm` displays an error message and goes to the next file. If a file does not have write permission set, `rm` asks you if you want to delete the file; type the yes expression defined in LC_MESSAGES (the English expression is typically y or yes) if you want it deleted.

**Restriction:** A file can be removed by any user who has write permission to the directory that contains the file, unless that directory has its sticky bit turned on. If the file is in a directory whose sticky bit is turned on, only the file owner, the owner of the directory, or a superuser can remove the file.

If you delete a file, remember that the space is not actually reclaimed until any processes that have that file open either terminate or close that file. See the “fuser - List process IDs of processes with open files” on page 321 command to find out how to get more information about what processes are accessing a particular file or directory.

**Options**

- **-f**
  Deletes read-only files immediately without asking for confirmation. When you specify this option and a file does not exist, `rm` does not display an error message and does not modify the exit status. If you
specify both -f and -i, rm uses the option that appears last on the command line. If no files are specified, rm -f will not issue an error.

-i
Prompts you for confirmation before deleting each file. If you specify both -R and -i, rm also prompts you for confirmation before deleting a directory. If you specify both -f and -i, rm uses the option that appears last on the command line.

-R
Recursively removes the entire directory structure if file is a directory.

-r
Is equivalent to -R.

-v
Displays a list of files that were removed.

Localization
rm uses the following localization environment variables:
• LANG
• LC_ALL
• LC_COLLATE
• LC_CTYPE
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH

Exit values
0
Successful completion.
1
Failure due to any of the following:
  • Inability to remove a file.
  • Attempt to remove directory without specifying -r or -R.
  • Inability to find file information when using -r or -R.
  • Inability to read directory when using -r or -R.
2
Failure due to any of the following:
  • Incorrect command-line option.
  • No file was specified.

Portability
POSIX.2, X/Open Portability GuideUNIX systems.

Related information
cp, mv, rmdir
rmdir - Remove a directory

Format

```
rmdir [-p] directory ...
```

Description

`rmdir` removes each requested `directory`. Each directory must be empty for `rmdir` to be successful.

Options

- `p`
  Removes all intermediate components. For example:

  ```
rmdir -p abc/def/ghi
  ```

  is equivalent to:

  ```
rmdir abc/def/ghi
  rmdir abc/def
  rmdir abc
  ```

Localization

`rmdir` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values

0
  Successful completion.
1
  Failure to remove the directory. For example, the object is not a directory, the directory still contains files or subdirectories, or the user is not authorized.
2
  Failure because either the command-line option was incorrect or a directory name was not specified.

Messages

Possible error messages include:

Nonempty directory
  Files or other directories are found under the directory to be removed. Use `rm -r` to remove the directory.

No such directory
  The requested directory does not exist or is otherwise inaccessible.
Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Related information
mkdir, rm

runcat - Pipe output from mkcatdefs to gencat

Format
runcat CatalogName SourceFile [CatalogFile]

Description
runcat invokes the mkcatdefs command and pipes the message catalog source data (the output from mkcatdefs) to the gencat utility.

The file specified by the SourceFile parameter contains the message text with your symbolic identifiers. The mkcatdefs program uses the CatalogName parameter to generate the name of the symbolic definition file by adding .h to the end of the CatalogName value, and to generate the symbolic name for the catalog file by adding MF_ to the beginning of the CatalogName value. The definition file must be included in your application program. The symbolic name for the catalog file can be used in the library functions (such as the catopen subroutine). SourceFile cannot be stdin.

The CatalogFile parameter is the name of the catalog file created by the gencat command. If you do not specify this parameter, the gencat command names the catalog file by adding .cat to the end of the CatalogName value. This filename can also be used in the catopen subroutine.

Examples
To generate a catalog named test.cat from the message source file test.msg, enter:

```
runcat test test.msg
```

Related information
dspcat, dspmsg, gencat, mkcatdefs

script - Makes a typescript of a terminal session

Format
script [-aq] [file]

Description
script makes a typescript of everything displayed on the terminal. The typescript is written to the file specified by the file parameter. If no file name is given, the typescript is saved in the current working directory with the file name typescript. If the file exists, the default behavior is to overwrite its contents.

script is useful for recording shell session activity for troubleshooting and documenting purposes.
Options

- `a`  
  Appends the typescript to the file.

- `q`  
  Quiet mode. All diagnostic messages are suppressed.

Examples

To record shell session activity in order to document the removal of files, follow these steps:

1. Begin recording the shell session activity by issuing the `script` command:

   ```
   script
   ```

2. Start removing the files and then end the recording of the shell session activity:

   ```
   rm -v removeme*  
   exit
   ```

3. After the recording ends, the contents of the typescript created by `script` will be similar to the following:

   ```
   Script command is started on Fri Jan 29 11:25:15 2010.  
   SYS1: /u/user1/mydir> rm -v removeme*  
   removeme1  
   removeme2  
   removeme3  
   SYS1: /u/user1/mydir> exit  
   Script command is complete on Fri Jan 29 11:25:17 2010.
   ```

Environment variables

`script` uses the following environment variable:

**SHELL**  
Contains the name of the shell to be forked by `script`.

Localization

`script` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_TOD`
- `NLSPATH`


Usage notes

1. `script` forks and executes a shell according to the value of the `SHELL` environment variable. If the environment variable is not set, `script` uses the `/bin/sh` shell. `script` ends when the shell process exits. Use either `exit` or `Ctrl-D` to exit the shell process.

2. Because `script` writes everything in the typescript to the file including backspaces and prompts, commands that modify terminals such as `vi` might create unexpected data in the typescript.

3. Before and after running `script`, ensure that access to the file containing the typescript is properly controlled because the file might contain sensitive data.
4. script does not support setting 3270 passthrough mode during the shell session. As a result, OEDIT, OBROWSE, and other utilities requiring 3270 passthrough mode will fail.

5. script creates a new session and controlling terminal for the shell process. A login accounting entry is not added to /etc/utmpx for this session and terminal.

6. script cannot be run in a background process. For example, using an & at the end of the script command is not supported. Executing the script command via BPXBATCH is also not supported.

7. Do not access the typescript file in use by script during the shell session, or unexpected results might occur.

**Exit values**

0  
Successful start of the script command

1  
Failure due to any of the following:

- Unable to access the standard output, input or error file descriptors
- An incorrect command-line option
- Unable to open, write to, or initialize the typescript file.
- Unable to open or write to the pseudoterminal
- Unable to access the controlling terminal
- Unable to allocate system resources

129-255  
script was interrupted by a signal. The exit value is the signal number combined with 128. For example, SIGTERM (signal number 15) results in an exit value of 143.

**Portability**

An approved POSIX standard does not exist for script.

**Related information**

tee

---

**sed - Start the sed noninteractive stream editor**

**Format**

```
   sed [-BEn] [-W option[option] ... script [file ...]  
   sed [-BEn] [-e script] ... [-f scriptfile] ... [-W option[option]] ... [file ...]
```

**Description**

The sed command applies a set of editing subcommands contained in script to each argument input file. If more than one file is specified, they are concatenated and treated as a single large file. script is the arguments of all -e and -f options and the contents of all script files. You can specify multiple -e and -f options; commands are added to script in the order specified.

If you did not specify file, sed reads the standard input.

sed reads each input line into a special area known as the pattern buffer. Certain subcommands [gGhHx] use a second area called the hold buffer. By default, after each pass through the script, sed writes the final contents of the pattern buffer to the standard output.
Options

-B   Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-E   Uses extended regular expressions. Normally, sed uses basic regular expressions.

-e script   Adds the editing subcommands script to the end of the script.

-f scriptfile   Adds the subcommands in the file scriptfile to the end of the script.

-n   Suppresses all output except that generated by explicit subcommands in the sed script [acilnpPr]

-W option[,...]   Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

  filecodeset=   Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

  If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

  If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

  When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

  pgmcodeset=   Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

  If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

  If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

  Restriction: The only supported values for pgmcodeset are IBM-1047 and 1047.

If you need only one script argument, you can omit the -e and use the first form of the command.

sed subcommands are similar to those of the interactive text editor ed, except that sed subcommands necessarily view the input text as a stream rather than as a directly addressable file.
Each line of a sed script consists of one or more editing commands. The commands can be preceded by either semicolons or blanks, or both. Each editing command contains up to two addresses, a single letter command, and possible command arguments. The last editing command is followed with a terminating newline. The newline is optional in script strings typed on the command line.

```
[addr[,addr]] command [arguments]
```

**Subcommands**

sed subcommands necessarily view the input text as a stream rather than as a directly addressable file. Script subcommands can begin with zero, one, or two addresses, as in ed.

- Zero-address subcommands refer to every input line.
- One-address subcommands select only those lines matching that address.
- Two-address subcommands select those input line ranges starting with a match on the first address up to an input line matching the second address, inclusive. If the second address is a number less than or equal to the line number first selected, only one line is selected.

Permissible addressing constructions are:

- `n` The number `n` matches only the `n`th input line.
- `$` This address matches the last input line.
- `/regexp/` This address selects an input line matching the specified regular expression `regexp`. If you do not want to use slash (/) characters around the regular expression, use a different character (but not backslash or newline) and put a backslash (\) before the first one. For example, if you want to use % to enclose the regular expression, write `\%regexp%`. If an `regexp` is empty (that is, no pattern is specified) sed behaves as if the last `regexp` used in the last command applied (either as an address or as part of a substitute command) was specified.

A command can be preceded by a '!' character, in which case the command is applied if the addresses do not select the pattern space. When the variable `_UNIX03=YES` is set, one or more '!' characters are allowed, and it is not allowed to follow a '!' character with blank(s). When the variable `_UNIX03` is unset or is not set to YES, only one '!' character is allowed, and it is not allowed to follow a '!' character with blank(s).

The following sed subcommand summary shows the subcommands with the maximum number of legitimate addresses. A subcommand can be given fewer than the number of addresses specified, but not more. A subcommand with the form `[a] command` supports up to one address and a subcommand with the form `[a[,b]] command` supports up to two addresses. All other subcommands do not support any addresses.

- `[a]a` Appends subsequent text lines from the script to the standard output. sed writes the text after completing all other script operations for that line and before reading the next record. Text lines are ended by the first line that does not end with a backslash (\). sed does not treat the \ characters on the end of lines as part of the text.

- `[a[,b]]b [label]` Branches to :`label`. If you omit `label`, sed branches to the end of the script.

- `[a[,b]]c\` Changes the addressed lines by deleting the contents of the pattern buffer (input line) and sending subsequent text (similar to the `a` command) to the standard output. When you specify two addresses, sed delays text output until the final line in the range of addresses; otherwise, the behavior would surprise many users. The rest of the script is skipped for each addressed line except the last.

- `[a[,b]]d` Deletes the contents of the pattern buffer (input line) and restarts the script with the next input line.
[a[,b]]D
Deletes the pattern buffer only up to and including the first newline. Then it restarts the script from the beginning and applies it to the text left in the pattern buffer.

[a[,b]]g
Grabs a copy of the text in the hold buffer and places it in the pattern buffer, overwriting the original contents.

[a[,b]]G
Grabs a copy of the text in the hold buffer and appends it to the end of the pattern buffer after appending a newline.

[a[,b]]h
Holds a copy of the text in the pattern buffer by placing it in the hold buffer, overwriting its original contents.

[a[,b]]H
Holds a copy of the text in the pattern buffer by appending it to the end of the hold buffer after appending a newline.

[a]i\n
Inserts text. This subcommand is similar to the a subcommand, except that its text is output immediately.

[a[,b]]l
Lists the pattern buffer (input line) to the standard output so that nonprintable characters are visible. The end-of-line is represented by $, and the characters \a, \b, \f, \r, \t, and \v are printed as escape sequences. Each byte of a nonprintable double-byte character appears as an escape sequence or as a 3-digit octal number. This subcommand is analogous to the l subcommand in ed.

sed folds long lines to suit the output device, indicating the point of folding with a backslash (\).

[a[,b]]n
Prints the pattern space on standard output if the default printing of the pattern space is not suppressed (because of the -n option). The next line of input is then read, and the processing of the line continues from the location of the n command in the script.

[a[,b]]N
Appends the next line of input to the end of the pattern buffer, using a new line to separate the appended material from the original. The current line number changes.

[a[,b]]p
Prints the text in the pattern buffer to the standard output. The -n option does not disable this form of output. If you do not use -n, the pattern buffer is printed twice.

[a[,b]]P
Operates like the p subcommand, except that it prints the text in the pattern buffer only up to and including the first newline character.

[a]q
Quits sed, skipping the rest of the script and reading no more input lines.

[a]x file
Reads text from file and writes it to the standard output before reading the next input line. The text conversion specified for the sed command (for example, the -B and -W option) is used. The timing of this operation is the same as for the a subcommand. If file does not exist or cannot be read, sed treats it as an empty file.

[a[,b]]s/reg/ sub/[gpn] [wfile]
Substitutes the new text string sub for text matching the regular expression, reg. Normally, the s subcommand replaces only the first such matching string in each input line. You can use any single printable character other than space or newline instead of the slash (/) to delimit reg and sub. The delimiter itself may appear as a literal character in reg or sub if you precede it with a backslash (\). You can omit the trailing delimiter.
If an ampersand (&) appears in sub, sed replaces it with the string matching reg. A \n in reg matches an embedded newline in the pattern buffer (resulting, for example, from an N subcommand). The subcommand can be followed by a combination of the following:

\n\n**n**
Substitutes only the nth occurrence of regexp.

**g**
Replaces all non-overlapping occurrences of regexp rather than the default first occurrence. If both g and n are specified, the last one specified takes precedence.

**p**
Executes the print (p) subcommand only if a successful substitution occurs.

**w file**
Writes the contents of the pattern buffer to the end of file, if a substitution occurs. The text conversion that is specified for the sed command (for example, the -B and -W option) is used. When the variable _UNIX03=YES is set, the file must be preceded with one or more <blank>s. When the variable _UNIX03 is unset or is not set to YES, zero <blank> separation between w and file is allowed.

[a[,b]]t [label]
Branches to the indicated label if a successful substitution occurred since either reading the last input line or running the last t subcommand. If you do not specify label, sed branches to the end of the script.

[a[,b]]w file
Writes the text in the pattern buffer to the end of file. The text conversion specified for the sed command (for example, the -B and -W) is used.

[a [, b ] ] x
Exchanges the text in the hold buffer with that in the pattern buffer.

[a[,b]]y/set1/set2/
Transliterates any input character occurring in set1 to the corresponding element of set2. The sets must be the same length. You can use any character other than backslash or newline instead of the slash to delimit the strings.

If the variable _UNIX03=YES is set and a backslash followed by an 'n' appear in set1 or set2, the two characters are handled as a single newline character. If the variable _UNIX03 is unset or is not set to YES, the two characters are handled as a single character 'n'.

If the delimiter is not n, within set1 and set2, the delimiter itself can be used as a literal character if it is preceded by a backslash. If a backslash character is immediately followed by a backslash character in set1 or set2, the two backslash characters are counted as a single literal backslash character.

[a[,b]]{}
Groups all commands until the next matching } subcommand, so that sed runs the entire group only if the { subcommand is selected by its addresses.

:label
Designates a label, which can be the destination of a bor t subcommand.

#
Treats the script line as a comment unless it is the first line in the script. Including the first line in a script as #n is equivalent to specifying -n on the command line. An empty script line is also treated as a comment.

[a]=
Writes the decimal value of the current line number to the standard output.

**Examples**
1. This filter switches desserts in a menu:

```bash
sed 's/cake\(ic\)*/cookies/g'
```
2. To substitute a pattern in a text file that contains ASCII characters, using the sed stream-oriented text editor and assuming that
   • The text file is untagged and you do not want to tag it or enable automatic conversion, and
   • You cannot alter the tag (for example, you are processing an untagged public text file or a read-only text file)
   then issue:

   ```bash
   sed -W filecodeset=819,pgmcodeset=1047 's/pattern1/pattern2/w myOutFile'
   myAsciiFile
   ```

3. To substitute a pattern in a text file using the sed stream-oriented text editor, assuming that automatic conversion was enabled but the text file is incorrectly tagged as UTF-8:

   ```bash
   sed -B 's/pattern1/pattern2/w myOutputFile' myMisTaggedFile
   ```

Environment variables
sed uses the following environment variables:

COLUMNS
Contains the width of the screen in columns. If set, sed uses this value to fold long lines on output. Otherwise, sed uses a default screen width of 80.

TEXT_CONV
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

UNIX03
For more information about the effect of UNIX03 on this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Localization
sed uses the following localization environment variables:

• LANG
• LC_ALL
• LC_COLLATE
• LC_CTYPE
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH


Exit values

0
   Successful completion

1
   Failure due to any of the following:
   • Missing script
   • Too many script arguments
   • Too few arguments
Messages

Possible error messages include:

badly formed filename for command command
The given subcommand required a file name, but its operand did not have the syntax of a file name.

subcommand command needs a label
The specified subcommand required a label, but you did not supply one.

must have at least one (noncomment) command
The input to sed must contain at least one active subcommand (that is, a subcommand that is not a comment).

No remembered regular expression
You issued a subcommand that tried to use a remembered regular expression; for example, s//abc. However, there is no remembered regular expression yet. Change the subcommand to use an explicit regular expression.
set

- Set or unset command options and positional parameters

**Format**

```
set [±abCefhiKkLmnPpstuvx-] [±o[flag]] [±Aname][parameter ...]
```

tcsh shell:

```
set [-r] set [-r] name ....
set [-r] name=word ....
set [-r] [-f|-l] name=(wordlist) ....
set name[index]=word ...
```

**Description**

Calling `set` without arguments displays the names and values of all shell variables, sorted by name, in the following format:

```
Variable="value"
```

The quoting allows the output to be reinput to the shell using the built-in command `eval`. Arguments of the form `~option` set each shell flag specified as an option. Similarly, arguments of the form `+option` turn off each of the shell flags specified as an option. (Contrary to what you might expect, `~` means on, and `+` means off.)

**Note:** All of the `set` options except ±A, -s, ~, and -- are shell flags. Shell flags can also be set on the `sh` command line at invocation.

**In the tcsh shell**

See the Format section to view the following forms:

1. The first form of the command prints the value of all shell variables. Variables which contain more than a single word print as a parenthesized word list.

   Variables that are read-only will only be displayed by using the `-r` option. For forms 2, 3 and 4, if `-r` is specified, the value is set to read-only.

2. The second form sets `name` to the null string.

3. The third form sets `name` to the single `word`.

4. The fourth form sets `name` to the list of words in `wordlist`. In all cases the value is command and file name expanded. If `-f` or `-l` is specified, `set` only unique words keeping their order. `-f` prefers the first occurrence of a word, and `-l` the last.

5. The fifth form sets the `index`'th component of `name` to `word`; this component must already exist.
These arguments can be repeated to either set or make read-only multiple variables in a single set command. However, variable expansion happens for all arguments before any setting occurs. Also, = can be next to both name and word or separated from both by white space, but cannot be next to only one or the other. For example:

```
set -r name=word and set -r name = word
```

are allowed, but

```
set -r name= word and set -r name =word
```

are not allowed.

For more information, see “tcsh: Invoke a C shell” on page 671.

**Options**

- **-a**
  Sets all subsequently defined variables for export.

- **-b**
  Notifies you when background jobs finish running.

- **-c**
  Prevents the output redirection operator > from overwriting an existing file. Use the alternate operator > | to force an overwrite.

- **-e**
  Tells a noninteractive shell to execute the ERR trap and then exit. This flag is disabled when reading profiles.

- **-f**
  Disables path name generation.

- **-h**
  Makes all commands use tracked aliases. (For an explanation of tracked aliases, see the Command execution section in sh.)

- **-i**
  Makes the shell interactive.

- **-K**
  Tells the shell to use Korn Shell compatible support of the ((expression)) syntax for arithmetic expressions and trap behavior within shell functions. Korn Shell behavior might conflict with UNIX standard-conforming behavior. For more details, see the let and trap command descriptions.

- **-k**
  Allows assignment parameters anywhere on the command line and still includes them in the environment of the command.

- **-L**
  Makes the shell a login shell. Setting this flag is effective only at shell invocation.

- **-m**
  Runs each background job in a separate process group and reports on each as they complete.

- **-n**
  Tells a noninteractive shell to read commands but not run them.

- **-o flag**
  Sets a shell flag. If you do not specify flag, this option lists all shell flags that are currently set. flag can be one of the following:

  - **allexport**
    Is the same as the –a option.

  - **errexit**
    Is the same as the –e option.
bgnice
   Runs background jobs at a lower priority.

emacs
   Specifies emacs-style inline editor for command entry. See shedit for information about the emacs editing mode.

gmacs
   Specifies gmacs-style inline editor for command entry. See shedit for information about the gmacs editing mode.

ignoreeof
   Tells the shell not to exit when an end-of-file character is entered.

interactive
   Is the same as the –i option.

keyword
   Is the same as the –k option.

korn
   Is the same as the –K option.

logical
   Specifies that cd, pwd, and the PWD variable use logical path names in directories with symbolic links. If this flag is not set, these built-ins and PWD use physical directory path names. For example, assume /usr/spool is a symbolic link to /var/spool, and that it is your current directory. If logical is not set, PWD has the value /var/spool, and cd changes the current directory to /var. If logical is set, PWD has the value /usr/spool and cd changes the current directory to /usr.

login
   Is the same as the –L option of sh.

markdirs
   Adds a trailing slash (/) to filename-generated directories.

monitor
   Is the same as the –m option.

noclobber
   Is the same as the –C option.

noexec
   Is the same as the –n option.

noglob
   Is the same as the –f option.

nolog
   Does not record function definitions in the history file.

notify
   Is the same as the –b option.

nounset
   Is the same as the -u option.

pipecurrent
   Is the same as the -P option.

privileged
   Is the same as the –p option.

trackall
   Is the same as the –h option.

verbose
   Is the same as the -v option.

xtrace
   Is the same as the -x option.
**vi**
Specifies vi-style inline editor. See `shedit` for information about the vi editing mode.

**warnstopped**
Tells the shell to issue a warning, but not to exit, when there are stopped jobs.

**-p**
Disables the processing of $HOME/.profile for a login shell and disables the processing of the script specified by the ENV variable. If /etc/suid_profile exists, sh runs it instead of the ENV script.

**-P**
Runs the last command of a pipeline in the current shell environment.

**-s**
Sorts the positional parameters.

**-t**
Exits after reading and running one command.

**-u**
Tells the shell to issue an error message if an unset parameter is used in a substitution.

**-v**
Prints shell input lines as they are read.

**-x**
Prints commands and their arguments as they run.

**Other options:**

- 
  Turns off the -v and -x options. Also, parameters that follow this option do not set shell flags, but are assigned to positional parameters (see `sh`).

- 
  Specifies that parameters following this option do not set shell flags, but are assigned to positional parameters.

**+A name**
Assigns the parameter list specified on the command line to the array elements of the variable name, starting at name[0]. For example, the following command assigns the values "a", "b", "c" and "d" to the array elements array[0-3]:

```
set +A array a b c d
echo ${array[*]}
a b c d
```

**-A name**
Unsets the variable name and then assigns the parameter list specified on the command line to the array elements of the variable name starting at name[0]. For example, if the variable array contains 4 elements, the following command discards the previous values and assigns the values "x" and "y" to the array elements array[0-1]:

```
set -A array a y
echo ${array[*]}
x y
```

**Usage notes**
`set` is a special built-in shell command.

**Localization**
`set` uses the following localization environment variables:

- LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH

For more information, see Appendix F, “Localization,” on page 965.

Exit values
0
   Successful completion.
1
   Failure due to an incorrect command-line argument.
2
   Failure resulting in a usage message, usually due to a missing argument.

Portability
Several shell flags are extensions of the POSIX standard: bgnice, ignoreeof, keyword, markdirs, monitor, noglob, nolog, privileged, and trackall are extensions of the POSIX standard, along with the shell flags ±A, ±h, ±k, ±p, ±s, and ±t.

Related information
alias, eval, export, sh, shedit, tcsh, trap, typeset

setfacl - Set, remove, and change access control lists (ACLs)

Format

```bash
setfacl [-ahqv] -s entries [path ...]
setfacl [-ahqv] -S file [path ...]
setfacl [-ahqv] -D type [...] [path ...]
setfacl [-ahqv] -m|M|x|X EntryOrFile [...] [path ...]
```

Description

setfacl sets (replaces), modifies, or removes the access control list (ACL) to regular files and directories. It also updates and deletes ACL entries for each file and directory that was specified by path. If path was not specified, then file and directory names are read from standard input (stdin). In this case, the input should give one path name per line.

To issue setfacl, you must be the file owner or have superuser authority (either UID 0 or READ access to SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class).

If you specify stdin ("-")) in place of a file name, you cannot specify it for any of the other options, and you cannot read the target path names from stdin.

The maximum number of ACL entries for a file or directory is regulated by the security product and the physical file system.

The first two forms allow you to set (replace) the entire ACL. The third form allows you to delete an entire extended ACL. The fourth form allows you to delete, add or modify ACL entries. You can specify the m, M, x, and X options on a single command line, but you can only specify each option once.

When you are setting the access ACL, the ACL entries must consist of three required base ACL entries that correspond to the file permission bits. The ACL entries must also consist of zero or more extended ACL entries, which will allow a greater level of granularity when controlling access. The permissions for base entries must be in absolute form.
When you are updating ACL entries, you can specify zero or more base entries. The three required base ACL entry types have the following format:

- `u[ser]::perm`
- `g[roup]::perm`
- `o[ther]::perm`

They correspond to the owner, group, and other fields of the file permission bits.

Extended ACL entries have the following format:

```
[d[efault]: | f[default]:]u[ser]:uid: [+|^]perm
[d[efault]: | f[default]:]g[roup]:gid: [+|^]perm
```

where:
- **d[efault]**
  - If specified, extended ACL refers to directory default ACL.
- **f[default]**
  - If specified, extended ACL refers to file default ACL.
- **u[ser]**
  - Extended ACL refers to a particular numeric user ID (UID) or user name.
- **g[roup]**
  - Extended ACL refers to a particular numeric group ID (GID) or group name.
- **uid**
  - User name or numeric user ID (UID).
- **gid**
  - Group name, or numeric group ID (GID).
- **perm**
  - Permissions specified either in absolute form (string `rwx` with `-` as a placeholder or octal form), or in relative format (using the `+` or `^` modifiers).

For relative permission settings, only one of `+` or `^` is allowed per ACL entry. When using relative permissions, you must have at least one of `r`, `w`, or `x`. For example, `+rw` or `^rwx`.

The first field of an ACL entry is optional; it specifies the type of ACL (access, directory default, or file default) that will be processed. If the type is not specified, the operation applies only to the access ACL. If you are updating the ACL entries, you can specify the base ACL entries; however, specifying the base ACL entries might cause the file or directory’s permission bits to change if what is specified is different than the current settings.

If the permissions are specified in relative format for an ACL entry that does not currently exist, then the permissions will be assigned as though they were given in absolute form. Any permissions that were not specified will default to no permission. For instance, if an extended ACL entry is given as follows to be updated:

```
user:BILLYJC:+rw
```

and user entry BILLYJC does not currently exist, then the resulting entry will be:

```
user:BILLYJC:rw-
```

Similarly, if you try to remove the permissions from an extended ACL entry that does not exist, the resulting permissions will be:

```
---
```

That is, no permission.

For additional information about ACLs and ACL entries, see *z/OS UNIX System Services Planning*. 
Options

-a
   Stops setfacl processing if one of the following errors or warnings occurs:
   1. During the attempt to change an ACL for a file or directory, setfacl performs a stat(), and the
      stat() fails with a unique reason code.
   2. The user tried to change the file default ACL or directory default ACL for a path name that is not a
      directory.
   3. An attempt to delete all extended ACL entries failed for the current path name.
   4. An attempt to set or modify extended ACL entries failed for the current path name.

   When you do not specify -a, the setfacl processing continues.

-D type
   Deletes all extended ACL entries for the ACL of type. For an access ACL, this leaves only the three
   required base entries intact. For a file default or directory default ACL, the entire ACL for the specified
   type is deleted. You can specify type as one of the following:
   a
      Access ACL
   d
      Directory default ACL
   f
      File default ACL
   e
      Every extended ACL for all ACL types that are applicable for the current path name

-h
   Does not follow symbolic links. Because ACLs are not associated with symbolic links, nothing will
   happen if a symbolic link is encountered.

-m EntryOrFile
   Modifies the ACL entries specified by EntryOrFile. EntryOrFile represents a string of ACL entries typed
   directly on the command line. If an ACL entry does not exist for a user or group that is specified in
   EntryOrFile, then it is created. If an ACL entry already exists for a user or group that was specified in
   EntryOrFile, then it is replaced.

   The specified entries must be unique for each ACL type and its associated user or group combinations.

-M EntryOrFile
   Modifies the ACL entries specified in EntryOrFile. EntryOrFile represents a file containing ACL entries.
   If an ACL entry does not exist for a user or group that is specified in EntryOrFile, then it is created. If an
   ACL entry already exists for a user or group that was specified in EntryOrFile, then it is replaced. If
   EntryOrFile is -, then entries are read from stdin.

   The specified entries must be unique for each ACL type and its associated user or group combinations.

-q
   Quiet mode. setfacl will suppress all warning and error messages for the following conditions:
   • During the attempt to change an ACL for a file or directory, setfacl performs a stat(), and the stat() fails with a unique reason code.
   • The user tried to change the file default ACL or directory default ACL for a path name that is not a directory.
   • The user tried to use setfacl on a non-regular file or directory.

   The condition that caused the warning or error will not affect the return code.

-s entries
   Sets (replaces) all ACLs with entries.
-S file
Sets (replaces) all ACLs with the entries specified in file. If file is -, then entries are read from stdin.

-v
Verbose

-x EntryOrFile
Deletes the extended ACL entries specified by EntryOrFile. EntryOrFile is a string of ACL entries typed directly on the command line. If an ACL entry does not exist for the user or group specified, then you will not get an error. If the permissions field is provided in EntryOrFile, then it is ignored when this option is processed. Users cannot delete the base ACL entries (file owner, owning group, and others). If base ACL entries are specified with this option, they are ignored. Deleting an extended ACL entry does not necessarily have the same effect as removing all the permissions from an entry.

-X EntryOrFile
Deletes the extended ACL entries specified by EntryOrFile. EntryOrFile is a file containing ACL entries. If an ACL entry does not exist for the user or group specified, then you will not get an error. If EntryOrFile is -, then entries are read from stdin. If the permissions field is provided in EntryOrFile, then it is ignored when this option is processed. Users cannot delete the base ACL entries (file owner, owning group, and others). If base ACL entries are specified with this option, they are ignored. Deleting an extended ACL entry does not necessarily have the same effect as removing all the permissions from an entry.

Examples
1. To set (replace) the current access ACL for file foo, giving only user Billy read and execute access:
   ```
   setfacl -s user::rwx,group::---,other::---,user:billy:r-x foo
   ```
   This might change the permission bits of the file.

2. To modify the current access ACL for file foo to contain an extended ACL entry for group cartoons, giving that group read access:
   ```
   setfacl -m group:cartoons:+r foo
   ```

3. To set (replace) the current access and directory default ACLs for directory Haunted so that users user1 and user2 have read and search permissions, while the group thegang has read permissions:
   ```
   setfacl -s "u::rwx,g::---,d::---,user:user1:r-x,group:thegang:r--,user:user2:r-x, \
   d:user:user1:r-x,d:group:thegang:r--,d:user:user2:r-x" Haunted
   ```

4. To copy the ACL from file foo such that the file bar will have the same ACL:
   ```
   getfacl foo | setfacl -S - bar
   ```

5. To delete all of the extended ACL entries for user user3 for all files and directories in the current directory:
   ```
   setfacl -x user:user3,d:user:user3,f:user:user3 *
   ```

6. To delete all of the extended ACL entries for all files and directories in the current working directory:
   ```
   setfacl -D e *
   ```

7. To change a directory’s access ACL so that user1 has read, write, and execute access for all files in the Haunted directory:
   ```
   setfacl -m user:user1:rwx Haunted
   ```
8. RACF recommends placing ACLs on directories, rather than on each file in a directory. To find and remove all of the extended ACL entries for user1 that are associated with only the files in directory Haunted:

```
setfacl -x user:user1 $(find Haunted -type f -acl_user user1)
```

Even if the `setfacl` command is successful in removing access from `user1`, `user1` might still be able to obtain access to the files in directory Haunted based on the file permission bits, assuming the user has search permission for Haunted.

**Localization**

`setfacl` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_SYNTAX
- NLSPATH


**Usage notes**

1. When you use `setfacl` to add, change and delete ACL entries, all deletion operations are performed first. In other words, deletion operations are processed before any change or add operations.
2. `setfacl` should not be considered an atomic operation because if multiple operations are requested and an error occurs, some of the operations might have been processed before the error was encountered. Note that the `-s` option is considered a multiple operation request.
3. `setfacl` only supports directories and regular files.

**Exit values**

0  
Success.

1  
Failure due to any of the following reasons:
- Incorrect command-line option.
- Too few arguments on the command line.
- An attempt was made to read from stdin in more than one place.
- An attempt was made to combine `setfacl` operations that are mutually exclusive.

2  
Failure due to any of the following reasons:
- A specified path name does not exist.
- An error occurred while attempting to read the entries file.
- An attempt was made to alter the file default ACL or directory default ACL for a path name that is not a directory.
- The specified path was not a regular file or directory.

3  
Failure due to any of the following reasons:
- Unable to delete all extended ACL entries from a path name.
- Unable to set or modify ACL entries for a path name.
• Improper syntax of ACL entries.
• An incorrect ACL was specified.
• Unable to allocate enough memory.
• Unable to determine the PATH_MAX.
• Unable to open the entries file for reading.
• The entries file is empty.

**Portability**

An approved POSIX standard does not exist for `setfacl`.

**Related information**

`chmod`, `find`, `getfacl`, `ls`, `filetest`, `pax`, `test`

---

**sh: Invoke a shell**

**Format**

```
[r]sh [±abCefhiKkLmnPprtuvx] [±o option] [cmd_file [argument ...]]
[r]sh -S [±abCefhiKkLmnPprtuvx] [±o option] [cmd_file [argument ...]]
[r]sh -c cmdstring [±abCefhiKkLmnPprtuvx] [±o option] [cmd_name [argument ...]]
[r]sh -s [±abCefhiKkLmnPprtuvx] [±o option] [argument ...]
```

**Description**

`sh` contains the following topics and subtopics:

• Options and invocations
• Options
• Command syntax
• Reserved word commands
• Command execution
• Quoting
• Directory substitution
• Parameter substitution
• Arithmetic substitution
• File description and redirection
• File name generation
• Variables
• Shell variables
• Shell variables for automatic conversion
• Shell execution environments
• Built-in commands
• Shell archives
• Files
• Localization
• Exit values
• Messages
• Limits
• Portability
• Related information

Subtopics dealing with substitution and interpretation of input appear in the order in which the shell performs those substitutions and interpretations.

Much of what the shell can do is provided through such built-in commands as cd and alias.

**Restriction:** If the tagged script is being run with automatic conversion enabled, the code page of the locale must be SBCS and the script must also be encoded in SBCS.

**Options and invocation**

The z/OS shell is upward-compatible with the Bourne shell.

Typically, you launch the shell by logging in. You can also invoke the shell by typing an explicit `sh` command. Some people find it useful to copy the `sh` file into a file named `rsh`. If you invoke the shell under the name `rsh`, the shell operates in restricted mode. This mode is described with `-r`.

If you invoke the shell with a name that begins with the `-` character, it is a *login shell*. (You can also get a login shell if you invoke the shell with the `-L` option.) A login shell begins by running the file `/etc/profile`. It then runs `$HOME/.profile` using the `.` command (see `dot`). If HOME is not set, the shell searches the working directory for:

```
.profile
```

and runs this file with the `.` command if it exists. You do not get an error message if any of these files cannot be found.

You can use these profile files to customize your session with `sh`. For example, your profile files can set options, create aliases, or define functions and variables.

If there is at least one argument on the `sh` command line, `sh` takes the first argument as the name of a shell script to run. (The exception to this is when `-s` is used.) Any additional arguments are assigned to the positional parameters; usually, these serve as arguments to the shell script. See “Parameter substitution” on page 598 for information about positional parameters. Also see `set` for information about changing these parameters.

If `sh` finds the ENV environment variable set when it begins running (after profile processing), `sh` runs the file named by the expansion of the value of this variable.

**Options**

The shell accepts the following options on the command line:

- **-c cmdstring**
  Runs `cmdstring` as if it were an input line to the shell and then exits. This is used by programs (for example, editors) that call the shell for a single command. `sh` assigns arguments after `cmdstring` to the positional parameters. If you specify `cmd_name`, special parameter 0 is set to this string for use when running the commands in `cmdstring`.

- **-i**
  Invokes an interactive shell, as opposed to running a script. With `-i`, the shell catches and ignores keyboard interrupts. Without `-i`, an interrupt ends the shell. For shells that read from the terminal, `-i` is the default.

- **-L**
  Makes the shell a *login shell*. (A login shell is an interactive shell.)

- **-r**
  Invokes a restricted shell. (You can also invoke a restricted shell by using the name `rsh`). In a restricted shell, you cannot do the following:
• Use the cd command.
• Change the values of the variables ENV, PATH, or SHELL.
• Use > or >> to redirect output; or specify command names containing /.

These restrictions do not apply during execution of your profile files.

-S
Reads commands from standard input and assigns all arguments to the positional parameters.
Normally, if there is at least one argument to the shell, the first such argument is the name of a file to be run.

-s
Searches the directories in the environment variable PATH for a file cmd_file that contains shell commands. The current working directory is not searched before PATH.

If you do not give either the -c> or -s> option, but you do specify cmd_file, the shell takes it as the name of a file that contains commands to be run. Special parameter 0 is set to this name.

If cmd_file contains a slash, the shell attempts to read that file name. If cmd_file does not contain a slash, the following can occur:
• If -s> is specified, the shell searches for the file in PATH. Only a file with executable access permissions for the user will be found in the PATH search.
• If -s> is not specified, the shell searches for the file in the current working directory, then in PATH. The file must have read access that is permitted for the user. Executable access permission is not necessary.

In addition to these options, you can use any valid option to the set command (including -o> option) as a command-line option to sh. See set for details.

Command syntax

The shell implements a sophisticated programming language that gives you complete control over the execution and combination of individual commands. When the shell scans its input, it always treats the following characters specially:

```
; & ( ) < > | ' 
```

If you want to use any of these characters inside an actual argument, you must quote the argument (so that the shell does not use the special meanings of the characters). See Quoting for more information.

A simple command is a list of arguments separated by characters in the IFS environment variable (the default value of IFS has blank, tabs, and newlines).

When a word is preceded by an unescaped number sign (#), the remainder of the line is treated as a comment, and the shell discards input up to but not including the next newline. When a command starts with a defined alias, sh replaces the alias with its definition (see alias).

A reserved-word command starts with a reserved word (for example, if, while, or for). Reserved-word commands provide flow of control operations for the shell and are listed in “Reserved-word commands” on page 592.

A command can be any of the following:

command:

• simple command
• reserved-word command
• (command)
• command | command
• command &command
The following is the order of precedence of the preceding operators. The highest priority operators are listed first, and operators on the same line have equal priority.

```
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |

The meaning of these operations is as follows:

**command**

- Runs `command` in a child shell. The current shell invokes a second shell, and this second shell actually runs `command`. In this way, `command` runs in a completely separate execution environment; it can change working directories, change variables, open files, and so on without affecting the first shell. The child shell’s environment begins as a copy of the current environment, so the value of the ENV environment variable is not run when a child shell starts.

**|**

- Creates a pipe between the two `command`s that the `|` operator connects. The standard output of the first `command` becomes the standard input of the second `command`. A series of commands connected by pipes is called a pipeline. The exit status is that of the last command in the pipeline.

**&&**

- Is the logical AND operator. The shell runs the second `command` if and only if the first `command` returns a true (zero) exit status.

**||**

- This is the logical OR operator. The shell runs the second `command` if and only if the first `command` returns a false (nonzero) exit status.

**&**

- Runs the `command` that precedes it asynchronously. The shell just starts the `command` running and then immediately goes on to take new input, before the `command` finishes execution. On systems where asynchronous execution is not possible, this operation is effectively equivalent to `;`.

**|&**

- Runs the `command` that precedes it as a co-process. The `command` runs asynchronously, as with the & operator, but the command’s standard input and standard output are connected to the shell by pipes. The shell sends input to `command`’s standard input with the `print -p` command, and reads from `command`’s standard output with the `read -p` command. The `command` should not buffer its output. Because of this and other limitations, coprocesses should be designed to be used as coprocesses. On systems where asynchronous execution is not possible, coprocesses are not supported.

**;**

- Is the sequential execution operator. The second `command` is run only after the first `command` has completed.

**newline**

- The unescaped newline is equivalent to the `;` operator.
Reserved-word commands

The shell contains a rich set of reserved-word commands, which provide flow of control and let you create compound commands. In the following list, a command can also be a sequence of commands separated by newlines. Square brackets ( [ ] ) indicate optional portions of commands, and are included as part of the command syntax except in the case of [ [ test_expr ] ], where square brackets are part of the command.

! The exclamation point is the logical NOT command. When its operand is false (nonzero), this command returns true (zero). When its operand is true (zero), this command returns false (nonzero).

{command;} Enclosing a command in braces is similar to the (command) construct, except that the shell runs the command in the same environment rather than under a child shell. { and } are reserved words to the shell. To make it possible for the shell to recognize these symbols, you must put a blank or newline after the {, and a semicolon or newline before the }.

[ [ test_expr ] ] The double-square-bracket command ( [ [ test_expr ] ] ) is a command that returns an exit status indicating whether the test_expr (test expression) is true or false.

Word-splitting and wildcard expansion (file name expansion or globbing) are not done within [ [ ] ]. This makes quoting less necessary than when you use the test (or [ ] ) command. Alias expansion is also not done within [ [ ] ].

The following primitives are used in ( [ [ test_expr ] ] ). Spaces or tabs are required to separate operators from operands.

-;a file True if file exists (-e is recommended to avoid confusion with the test command syntax)

-Aa file True if file has an extended access ACL entry.

-Ad file True if file has a directory default ACL.

-Af file True if file has a file default ACL.

-b file True if file is a block special file (block special files are not supported in z/OS)

-b file True if the file is tagged as binary (not text)

-c file True if file is a character special file

-d file True if file is a directory

-e file True if file exists

-Ea file True if the file has the APF extended attribute

-El file True if the file has the shared library extended attribute

-Ep file True if the file has the program control extended attribute

-Es file True if the file has the shared address space extended attribute

-f file True if file is an ordinary file
-g file
   True if the set-group-ID attribute of file is on
-G file
   True if file group owner is the effective group ID
-h file
   True if file is a symbolic link
-k file
   True if file has the "sticky" bit on
-L file
   True if file is a symbolic link
-Ma file
   True if the file has a security label
-n string
   True if the length of the string is greater than zero
-o option
   True if shell option is on
-o file
   True if file owner is the effective user ID
-p file
   True if file is a FIFO (named pipe)
-x file
   True if file is readable (checks permission bits and access control)
-s file
   True if size of the file is nonzero
-S file
   True if file is a socket
-t fd
   True if the numeric file descriptor fd is open and associated with a terminal
-T file
   True if the file is tagged as text
-u file
   True if the set-user-ID attribute of file is on
-w file
   True if file is writable (checks permission bits and access control)
-x file
   True if file is executable (checks permission bits and access control)
-z string
   True if length of the string is zero
string
   True if string is not a null string
string = pattern
   True if string matches pattern (== is recommended to avoid confusion with the test command syntax)
string == pattern
   True if string matches pattern. Quote pattern to treat it as a string. For information about patterns, see “File name generation” on page 604.
string1 != pattern
   True if string does not match patterns. For information about patterns, see “File name generation” on page 604.
string1 < string2
True if string1 comes before string2 in the collation order defined in the current locale

string1 > string2
True if string1 comes after string2 in the collation order defined in the current locale

exp1 -eq exp2
True if arithmetic expression exp1 and exp2 are equal

exp1 -ge exp2
True if arithmetic expression exp1 is greater than or equal to exp2

exp1 -gt exp2
True if arithmetic expression exp1 is greater than exp2

exp1 -le exp2
True if arithmetic expression exp1 is less than or equal to exp2

exp1 -lt exp2
True if arithmetic expression exp1 is less than exp2

exp1 -ne exp2
True if arithmetic expression exp1 is not equal to exp2

file1 -nt file2
True if file1 is newer than file2

file1 -ot file2
True if file1 is older than file2

file1 -ef file2
True if file1 is a hard link or symbolic link to file2 (this is different than the test command which only tests for hard links on z/OS)

file-CS codeset
True if the file is tagged with the codeset

file -Ml seclabel
True if the file has a security label. False if the file does not have a security label that matches the specified seclabel.

( test_expr )
Grouping to override normal precedence; true if test_expr is true

! test_expr
Logical negation; true if test_expr is false

test_expr1 && test_expr2
Logical AND; true if both test_expr1 and test_expr2 are true

test_expr || test_expr2
Logical OR; true if either test_expr1 or test_expr2 is true

Patterns tested in double-square-bracket conditions are composed of special characters and regular characters. Patterns follow the rules given in “File name generation” on page 604, except that the period (.) and the slash (/) are not treated specially. Note that pattern matching is similar to regular expression processing, but different in syntax.

case word in [()pattern[|pattern] &... ) command ; ; ]... [()pattern[| pattern] ... )command ; ; ]...

esac
The case statement is similar to the switch statement of the C programming language or the case statement of Pascal. If the given word matches any one of the patterns separated by “or” bar (|) characters, sh runs the corresponding command. The patterns should follow the rules given in “File name generation” on page 604, except that the period (.) and slash (/) are not treated specially. Patterns are matched in the order they are given, so more inclusive patterns should be mentioned later. You must use the double semicolon (;;) to delimit command and introduce the next pattern.
for variable [in word ...] do command done
The for statement sets variable to each word argument in turn, and runs the set of commands once for each setting of variable. If you omit the in word part, sh sets variable to each positional parameter. You can divert the flow of control within the loop with the break or continue statements.

function variable { command ... } ... variable() { command ... }
Either one of these forms defines a function named variable, the body of which consists of the sequence of commands. You invoke a function just like any other command; when you actually call the function, sh saves the current positional parameters. The function's command-line arguments then replaces these parameters until the function finishes. sh also saves the current ERR and EXIT traps, as well as any flags manipulated by EXIT with the set command; these are restored when the function finishes. The function ends either by falling off the end of the code of the function body, or by reaching a return statement. If the function uses typeset to declare any variables in the function body, the variables are local to the function.

if command then command [elif command then command] ... [else command] fi
In the if statement, if the first (leftmost) command succeeds (returns a zero exit status), sh runs the command following then. Otherwise, sh runs the command (if any) following the elif (which is short for “else if”); if that succeeds, sh runs the command following the next then. If neither case succeeds, sh runs the command following the else (if any).

select variable [in word ...] do commands done
The select statement can handle menu-like interactions with the user. Its syntax is like the for statement. Each word is printed on the standard error file, one per line, with an accompanying number. If you omit the “in word ...” part, sh uses the positional parameters. sh then displays the value of the variable PS3 to prompt the user to enter a numerical reply. If the reply is an empty line, sh displays the menu again; otherwise, sh assigns the input line to the variable REPLY, sets variable to the word selected, and then runs the commands. sh does this over and over until the loop is ended by an interrupt, an end-of-file, or an explicit break statement in the commands.

until command1 do command2 done
The until statement runs command1 and tests its exit status for success (zero) or failure (nonzero). If command1 succeeds, the loop ends; otherwise, sh runs command2 and then goes back to run and test command1 again. break and continue commands in the commands can affect the operation of the loop.

while command1 do command2 done
The while statement works similarly to the until statement. However, the loop ends whenever command1 is unsuccessful (nonzero exit status).

Shell reserved words are recognized only when they are the unquoted first token of a command. This lets you pass these reserved words as arguments to commands run from the shell. The full list of reserved words is:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>done</td>
<td>function</td>
<td>while</td>
</tr>
<tr>
<td>[[]</td>
<td>elif</td>
<td>if</td>
<td></td>
</tr>
<tr>
<td>else</td>
<td>select</td>
<td></td>
<td></td>
</tr>
<tr>
<td>esac</td>
<td>then</td>
<td></td>
<td></td>
</tr>
<tr>
<td>case</td>
<td>fi</td>
<td>time</td>
<td></td>
</tr>
<tr>
<td>do</td>
<td>for</td>
<td>until</td>
<td></td>
</tr>
</tbody>
</table>

Command execution

Before running a simple command, the shell processes the command line, performing expansion, assignments, and redirection.

First, sh examines the command line and divides it into a series of tokens, which are either operators or words. An operator is either a control operator, which is described in “Command syntax” on page 590. Or it can be a redirection operator, described in “File descriptors and redirection” on page 603. A word is any token that is not an operator.
Next, the shell expands words in the following order:

1. **sh** performs directory substitution.
2. **sh** performs parameter substitution, command substitution, or arithmetic substitution, as appropriate, in the order that the words appear on the command line, expanding each word to a *field* (see the appropriate topics).
3. **sh** scans each field produced in step 2 for unquoted characters from the IFS environment variable and further subdivides this field into one or more new fields.
4. **sh** expands any aliases to their definitions.
5. **sh** performs path name expansion on each unquoted field from step 3.
6. **sh** removes all quote mechanisms (\, ' and ") that were present in the original word unless they have themselves been quoted.

The shell considers the first field of the expanded result to be a command.

The expanded simple command can contain variable assignments and redirections. Variable assignments affect the current execution environment. After expansion, the shell handles all redirection constructs, and the command, if one was found, it performs the redirection in a child shell environment (see “Shell execution environments” on page 611).

When a simple command contains a command name, variable assignments in the command affect only the execution of that command.

After the shell has expanded all appropriate arguments in a simple command, but before it performs file name generation, it examines the command name (if the command has one). **sh** first checks the names against currently defined aliases (see the `alias` command) and functions (see `function` in “Reserved-word commands” on page 592), and finally against the set of built-in commands: commands that the shell can run directly without searching for program files.

The `autoload` command, an alias of `typeset -fu`, identifies functions that are not yet defined. The first time an undefined function is called within the shell, the shell will search directories in the FPATH shell variable for a file with the same name as the function. If a matching file is found, it is assumed to contain the function definition of the same name. The file is read and executed in the current shell environment, storing the function in the shell's memory for subsequent execution. (Multiple function definitions may be contained in the same file. When the file is processed by the shell, all the functions will be defined. Every function definition in the file should be a link name to the file.)

If the command is a built-in or function, the shell executes it.

If the command name is not a function or a built-in command, the z/OS shell looks for a program file or script file that contains an executable version of that command. The shell uses the following procedure to locate the program file:

- If the command name typed to the shell has slash (/) characters in its name, the command is taken to be a full path name (absolute or relative). The shell tries to execute the contents of that file.
- Otherwise, the shell performs a path search. To do this, the shell obtains the value of the PATH environment variable. The value should be a list of directory names. **sh** searches under each directory for a file, the name of which matches the command name. If the FPATH shell variable is set, the shell will search the PATH and FPATH directories. If a file with a name matching the command name is found in the same directory in both PATH and FPATH, or if a matching file is found only in FPATH, this file will be read and executed in the current shell environment (defining the functions contained in the file). The shell will then execute the function matching the command name. This allows users to use FPATH for locating functions without the need to identify every function with the `autoload` command.

If FPATH is not set, or if the command is not found in FPATH, the shell executes the first matching file found in the PATH directories. For more information about specifying the PATH variable, see the topic in *z/OS UNIX System Services User’s Guide* on using the PATH variable when customizing the search path for commands.
Command names can be marked as tracked aliases. The first time you run a command with a tracked alias, the shell does a normal PATH search. If the search is successful, the shell remembers the file that it finds. The next time you run a command with the same name, sh immediately runs the file found on the last PATH search; there is no new search. This speeds up the time that it takes the shell to find the appropriate file.

The set -h command tells the shell that all commands should be treated as tracked aliases. See alias and set for more information.

**Quoting**

To let you override the special meaning of certain words or special characters, the shell provides several quoting mechanisms. In general, you can turn off the special meaning of any character by putting a backslash (\) in front of the character. This is called escaping the character.

For example, you can tell the shell to disregard the special meaning of the newline character by putting a backslash at the very end of a line. The shell ignores the escaped newline, and joins the next line of input to the end of the current line. In this way, you can enter long lines in a convenient and readable fashion.

Escaping characters by putting a backslash in front of them is the most direct way of telling the shell to disregard special meanings. However, it can be awkward and confusing if you have several characters to escape.

As an alternative, you can put arguments in various types of quotation marks. Different quotation mark characters have different “strengths.” The single quotation marks are the strongest. When you enclose a command-line argument in single quotation marks, the shell disregards the special meanings of everything inside the single quotation mark. For example:

```
echo 'a'
```

Double quotation marks are weaker. Inside double quotation marks, the shell performs command substitutions (see “Command substitution” on page 602), parameter substitutions (see “Parameter substitution” on page 598) and arithmetic substitutions (see “Arithmetic substitution” on page 600). The shell does not perform such substitutions when they appear inside single quotation marks. You can use the backslash to escape another character when they appear inside double quotation marks, but inside single quotation marks the shell ignores this special meaning.

The shell treats internal field separator characters (that is, characters in the value of the IFS variable) literally inside quoted arguments, whether they’re quoted with double quotation marks or single quotation marks. This means that a quoted argument is considered a single entity, even if it contains IFS characters.

Quoting can override the special meanings of reserved words and aliases. For example, in:

```
"time" program
```

the quotes around time tell the shell not to interpret time as a shell reserved word. Instead, sh does a normal command search for a command named time.

You must always quote the following characters if you want sh to interpret them literally:

```
| & ; < > ( ) $ ' " \ 

<space> <tab> <newline>
```

The following characters need to be quoted in certain contexts if they are to be interpreted literally:

```
* ? [ # % =
```

~
Directory substitution

When a word begins with an unquoted tilde (~), `sh` tries to perform directory substitution on the word. `sh` obtains all characters from the tilde (~) to the first slash (/) and uses this as a user name. `sh` looks for this name in the user profile, the file that contains information about all the system's users. If `sh` finds a matching name, it replaces ~name with the name of the user's home directory, as given in the matching RACF user profile entry.

For example, if you specify a file name as:

```
~j smith/file
```

`sh` would look up jsmith's home directory and put that directory name in place of the ~j smith construct.

If you specify a ~ without an accompanying name, `sh` replaces the ~ with the current value of your HOME variable. For example:

```
echo ~
```

displays the name of your home directory. Similarly, `sh` replaces the construct ~+ with the value of the PWD variable (the name of the working directory), and replaces the tilde hyphen (~-) with the value of OLDPWD (the name of your previous working directory). In variable assignments, tilde expansion is also performed after colons (:)..

Parameter substitution

The shell uses three types of parameters: positional parameters, special parameters, and variables. A positional parameter is represented with either a single digit (except 0) or one or more digits in braces. For example, 7 and {15} are both valid representations of positional parameters. Positional parameters are assigned values from the command line when you invoke `sh`.

A special parameter is represented with one of the following characters:

```
*    @    #    ?    !    -    $     0
```

The values to which special parameters expand are listed in the following paragraphs.

Variables are named parameters. For details on naming and declaring variables, see “Variables” on page 605.

The simplest way to use a parameter in a command line is to enter a dollar sign ($) followed by the name of the parameter. For example, if you enter the command:

```
echo $x
```

`sh` replaces $x with the value of the parameter x and then displays the results (because `echo` displays its arguments). Other ways to expand parameters are shown in the following paragraphs.

The following parameters are built in to the shell:

```
$1, $2, ... $9
```

Expands to the d positional parameter (where d is the single digit following the $). If there is no such parameter, $d expands to a null string.

```
$0
```

Expands to the name of the shell, the shell script, or a value assigned when you invoked the shell.

```
$#
```

Expands to the number of positional parameters.
$@
Expands to the complete list of positional parameters. If $@ is quoted, the result is separate arguments, each quoted. This means that:

```
"$@
```

is equivalent to:

```
"$1" "$2" ...
```

$*
Expands to the complete list of positional parameters. If $* is quoted, the result is concatenated into a single argument, with parameters separated by the first character of the value of IFS (see “Variables” on page 605). For example, if the first character of IFS is a blank, then:

```
"$*
```

is equivalent to:

```
"$1 $2 ...
```

$-
Expands to all options that are in effect from previous calls to the `set` command and from options on the `sh` command line.

$?
Expands to the exit status of the last command run.

$$
Expands to the process ID of the shell. If running in a child shell environment (see “Shell execution environments” on page 611), it is the process ID of the parent shell. Otherwise, it is the process ID of the current shell.

$!
Expands to the process number of the last asynchronous command.

These constructs are called *parameters* of the shell. They include the positional parameters, but are not restricted to the positional parameters.

We have already mentioned that you can expand a parameter by putting a $ in front of the parameter name. More sophisticated ways to expand parameters are:

$\{\text{parameter}\}$
Expands any parameter.

$\{\text{number}\}$
Expands to the positional parameter with the given number. (Remember that if you just enter $d to refer to the $dth positional parameter, $d can only be a single digit; with brace brackets, number can be greater than 9.) Since braces mark the beginning and end of the name, you can have a letter or digit immediately following the expression.

$\{\text{variable}[\text{arithmetic expression}]\}$
Expands to the value of an element in an array named variable. The arithmetic expression gives the subscript of the array. (See “Arithmetic substitution” on page 600.)

$\{\text{variable}[*]\}$
Expands to all the elements in the array variable, separated by the first character of the value of $\text{IFS}$. A

$\{\text{variable}[@]\}$
When unquoted, is the same as $\{\text{variable}[*]\}$ When quoted as “$\{\text{variable}[@]\}$,” it expands to all the elements in the array variable, with each element quoted individually.

$\{\#\text{parameter}\}$
Expands to the number of characters in the value of the given parameter.

$\{\#\}$
Expands to the number of positional parameters.
\$\{# \ast\}
Expands to the number of positional parameters.

\$\{#@\}
Expands to the number of positional parameters.

\$\{\texttt{variable} \[*\]\}
Expands to the number of elements in the array named \texttt{variable}. Elements that do not have assigned values do not count. For example, if you only assign values to elements 0 and 4, the number of elements is 2. Elements 1 through 3 do not count.

\$\{\texttt{parameter-word}\}
Expands to the value of \texttt{parameter} if it is defined and has a nonempty value; otherwise, it expands \texttt{word}. This means that you can use \texttt{word} as a default value if the parameter isn’t defined.

\$\{\texttt{parameter-word}\}
Is similar to the preceding construct, except that the parameter is expanded if defined, even if the value is empty.

\$\{\texttt{variable} = \texttt{word}\}
Expands \texttt{word} with parameter expansion and assigns the result to \texttt{variable}, provided that \texttt{variable} is not defined or has an empty value. The result is the expansion of \texttt{variable}, whether or not \texttt{word} was expanded.

\$\{\texttt{variable}=\texttt{word}\}
Is similar to the preceding construct, except that \texttt{variable} must be undefined (it cannot just be null) for \texttt{word} to be expanded.

\$\{\texttt{parameter} : ? \texttt{word}\}
Expands to the value of \texttt{parameter} provided that it is defined and non-empty. If \texttt{parameter} isn’t defined or is null, \texttt{sh} expands and displays \texttt{word} as a message. If \texttt{word} is empty, \texttt{sh} displays a default message. After a non-interactive shell has displayed a message, it ends.

\$\{\texttt{parameter}\?\texttt{word}\}
Is similar to the preceding construct, except that \texttt{sh} displays \texttt{word} only if \texttt{parameter} is undefined.

\$\{\texttt{parameter} + \texttt{word}\}
Expands to \texttt{word}, provided that \texttt{parameter} is defined and non-empty.

\$\{\texttt{parameter}+\texttt{word}\}
Expands to \texttt{word}, provided that \texttt{parameter} is defined.

\$\{\texttt{parameter}\#\texttt{pattern}\}
Attempts to match \texttt{pattern} against the value of the specified \texttt{parameter}. The \texttt{pattern} is the same as a case \texttt{pattern}. \texttt{sh} searches for the shortest prefix of the value of \texttt{parameter} that matches \texttt{pattern}. If \texttt{sh} finds no match, the previous construct expands to the value of \texttt{parameter}; otherwise, the portion of the value that matched \texttt{pattern} is deleted from the expansion.

\$\{\texttt{parameter}\##\texttt{pattern}\}
Is similar to the preceding construct, except that \texttt{sh} deletes the longest part that matches \texttt{pattern} if it finds such a match.

\$\{\texttt{parameter}\%\texttt{pattern}\}
Searches for the shortest suffix of the value of \texttt{parameter} matching \texttt{pattern} and deletes the matching string from the expansion.

\$\{\texttt{parameter}\%%\texttt{pattern}\}
Is similar to the preceding construct, except that \texttt{sh} deletes the longest part that matches \texttt{pattern} if it finds such a match.

**Arithmetic substitution**
Arithmetic substitution is available with the syntax:

\$((\texttt{arithmetic expression}))
or:

```
$[arithmetic expression]
```

This sequence is replaced with the value of arithmetic expression. Arithmetic expressions consist of expanded variables, numeric constants, and operators. Numeric constants have the forms:

- A number that starts with 0x is hexadecimal.
- A number that starts with 0 is octal.
- A number that does not start with 0x or 0 is decimal.
- base #number, where base is a decimal integer between 2 and 36 inclusive, and number is any nonnegative number in the given base.

Undefined variables evaluate to zero.

If the shell variable x contains a value that forms a valid integer constant, then the arithmetic expansions "$((x))" and "$((x))" or $[x] or $[$x] return the same value.

The following operators are listed in decreasing order of precedence. Operators sharing a heading have the same precedence. Evaluation within a precedence group is from left to right, except for the assignment operator, which evaluates from right to left.

**Unary operators:**

- `-`
  Unary minus

- `!`
  Logical negation

- `+ ~`
  Identity, bitwise negation

**Multiplicative operators:**

- `* / %`
  Multiplication, division, remainder

**Additive operators:**

- `+ -`
  Addition, subtraction

**Bitwise Shift Operators:**

- `<< >>`
  Bitwise shift right, bitwise shift left

**Relational Operators:**

- `< >`
  Less than, greater than

- `<= >=`
  Less than or equal, greater than or equal

- `= !=`
  Equal to, not equal to

**Bitwise AND/OR Operators:**

- `&`
  AND

- `^`
  Exclusive OR

- `|`
  Inclusive OR
Logical AND/OR Operators:

&&
   Logical AND
||
   Logical OR
? :
   If-else

Assignment Operator:

&&
   Logical AND
= *= /= %=
   Assignment
+= -= <<=
   Assignment
>>= &= ^= |=
   Assignment

You do not need the $(( )) syntax to enclose an arithmetic expression in these situations:

- In assignment to an integer variable. (See typeset.)
- As an argument to the following built-in shell commands:

<table>
<thead>
<tr>
<th>break</th>
<th>exit</th>
<th>return</th>
<th>continue</th>
<th>let</th>
<th>shift</th>
</tr>
</thead>
</table>

- When used as arguments in the test built-in shell command numeric comparisons (-eq, -ge, -gt, -le, -lt, and -ne). See test.

Command substitution

In command substitution, sh uses the expansion of the standard output of one command in the command line for a second command. There are two syntaxes.

The first syntax (called backquoting) surrounds a command with grave accents ``, as in:

```
ls `cat list`
```

To process this command line, sh first runs the cat command and collects its standard output. The shell then breaks this output into arguments and puts the result into the command line of the ls command. The previous command therefore lists the attributes of all files, the names of which are contained in the file list.

This syntax is easy to type, but is not useful if you want to put one command substitution inside another (nesting command substitutions). A more useful syntax is:

```
$(command)
```

as in:

```
ed $(grep -f -l function $(find . -name '*.c'))
```

This command uses find to search the current directory and its subdirectories to find all files, the names of which end in .c. It then uses grep -f to search each such file for those that contain the string function. Finally, it calls ed to edit each such file.

There is a historical inconsistency in the backquoting syntax. A backslash (\) within a backquoted command is interpreted differently depending on its context. Backslashes are interpreted literally unless
they precede a dollar sign ($), grave accent (´), or another backslash (\). In these cases, the leading backslash becomes an escape character to force the literal interpretation of the $, ´, or \. Consequently, the command:

```
echo '\$x'
```

issued at system level produces the output:

```
\$x
```

whereas the same command nested in a backquoted syntax:

```
echo `echo '\$x'`
```

produces the output:

```
$x
```

We recommend the $(command) syntax for command substitutions.

sh performs command substitutions as if a new copy of the shell is invoked to run the command. This affects the behavior of $- (standing for the list of options passed to the shell). If a command substitution contains $-, the expansion of $- does not include the -i option, since the command is being run by a non-interactive shell.

### File descriptors and redirection

The shell sometimes refers to files using file descriptors. A file descriptor is a number in the range 0 to 9. It can have any number of digits. For example, the file descriptors 001 and 01 are identical to file descriptor 1. Various operations (for example, exec) can associate a file descriptor with a particular file.

Some file descriptors are set up at the time the shell starts up. These are the standard input/output streams:

- Standard input (file descriptor 0)
- Standard output (file descriptor 1)
- Standard error (file descriptor 2)

Commands running under the shell can use these descriptors and streams too. When a command runs under the shell, the streams are normally associated with your terminal. However, you can redirect these file descriptors to associate them with other files (so that I/O on the stream takes place on the associated file instead of your terminal). In fact, the shell lets you redirect the I/O streams associated with file descriptors 0 through 9, using the following command-line constructs.

**number<file**

Uses file for input on the file descriptor, the number of which is number. If you omit number, as in <file, the default is 0; this redirects the standard input.

**number>file**

Uses file for output on the file descriptor, the number of which is number. If you omit number, as in >file, the default is 1; this redirects the standard output. The shell creates the file if it does not already exist. The redirection fails if the file already exists and noclobber is set (see set).

**number> | file**

Is similar to number>file but if file already exists, the output written to the file overwrites its current contents.

**number< >file**

Uses file for input and output with the file descriptor, the number of which is number. This is most useful when the file is another terminal or modem line. If you omit number, as in < >file, the default
number is zero; this redirects the standard input. Output written to the file overwrites the current contents of the file (if any). The shell creates the file if it does not already exist.

**number>>name**

Is similar to **number > file**, except that output is appended to the current contents of the file (if any).

**number<<[-]name**

Lets you specify input to a command from your terminal (or from the body of a shell script). This notation is known as a here-document. The shell reads from the standard input and feeds that as input to file descriptor number until it finds a line that exactly matches the given name. If you omit number, the default is the standard input. For example, to process the command:

```
cat <<abc >out
```

the shell reads input from the terminal until you enter a line that consists of the word abc. This input is passed as the standard input to the cat command, which then copies the text to the file out.

If in "name" any part (empty or not) is quoted or any character escaped, does not perform substitutions on the input; otherwise, it performs variable and command substitutions, respecting the usual quoting and escape conventions. If you put - before name, sh deletes all leading tabs in the here-document.

**number1<&number2**

Makes the input file descriptor number1 a duplicate of file descriptor number2. If you omit number1, the default is the standard input (file descriptor 0). For example, <&4 makes the standard input a duplicate of file descriptor 4. In this case, entering input on 4 has the same effect as entering input on standard input.

**number1>&number2**

Makes the output file descriptor number1 a duplicate of file descriptor number2. If you omit number1, the default is the standard output (file descriptor 1). For example, >&2 makes the standard output a duplicate of file descriptor 2 (the standard error). In this case, writing output on standard output has the same effect as writing output on standard error.

**number<&-**

Closes input descriptor number. If you omit number, it closes the standard input.

**number>&-**

Closes output descriptor number. If you omit number, it closes the standard output.

Normally, redirection applies only to the command where the redirection construct appears; however, see exec.

The order of redirection specifications is significant, since an earlier redirection can affect a later one. However, these specifications can be freely intermixed with other command arguments. Since the shell takes care of the redirection, the redirection constructs are not passed to the command itself.

**Note:** The shell performs the implicit redirections needed for pipelines before performing any explicit redirections.

**File name generation**

The characters * ? [ are called glob characters, or wildcard characters. If an unquoted argument contains one or more glob characters, the shell processes the argument for file name generation. The glob characters are part of glob patterns, which represent file and directory names. These patterns are similar to regular expressions, but differ in syntax, since they are intended to match file names and words (not arbitrary strings). The special constructions that may appear in glob patterns are:

? Matches exactly one character of a file name, except for the separator character / and a . at the beginning of a file name. ? only matches an actual file name character and does not match
nonexistent characters at the end of the file name. ? is analogous to the metacharacter . in regular expressions.

* Matches zero or more characters in a file name, subject to the same restrictions as ?. * is analogous to the regular expression .*.

[chars]
Defines a class of characters; the glob pattern matches any single character in the class. A class can contain a range of characters by writing the first character in the range, a dash -, and the last character. For example, [A-Za-z], in the POSIX locale, stands for all the uppercase and lowercase letters. If you want a literal - character (or other glob character) in the class, use the backslash to escape the character, causing it to lose its special meaning within the pattern expression. If the first character inside the brackets is an exclamation mark (!), the pattern matches any single character that is not in the class.

Some sample patterns are:

[!a-f]*.c
Matches all .c files beginning with something other than the letters from a through f.

/???/??.
Matches all files that are under the root directory in a directory with a three-letter name, and that have a basename containing one character followed by a . followed by another single character.

*/.[chyl]
Matches all .c, .h, .y, and .l files in a subdirectory of the working directory.

~mks/*.ksh
Matches all shell scripts in the home directory of user mks (see “Directory substitution” on page 598 for the use of ~).

If no files match the pattern, sh leaves the argument untouched. If the set option -f or “-o noglob” is in effect, the shell does not perform file name generation.

Double-byte characters in a file name may cause problems. For instance, if you use a double-byte character in which one of the bytes is a . (dot) or / (slash), the file system treats this as part of the path name.

Variables

The shell maintains variables and can expand them where they are used in command lines; see “Parameter substitution” on page 598 for details.

A variable name must begin with an uppercase or lowercase letter or an underscore (_). Subsequent characters in the name, if any, can be uppercase or lowercase letters, underscores, or digits 0 through 9. You can assign a value to a variable with:

```
variable=value
```

For integer variables (see “Options” on page 762 for details), the value may be specified as an arithmetic expression. For the syntax of an arithmetic expression, see “Arithmetic substitution” on page 600.

You can implicitly declare a variable as an array by using a subscript expression when assigning a value, as in:

```
variable[arithmetic expression]=value
```

You can use a subscripted array variable anywhere that the shell allows an ordinary variable. For the syntax of an arithmetic expression, see “Arithmetic substitution” on page 600. Also see `typeset`, `export`, and `readonly` for details about the attributes of shell variables, and how shell variables can be exported to child processes.
For a list of variables that the shell either sets or understands, see Shell variables.

Shell variables

You cannot use double-byte characters for a shell variable name, but you can use them for shell variable values. Double-byte characters in file names and path names are treated as single-byte characters.

Shell variables that are exported are called environment variables and are made available in the environment of all commands that are run from the shell. Table 29 on page 606 contains a list of built-in shell variables and also includes frequently-used environment variables. For more information about environment variables that are used by the C-RTL, see z/OS XL C/C++ Programming Guide. A list of other environment variables can be found in z/OS UNIX System Services User's Guide. Table 29 on page 606 lists frequently-used shell variables and their purposes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>(Underscore) For every command that is run as a child of the shell, sh sets this variable to the full path name of the executable file and passes this value through the environment to that child process. When processing the MAILPATH variable, this variable holds the value of the corresponding mail file.</td>
</tr>
<tr>
<td>~</td>
<td>(Tilde) expands to value of the HOME directory.</td>
</tr>
<tr>
<td>_UNIX03</td>
<td>When _UNIX03 is set to YES, the utilities that have implemented support for the UNIX03 specification will conform to the UNIX03 specification. This variable is only needed when the syntax or behavior of UNIX03 conflicts with the existing implementation. The value YES must be specified in uppercase.</td>
</tr>
<tr>
<td>CPATH</td>
<td>Contains a list of directories for the cd command to search. Directory names are separated with colons. CPATH works like the PATH variable.</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>Used by several commands to define the width of the terminal output device.</td>
</tr>
<tr>
<td>EDITOR</td>
<td>Enables the corresponding editing mode (see set and shedit) when using vi, emacs, or gmacs.</td>
</tr>
<tr>
<td>ENV</td>
<td>Contains the path name of a setup script that contains commands and aliases. When you invoke sh as a login shell, the ENV script is run after the login profiles (/etc/profile, $HOME/.profile), before the shell accepts commands. For other sh invocations, the ENV script is run before the shell accepts commands. It is typically used to define shell options, functions and aliases. sh performs parameter substitution on this value and uses the results as the name of a setup script. This script is run in the current shell environment. The ENV variable is usually set in your .profile.</td>
</tr>
<tr>
<td>ERRNO</td>
<td>Contains the system error number of the most recently failed system call. The shell sets this variable only for errors that occur in the current environment. Assigning a value of 0 to this variable clears it.</td>
</tr>
<tr>
<td>FCEDIT</td>
<td>Contains the name of the default editor for the fc command. If this variable is not set, the default is the ed command.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FPATH</td>
<td>Contains a list of directories that the system searches to find executable functions. Directories in this list are separated with colons. sh searches each directory in the order specified in the list until it finds a matching function. If you want the shell to search the working directory, put a dot (.) or a null string in the list of directories (for example, to tell the shell to search the working directory first, start the list with a colon or semicolon).</td>
</tr>
<tr>
<td>HISTFILE</td>
<td>Contains the path name of a file to be used as the history file. When the shell starts, the value of this variable overrides the default history file.</td>
</tr>
<tr>
<td>HISTSIZE</td>
<td>Contains the maximum number of commands that the shell keeps in the history file. If this variable contains a valid number when the shell starts, it overrides the default of 127.</td>
</tr>
<tr>
<td>HOME</td>
<td>Contains your home directory. This is also the default directory for the cd command. The HOME variable is set automatically from the RACF user profile when the user logs in.</td>
</tr>
<tr>
<td>IFS</td>
<td>Contains a series of characters to be used as internal field separator characters. Any of these characters can separate arguments in unquoted command substitutions such as <code>command</code> or $(command), or in parameter substitutions. In addition, the shell uses these characters to separate values put into variables with the read command. Finally, the first character in the value of IFS separates the positional parameters in $* expansion. By default, IFS contains space, tab, and newline.</td>
</tr>
<tr>
<td>LANG</td>
<td>Contains the default locale value.</td>
</tr>
<tr>
<td>LIBPATH</td>
<td>Used to specify the directory to search for a DLL (Dynamic Link Library) file name. If it is not set, the working directory is searched. For more information, see d11oad in z/OS XL C/C++ Runtime Library Reference. LIBPATH can be updated by the _CEE_ENVFILE or _CEE_ENVFILE_S environment variables. For more information about _CEE_ENVFILE, see z/OS XL C/C++ Programming Guide.</td>
</tr>
<tr>
<td>LINENO</td>
<td>Contains the number of the line currently being run by a shell script or within a function.</td>
</tr>
<tr>
<td>LINES</td>
<td>Used by several commands to define the number of lines on the terminal output device.</td>
</tr>
<tr>
<td>LOCPATH</td>
<td>Tells the setlocale() function the name of the directory in the z/OS UNIX file system from which to load locale object files. (localedef produces locale object files by processing locale source files.)</td>
</tr>
<tr>
<td>LOGNAME</td>
<td>Contains the user login name. This is set automatically from the RACF user profile when the user logs in.</td>
</tr>
<tr>
<td>MAILCHECK</td>
<td>Contains the number of seconds of elapsed time that must pass before the system checks for mail; the default value is 600 seconds. When using the MAIL or MAILPATH variables, the shell checks for mail before issuing a prompt.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>MAILPATH</td>
<td>Contains a list of mailbox files. MAILPATH overrides the MAIL variable. The mailbox list is separated by colons. If any name is followed by <code>?message</code> or <code>%message</code>, <code>sh</code> displays the message if the corresponding file has changed. <code>sh</code> performs parameter and command substitution on message, and the variable <code>_</code> (temporarily) expands to the name of the mailbox file. If no <code>?message</code> or <code>%message</code> is present, the default message is <code>you have mail in $_</code>.</td>
</tr>
<tr>
<td>MANPATH</td>
<td>Contains a list of paths to search for man pages.</td>
</tr>
<tr>
<td>MBOX</td>
<td>Contains the path name of your personal mailbox, usually $HOME/mbox, used to store messages that have been read from your system mailbox. This variable is usually set in your <code>.profile</code>.</td>
</tr>
<tr>
<td>NLSPATH</td>
<td>Specifies where the message catalogs are to be found.</td>
</tr>
<tr>
<td>OLDPWD</td>
<td>Contains the name of the directory you were previously working in. The cd command sets this variable.</td>
</tr>
<tr>
<td>PATH</td>
<td>Contains a list of directories that the system searches to find executable commands. Directories in this list are separated with colons. <code>sh</code> searches each directory in the order specified in the list until it finds a matching executable. If you want the shell to search the working directory, put a dot (.) or a null string in the list of directories (for example, to tell the shell to search the working directory first, start the list with a colon or semicolon). The shell commands directory <code>/bin</code> must always be in the list of directories. For more information about specifying the PATH variable, see the topic in z/OS UNIX System Services User’s Guide on using the PATH variable when customizing the search path for commands.</td>
</tr>
<tr>
<td>PPID</td>
<td>Contains the decimal value of the process ID of the parent of the shell. If running in a child shell environment (see “Shell execution environments” on page 611), the PPID value is the same as the PPID value of the current shell.</td>
</tr>
<tr>
<td>PS1</td>
<td>Contains the primary prompt string used when the shell is interactive. The default value is a dollar sign followed by a space (<code>$</code>). The shell expands parameters before the prompt is printed. A single exclamation mark (!) in the prompt string is replaced by the command number from the history list; see the <code>fc</code> command. For a real exclamation mark in the prompt, use <code>!!</code>. This variable is usually set in your <code>.profile</code>.</td>
</tr>
<tr>
<td>PS2</td>
<td>Contains the secondary prompt, or continuation prompt, used when completing the input of such things as reserved-word commands, quoted strings, and here-documents. The default value of this variable is a greater than sign followed by a space (<code>&gt;</code>).</td>
</tr>
<tr>
<td>PS3</td>
<td>Contains the prompt string used with the <code>select</code> reserved word. The default value is a number sign followed by a question mark and a space (<code>#?</code>).</td>
</tr>
<tr>
<td>PS4</td>
<td>Contains the prefix for traced commands with <code>set -x</code>. The default value is a plus sign followed by a space (<code>+</code>).</td>
</tr>
<tr>
<td>PWD</td>
<td>Contains the name of the working directory. When the shell starts, the working directory name is assigned to PWD unless the variable already has a value.</td>
</tr>
<tr>
<td>RANDOM</td>
<td>Returns a random integer. Setting this variable sets a new seed for the random number generator.</td>
</tr>
</tbody>
</table>
### Table 29: Built-in shell variables (sh command) (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SECONDS</strong></td>
<td>Contains elapsed time. The value of this variable grows by 1 for each elapsed second of real time. Any value assigned to this variable sets the SECONDS counter to that value; initially the shell sets the value to 0.</td>
</tr>
<tr>
<td><strong>SHELL</strong></td>
<td>Contains the full path name of the current shell. It is not set by the shell, but is used by various other commands to invoke the shell. This is set automatically from the RACF user profile when the user logs in.</td>
</tr>
<tr>
<td><strong>STEPLIB</strong></td>
<td>Identifies a STEPLIB variable to be used in building a process image for running an executable file. A STEPLIB is a set of private libraries used to store a new or test version of an application program, such as a new version of a runtime library. STEPLIB can be set to the values CURRENT or NONE or to a list of MVS data set names.</td>
</tr>
<tr>
<td></td>
<td>If STEPLIB is not set, it defaults to CURRENT, which passes on the TASKLIB, STEPLIB, or JOBLIB allocations that are part of the invoker’s MVS program search order environment to the process image created for an executable file.</td>
</tr>
<tr>
<td></td>
<td>IBM recommends that STEPLIB be set to NONE, which indicates you do not want a STEPLIB environment for executable files. You can specify up to 255 MVS data set names, separated by colons, as a list of data sets used to build a STEPLIB variable. Refer to z/OS UNIX System Services Planning for more information about building a STEPLIB environment.</td>
</tr>
<tr>
<td><strong>TMOUT</strong></td>
<td>Contains the number of seconds before user input times out. If user input has not been received within this length of time, the shell ends.</td>
</tr>
<tr>
<td></td>
<td>The _BPXK_TIMEOUT environment variable allows the timeout value of the job to be overridden on an individual process basis.</td>
</tr>
<tr>
<td></td>
<td>The system administrator can specify the BPXPRMxx PWT option to honor the SMF job wait time values and to allow specification of the _BPXK_TIMEOUT variable. These settings can be used instead of TMOUT to control when the shell times out. The PWT and _BPXK_TIMEOUT options will also time out commands running in the shell, such as vi or oedit, providing more control over timing out processes. If the TMOUT variable is set in combination with the BPXPRMxx PWT option or the _BPXK_TIMEOUT variable (or both), then the TMOUT setting is honored for timing out the shell if the TMOUT time value is less than the SMF job wait times.</td>
</tr>
<tr>
<td><strong>TMPDIR</strong></td>
<td>Is the path name of the directory being used for temporary files. If it is not set, the z/OS shell uses /tmp.</td>
</tr>
<tr>
<td><strong>TZ</strong></td>
<td>Contains the system time zone value used for displaying date and time. You can set the TZ variable in your $HOME/.profile file used during shell startup.</td>
</tr>
<tr>
<td></td>
<td>The system administrator can also define a TZ default for all shell users in the /etc/profile file. If you are not in the same time zone, you can set TZ yourself.</td>
</tr>
<tr>
<td></td>
<td>The system administrator can also define TZ for the /etc/init process in the /etc/init.options file.</td>
</tr>
<tr>
<td><strong>VISUAL</strong></td>
<td>Overrides the EDITOR environment variable in setting vi, emacs, or gmacs editing modes (see shedit).</td>
</tr>
</tbody>
</table>
Shell variables for automatic conversion

When the shell is redirecting standard input, standard output, or standard error, it will default to no automatic conversion of tagged files, and no tagging of files created by the redirection. The following shell variables will override this behavior:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>_TAG_REDIR_IN=TXT</td>
<td>Redirected standard input will override the file's text flag (TXTFLAG), treating it as if it were tagged as: TXTFLAG = ON, CCSID = existing file tag CCSID. This has no effect if CCSID = 0.</td>
</tr>
<tr>
<td>_TAG_REDIR_IN=BIN</td>
<td>Redirected standard input will override the file's TXTFLAG, treating it as if it were tagged as: TXTFLAG = OFF, CCSID = existing file tag CCSID. This effectively disables automatic conversion.</td>
</tr>
<tr>
<td>_TAG_REDIR_OUT=TXT</td>
<td>Redirected standard output will be tagged as: TXTFLAG = ON, CCSID = program CCSID at the time of the first write (if not already tagged)</td>
</tr>
<tr>
<td>_TAG_REDIR_OUT=BIN</td>
<td>Redirected standard output will be tagged as: TXTFLAG = OFF, CCSID = program CCSID at the time of the first write (if not already tagged)</td>
</tr>
<tr>
<td>_TAG_REDIR_ERR=TXT</td>
<td>Redirected standard error will be tagged as: TXTFLAG = ON, CCSID = program CCSID at the time of the first write (if not already tagged)</td>
</tr>
<tr>
<td>_TAG_REDIR_ERR=BIN</td>
<td>Redirected standard error will be tagged as: TXTFLAG = OFF, CCSID = program CCSID at the time of the first write (if not already tagged)</td>
</tr>
</tbody>
</table>

The automatic conversion shell variable can be specified for one command, or for multiple commands within a shell session or shell script. If the variable is exported, it will affect child shells, that is, nested shell scripts.

Note: Because the standard shell execution performs redirection before variable assignment, the syntax for specifying the shell variable for one command is:

```bash
(_TAG_REDIR_OUT=TXT; command >file)
```

You can also use these shell variables for commands in a pipeline. For example, they can be used to tag the standard output of each command that is writing to a pipeline or to tag the standard input of each command that is reading from a pipeline.

The _TAG_REDIR_IN shell variable can also be used with here-documents to tag the input that is fed into the associated command.
Shell execution environments

A shell execution environment is the set of conditions affecting most commands run within the shell. It consists of:

- Open files
- The working directory (see cd)
- The file creation mask (see umask)
- The traps currently set (see trap)
- The shell parameters (see set and export)
- The shell functions currently defined (see Command execution)
- Options (see set)

A child shell environment starts as a duplicate of the shell environment, except that traps caught by the shell are set to default values in the child shell. Since the child shell environment starts as a duplicate, the value of the ENV environment variable is not run. Changes made to a child shell environment do not affect the shell environment.

Command substitutions (such as $command), commands within parentheses (such as (command)), and commands to be run asynchronously (such as command&), all run in child shell environments. Each command in a pipeline (such as “command | command”) runs in a child shell environment, unless the pipecurrent shell option is in effect. If pipecurrent is set on (with set -o pipecurrent or set -P), then the last command of the pipeline is executed in the current shell environment.

Shell commands also run in a separate environment that does not affect the shell environment, except for certain built-in commands (for example, cd and umask) that explicitly alter the shell environment. The environment of a shell command is set up by the shell to include the following:

- Open files, subject to redirection.
- Working directory (see cd).
- File creation mask (see umask).
- Traps; traps caught by the shell are set to default values and traps ignored by the shell are ignored by the command.
- Variables defined inside the shell and having the export attribute.

Built-in commands

Some commands are built into the shell to increase performance of shell scripts or to access the shell's internal data structures and variables. These internal commands are designed to have semantics indistinguishable from external commands.

```
:       chgrp       exit       let       r       times       whence
.      chmod       export      link      read      trap      writedown
[ [    chown       false      ln      readonly       true
alias    comm       fc       login      return      type
autoload    command      fg       ls       rm       typeset
basename    cp       getopt      mv       shift      unalias
bg      du       history      newgrp      stop       unlink
break     echo       integer      print      suspend      unset
cat      eval       jobs       printf      test       unset
cd      exec       kill       pwd       time       wait
```
POSIX.2 recognizes a subset of these commands as special built-ins. Syntax errors in special built-in commands may cause a shell executing that command to terminate, while syntax errors in regular built-in commands will not cause the shell executing that command to terminate. If a special built-in command encountering a syntax error does not terminate the shell, its exit value is nonzero.

Also, shell variable assignments included on shell command lines that invoke special built-in commands remain in effect after the built-in command completes; this is not the case with regular built-in commands or other utilities. These special built-in commands are as follows:

\begin{verbatim}
  :  
  break
  continue
  eval
  exec
  exit
  export
  set
  readonly
  return
  typeset
  shift
  trap
  unset
\end{verbatim}

As well as built-in commands, the shell has a set of predefined aliases as follows:

\begin{verbatim}
  autoload
  hash
  functions
  integer
  history
  r
  nohup
  suspend
  stop
\end{verbatim}

See alias for details.

**Shell archives**

Software distributed over computer networks such as Usenet is often distributed in a form known as a shell archive. In essence, a shell archive is a shell script containing the data of one or more files, plus commands to reconstruct the data files and check that the data was sent correctly. The following shows a sample shell archive:

```bash
# This is a shell archive.
# It contains the one file "frag.ksh"
# To extract contents, type
# sh file
#
if      [ -f frag.ksh ]
then    echo frag.ksh exists: will not overwrite
else    echo extracting frag.ksh
        sed 's/^X//' >frag.ksh <<_EOF_
        X# This is frag.ksh
        X# Not very interesting, really.
        Xecho frag.ksh here!
_EOF_
        if [ "`sum frag.ksh|awk '{print $1["'"]'" != 52575 ]
then    echo frag.ksh damaged in transit
        fi
```

612  z/OS: UNIX System Services Command Reference
The following is a simple script to produce as much of the Fibonacci sequence as can be calculated in integers:

```
# Print out Fibonacci sequence; start sequence
# with first two positional parameters:
# default 1 1
typeset -i x=${1:-1} y=${2:-1} z
while [ x -gt 0 ] # until overflow
do
  echo $x
  let z=y+x y=x x=z
done
```

The following implements the basename command as a shell function:

```
# basename command as shell function
function basename {
  case $# in
    1) ;;
    2) eval set \${1%$2} ;;
    *) echo Usage: $0 pathname '[suffix]' 
        return 1 ;;
  esac
  echo ${1##*/}
  return 0
}
```

### Files

- **sh_history**
  - The default history storage file.

- **.profile**
  - The user profile for login shell.

- **/etc/profile**
  - The system-wide profile for login shells.

- **/tmp/sh***
  - Temporary files for here-documents, command substitution, history re-execution, and so on. The default directory /tmp can be overridden by setting the shell variable TMPDIR to the name of some other directory.

- **/etc/suid_profile**
  - Used instead of the script specified by the ENV variable (and the $HOME/.profile for a login shell) under the privileged option or when the real and effective UIDs are different, or the real and effective GIDs are different.

### Localization

**sh** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX

Exit values

0
  Successful completion

1
  Failure due to any of the following:
  • The shell was invoked with an incorrect option.
  • The shell was invoked to run a shell script and the command.
  • A command syntax error.
  • A redirection error.
  • A variable expansion error.

Otherwise, the exit status of the shell defaults to the exit status of the last command run by the shell. This
default can be overridden by explicit use of the exit or return commands. The exit status of a pipeline is
the exit status of the last command in the pipeline.

Messages

Ambiguous redirection
  A redirection construct expanded to more than one path name.

Argument too long
  Any single argument to a command is limited in length (see "Limits" on page 615). Command and
  parameter substitution may exceed this limit.

Cannot restore privileged state
  This message occurs only when the implementation of POSIX does not support the saved IDs option
  (_POSIX_SAVED_IDS). The message is generated if you tried to use a saved ID feature to return to a
  privileged state.

File file already exists
  You are attempting to redirect output into an existing file, but you have turned on the noclobber option
  (see the set command). If you really want to redirect output into an existing file, use the
  construct > [filename], or turn off the option with:

  set +o noclobber

File descriptor number already redirected
  You attempted to redirect a file descriptor that was already being redirected in the same command.
  You can redirect a file descriptor only once.

Hangup
  The shell received a hangup signal. This signal typically arises when a communication line is
  disconnected, for example, when a phone connection is cut off.

In base#number: base must be in [2,36]
  In a number of the form base#number, the value of the base was larger than 36 or less than 2. The
  only valid range for bases is from 2 through 36.

Invalid subscript
  A shell array was indexed with a subscript that was outside the defined bounds.

Illegal instruction
  The shell received an illegal instruction signal. This signal typically occurs when a process tries to
  execute something that is not a valid machine instruction recognized by the hardware.

Misplaced subscript array name
  The subscript for an array was missing or incorrect.

name is not an identifier
  You attempted to use a non-alphanumeric name.

name: readonly variable
  The given name is a read-only variable, and cannot be removed or changed (see readonly).
**name: no expansion of unset variable**

The shell is operating with `set -u`, and you used an unset variable in a substitution. For more information, see the `set` command.

**No file descriptor available for redirection**

When a file descriptor is redirected, the old value is remembered by the shell by a duplication to yet another file descriptor. The total number of file descriptors is limited by the system; hence, the shell may run out, even though your command appears to be using far fewer than the maximum number of descriptors.

**Nested aliases**

You have more than nine levels of aliases. For example:

```
alias a1=a2 a2=a3 a3=a4 ... a19=command
```

causes this error.

**Pipe for coprocess**

The shell cannot create a pipe for a coprocess. This might mean that your session or the system as a whole has already set up its maximum number of pipes.

**...: restricted**

If the shell has been invoked as a restricted shell, certain things are disallowed; for example, the `cd` command, setting `PATH`, and output redirection.

**Temporary file error using here-document**

`sh` tried to create a temporary file holding the contents of a `<<word` here-document. However, the temporary file could not be created. This may indicate a lack of space on the disk where temporary files are created.

**Word after ... expanded to more than one argument**

In a context where only one argument was expected, a construct expanded to more than one argument.

**Limits**

The maximum length of an executable file name, including subdirectories and extensions, is 1023 bytes.

**Portability**

POSIX.2, X/Open Portability Guide.

The construct `$[arithmetic expression]` is an extension of the POSIX standard.

**Related information**

`alias`, `break`, `cd`, `continue`, `dot`, `echo`, `eval`, `exec`, `exit`, `export`, `fc`, `getopts`, `let`, `print`, `ps`, `pwd`, `read`, `readonly`, `return`, `set`, `shift`, `test`, `time`, `trap`, `true`, `typeset`, `ulimit`, `unalias`, `unset`, `whence`, `shedit`

Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.

**shedit - Interactive command and history editing in the shell**

**Format**

```
set -o editmode
EDITOR=editprog
VISUAL=editprog
```
Usage notes

POSIX uses a number of keys for such things as erase and kill processing. By default, the shell leaves command-line editing to POSIX, using these familiar editing keys. However, these functions are not particularly powerful or friendly. As an alternative, the shell has built-in facilities for interactive command editing and file name generation that not only aid in composing new commands but also make it easy for you to modify and re-execute previous commands. This capability is distinct from that provided by the `fc` command, which passes previous command lines to a separate program for editing. The built-in facilities mimic the emacs, gmacs, or vi screen editors, and enable the following commands (see `set` and `vi` for details).

```
set -o emacs
set -o gmacs
set -o vi
```

These facilities are also enabled (with the corresponding option set) by assigning a value ending in vi to the environment variables EDITOR or VISUAL. (See `sh`.)

Unlike full-screen editors, shell editing uses a one-line window, extending from the end of the prompt to the next-to-last column. Multiline history entries are displayed with newlines represented as ^J.

The number of columns on the output device is obtained from the COLUMNS environment variable if defined; otherwise, it is assumed to be 80.

A command line that extends into the rightmost column can be scrolled horizontally. If you try to move the cursor beyond the edge of the window, the line is scrolled to approximately center the cursor in the window. The second last column displays a character marking a scrollable line: < indicates extra data off the left, > indicates extra data off the right, and * indicates extra data off both sides.

emacs/gmacs editing mode

When the emacs/gmacs editing mode has been enabled, ordinary printable characters from the keyboard are entered in the command line and echoed. Various control characters introduce command sequences for such things as moving the cursor, scrolling through the command history, and modifying the current command. The only difference between emacs and gmacs is in the handling of Ctrl-T.

The command sequences recognized are listed in functional groups. The notation Meta– represents EscK, followed by the letter. The terminology is historical. Many commands accept an optional preceding count which is entered in decimal as Meta–digits, or as Ctrl–, which multiplies the current count (initially 1) by 4.

List of cursor movement combinations:

- `nCtrl-B`
  Moves the cursor back $n$ characters.
- `nCtrl-F`
  Moves the cursor forward $n$ characters.
- `Ctrl-A`
  Moves the cursor to beginning of line.
- `Ctrl-E`
  Moves the cursor to end of line.
- `nMeta-b`
  Moves the cursor back to the $n$th previous beginning of word (string of alphanumerics).
- `nMeta-f`
  Moves the cursor forward to $n$th beginning of word.
- `Ctrl-Jc`
  Moves the cursor forward to next character $c$ on current line.
- `Meta-space`
  Sets mark at cursor position.
\texttt{Ctrl-@}

Sets mark at cursor position.

\texttt{Ctrl-x Ctrl-X}

Exchanges cursor position and mark.

List of \texttt{line search} combinations; these numbers display a different history line.

\texttt{nCtrl-p}

Selects the \textit{n}th previous command line from history.

\texttt{nCtrl-n}

Selects the \textit{n}th next command line from history.

\texttt{Meta-<}

Selects the earliest command line from history.

\texttt{Meta->}

Selects the latest command line from history.

\texttt{nCtrl-r \texttt{stringEnter}}

Selects the \textit{n}th previous command line matching \textit{string}. If \textit{n} is zero, then select the next matching command after the current line.

List of \texttt{text change} combinations:

\texttt{n erase}

Deletes \textit{n} characters to the left of the cursor. This is the erase character.

\texttt{nBackspace}

Deletes \textit{n} characters to the left of the cursor.

\texttt{nCtrl-h}

Deletes \textit{n} characters to the left of the cursor.

\texttt{nCtrl-d}

Deletes \textit{n} characters to the right of the cursor. If the current line is empty, the shell is ended.

\texttt{nMeta-Ctrl-h}

Deletes to the \textit{n}th beginning of word before the cursor.

\texttt{nMeta-h}

Deletes to the \textit{n}th beginning of word before the cursor.

\texttt{nMeta-d}

Deletes to the \textit{n}th beginning of word after the cursor.

\texttt{nCtrl-k}

Deletes from the cursor to the end of line. If \textit{n} is zero, then deletes from the beginning of line to the cursor.

\texttt{kill}

Deletes the entire current line. This is the line kill character.

\texttt{Ctrl-G}

Deletes the entire current line.

\texttt{Ctrl-W}

Deletes from cursor position to the mark (set with Meta-space or \texttt{Ctrl-@}).

\texttt{Ctrl-T}

In emacs mode, transposes the current character with the previous character and moves the cursor forward. If the cursor is at the end of the line, or in gmacs mode, transposes the previous two characters.

\texttt{Ctrl-Y}

Restores the last text deleted in emacs mode.

\texttt{Ctrl-C}

Capitalizes character under cursor.
**Ctrl-^**
Capitalizes character under cursor.

**Meta-c**
Capitalizes word to right of cursor.

**Meta-l**
Lowercases word to right of cursor.

**Meta-u**
Uppercases word to right of cursor.

**nMeta-.**
Inserts the nth word of the previous command. If n is not given or it is zero, inserts the last word of the previous command.

**nMeta-_**
Inserts the nth word of the previous command. If n is not given or it is zero, inserts the last word of the previous command.

**Meta-**
Replaces the current word with the list of files which would match that word with an * appended.

**Meta-Esc**
Used to complete a path name. If there is only one existing path name that matches as much as you've typed, the path name is completed and a space is added after the complete path name. If there are several matching path names, the shell expands what you've typed by adding all the characters that are common to all matching path names.

**Meta-=**
Lists all path names matching the current word.

List of **miscellaneous** combinations:

**Ctrl-J**
Executes the current command line.

**Ctrl-M**
Executes the current command line.

**Ctrl-L**
Redisplays the current command line.

**Ctrl-O**
Remembers the next command line, executes the current command line, then selects the remembered line.

**Ctrl-U**
Multiplies the count on the following command by 4 (for each Ctrl-U).

**Ctrl-V**
Displays the version of the shell.

**-**
Takes the next character literally. Thus, you can enter command and control characters in a command line or search string.

**eof**
Terminates the shell. This is the end-of-file character.

**Ctrl-D**
Terminates the shell.

**Meta-n**
Enteres a count for the following command.
**vi editing mode**

When the vi editing facilities have been enabled, the shell is initially in input mode after each new prompt. Keyboard input is normally inserted at the current position in the current command line; the exceptions are the following action keys.

**erase**
- Deletes the character to the left of the cursor. This is the erase character.

**Backspace**
- Deletes the character to the left of the cursor.

**eof**
- Terminates the shell. This is the end-of-file character.

**Ctrl-D**
- Terminates the shell.

**Ctrl-W**
- Deletes the word (white-space delimited string) to the left of the cursor.

**kill**
- Deletes the current line. This is the line kill character.

**Ctrl-X**
- Deletes the current line.

**Ctrl-J**
- Deletes the current line.

**Enter**
- Executes the current line.

**Esc**
- Switches from input mode to command mode.

If you press the Esc key, the shell enters command mode and keyboard input is interpreted as commands to reposition the cursor, scroll through the command history, delete or change text, or reenter input mode. In command mode, input is not echoed; it is acted upon. Many commands take an optional count, \( n \), which is entered as a preceding decimal number (not echoed); the command is executed that number of times. Except where otherwise noted, \( n \) defaults to 1.

**Ctrl-V**
- Takes the next character literally; useful for entering any of these action keys as text.

\`
\`
- Escapes the following action key. If the next character is any action key except Ctrl-J, Ctrl-M, or Enter, the - is erased and the escaped character is entered literally. Otherwise, the – is entered and the next character is treated normally.

**Cursor movement**

These commands reposition the cursor in the command line.

**nh**
- Moves back \( n \) characters.

**n1**
- Moves forward \( n \) characters.

**0**
- Moves to the first character on the line.

**^**
- Moves to the first nonblank character on the line.
$  
Moves to the last character on the line.

nw  
Moves to the beginning of the $n$th next word (string of alphanumerics, or of nonblank nonalphanumerics).

nW  
Moves to the beginning of the $n$th next fullword (string of nonblanks).

nb  
Moves to the $n$th previous beginning of word.

nB  
Moves to the $n$th previous beginning of fullword.

ne  
Moves to the $n$th next end of word.

nE  
Moves to the $n$th next end of fullword.

nfc  
Moves to the $n$th next character $c$.

nFc  
Moves to the $n$th previous character $c$.

ntc  
Moves to the character before the $n$th next character $c$.

nTc  
Moves to the character after the $n$th previous character $c$.

;  
Repeats the previous f, F, t, or T command.

,  
Repeats the previous f, F, t, or T command, but in the opposite direction.

Line search
These commands change the current displayed command line.

nj  
Selects the $n$th next command line from history.

n+  
Selects the $n$th next command line from history.

nk  
Selects the $n$th previous command line from history.

n–  
Selects the $n$th previous command line from history.

nG  
Selects the command with history number $n$, or the latest command if $n$ is omitted.

/stringEnter  
Selects the first command line, searching backwards, that matches string. If string is omitted, the previous search string is used.

?stringEnter  
Selects the first command line, searching forwards, that matches string. If string is omitted, the previous search string is used.

n  
Repeats the last string search (’/’ or ‘?’) command.

N  
Repeats the last string search, but in the opposite direction.
Text change
The following commands alter the text in the current command line. Some of these commands operate on
a text block, defined by an immediately following cursor movement command. This is designated by \texttt{m}
(for movement) in the text change command. The text block extends from the current cursor position to
the new position determined by the movement command.

\texttt{i}
Enters input mode, inserting text before the character under the cursor.

\texttt{I}
Inserts before the first nonblank on line (\texttt{^i}).

\texttt{a}
Moves the cursor forward one character and enter input mode, appending text after the character
originally under the cursor.

\texttt{A}
Appends to end of line (\texttt{$a$}).

\texttt{ndm}
Deletes text block. If \texttt{n} is given, it is applied to the \textit{movement}.

\texttt{dd}
Deletes entire command line.

\texttt{D}
Deletes from cursor to end of line (\texttt{d$}).

\texttt{nx}
Deletes \texttt{n} characters to right of cursor (\texttt{ndl}).

\texttt{nX}
Deletes \texttt{n} characters to left of cursor (\texttt{ndh}).

\texttt{ncm}
Change text block; deletes block of text and enters input mode. If \texttt{n} is given, it is applied to the
\textit{movement}.

\texttt{cc}
Change entire command line.

\texttt{s}
Change entire command line.

\texttt{ns}
Change next \texttt{n} characters from cursor.

\texttt{np}
Puts back, after the character under the cursor, \texttt{n} copies of the last block deleted by a text change
command.

\texttt{nP}
Puts back, before the character under the cursor, \texttt{n} copies of the last block deleted by a text change
command.

\texttt{rc}
Replaces the single character under the cursor with the character \texttt{c}, and advances the cursor one
position.

\texttt{R}
Enteres \textit{replace mode}: a special case of input mode in which each character entered overwrites that
under the cursor, and advances the cursor one position.

\texttt{u}
Undoes the last text change to the current line. This is itself a text change command, and so acts as a
toggle for the last change.

\texttt{U}
Undoes all changes to the current line.
n
Inverts the case of the next n characters, advancing the cursor over them.

n.
Repeats the last text change command. If n is given, it overrides the count originally given with the repeated command.

n_
Appends after the character under the cursor, the nth argument from the previous command line (default last), and enter input mode.

*
Replaces the current word with the list of file names that matches the word with an * appended. If there is no match, an audible alarm sounds and the word is not changed. Otherwise, the cursor is positioned at the end of the list and input mode is entered.
\
Used to complete a path name. If there is only one existing path name that matches as much as you've typed, the path name is completed and a space is added after the complete path name. If there are several matching path names, the shell expands what you've typed by adding all the characters that are common to all matching path names.
=
Lists all path names matching the current word.

Miscellaneous
nym
Yanks text block into the delete buffer. Does not alter the command line or cursor position, but makes the text block available to subsequent put or p commands. If n is given, it is applied to the movement.

yy
Yanks the entire command line.

Y
Equivalent to y$. Yanks the rest of the line.

#
Equivalent to I#Enter.

nv
Executes fc -e ${VISUAL:--${EDITOR:--vi}} n. If n is omitted, the history number of the current line is used.

Ctrl-L
Redisplays the current line.

Ctrl-J
Executes the current line.

cm
Executes the current line.

Enter
Executes the current line.

@letter
Inserts the value of the alias named _letter. The symbol letter represents a single alphabetic character from the portable character set; implementations may support additional characters as an extension. If the alias _letter contains other editing commands, these commands are performed as part of the insertion. If the _letter alias is not enabled, this command has no effect.

Limits
Selecting a previous history line for editing while at a secondary prompt (that is, while entering a subsequent line of a new multiline command) yields unexpected results.
shift - Shift positional parameters

Format

```
shift [expression]
tcsh shell: shift [variable]
```

Description

**shift** renames the positional parameters so that \( i+n \)th positional parameter becomes the \( i \)th positional parameter, where \( n \) is the value of the given arithmetic *expression*. If you omit *expression*, the default value is 1. The value of *expression* must be between zero and the number of positional parameters (\( $# \)), inclusive. The value of \( $# \) is updated.

**shift** is a special built-in shell command.

In the tcsh shell, without arguments, **shift** discards \( argv[1] \) and shifts the members of \( argv \) to the left. It is an error for \( argv \) not to be set or to have less than one word as value. With *variable*, **shift** performs the same function on *variable*. See “tcsh: Invoke a C shell” on page 671.

Examples

The commands:

```
set a b c d
shift 2
echo $*
```

produce:

```
c d
```

Localization

**shift** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


Exit values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion</td>
</tr>
<tr>
<td>1</td>
<td>Failure because the <em>expression</em> had a negative value or was greater than the number of positional parameters.</td>
</tr>
</tbody>
</table>

Messages

Possible error messages include:
bad shift count expr

You specified an expression that did not evaluate to a number in the range from 0 to the number of remaining positional parameters.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

Allowing an expression, rather than just a number, is an extension that is found in the z/OS UNIX shell.

Related information

set, sh, tcsh

sleep - Suspend execution of a process for an interval of time

Format

sleep seconds

Description

sleep continues running until the specified number of seconds has elapsed. sleep can delay execution of a program or produce periodic execution in conjunction with shell commands.

The seconds argument can be either a number of seconds, or a more general time description of the form nhmns, with the nh, nm, and the s being optional.

Examples

sleep 20h10m

sleeps for 20 hours and 10 minutes (or 72600 seconds).

Localization

sleep uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Exit values

0

Successful completion

2

Failure because you specified no seconds value or because seconds is an incorrect argument (for example, incorrect format).

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.
Related information

date

---

**skulker** - Remove old files from a directory

**Format**

```
skulker [-iw][-r][-R] [-l logfile] directory days_old
```

**Description**

**skulker** finds files that are candidates for deletion in directory, based on comparing the file's access time to the age specified by **days_old**.

When you call **skulker** without any options, the files that are candidates for deletion are regular files found using the primaries as shown in the following **find** command line:

```
find directory -type f -atime +days_old -level 0 ! -name "*'*" ! -name "*" -print
```

The preceding **find** command has a deliberate newline inserted as part of the second -name request.

For example, specifying 5 for **days_old** causes the **find** command to find files that are equal to or older than five 24-hour intervals earlier than now.

This script ignores path names that contain single quotes or newlines when determining the list of objects that are candidates for deletion.

The **skulker** script (which is a z/OS shell script in /samples) should be copied and can be modified to suit your particular needs. Possible locations for placing the script include /bin or /usr/sbin, especially if **skulker** is to be run from a UID(0) program. If **skulker** is to be run by users, /usr/sbin is another possibility, but check that the sticky bit is on in the directory. If the script is called from a privileged user (a superuser, a user with a UID of 0, or a user running with the RACF trusted or privileged attribute), it is important to protect the script from any modifications by a non-privileged user.

The code page in which a shell script is encoded must match the code page of the locale in which it is run. For a shell script to be shared by multiple users, they must all be in a locale that uses the same code page as the code page in which the shell script is encoded. If you have different users operating in various locales, you need multiple copies of the skulker shell script, one for each different locale code page. You can use the **iconv** command to convert the skulker shell script from one code page to another.

**Options**

- **-i**
  Displays the files that are candidates for deletion, and prompts the user to stop or continue with file removal. Do not use this option if you are invoking **skulker** from a **cron** job. If **skulker** is invoked with `-i` from a **cron** job, no files will be deleted. A message will be mailed to the caller, showing the **skulker** output that includes the message "Request canceled".

- **-l logfile**
  Specifies a **logfile** to store a list of files that have been deleted, are candidates for deletion, or for which warnings have been mailed; and any errors that might have occurred.

- **-r**
  Moves recursively through subdirectories, finding non-directory files that are equal to or older than the specified number of days. The files that are candidates for deletion are found by using the primaries as shown in the following **find** command line:

```
find directory -atime +days_old ! -type d ! -name "*'*" ! -name "*" -print
```
The `find` command in the preceding example has a deliberate newline inserted as part of the second -name request.

The `-r` option is mutually exclusive with the `-R` option.

- **-R**
  Moves recursively through subdirectories, finding both non-directory files and subdirectories that are equal to or older than the specified number of days. Any subdirectories that are found as candidates for deletion are only deleted if they are empty after all their contents (files, subdirectories, and files in subdirectories) that are candidates for deletion have been deleted.

The files that are candidates for deletion are found by using the primaries as shown in the following `find` command line:

```bash
find directory -atime +days_old ! -name directory ! -name "!*" ! -name "+*
```

The `! -name directory` primary prevents `skulker` from deleting the actual directory that was entered as a start point (for example, `/tmp`).

The `find` command in the preceding example has a deliberate newline inserted as part of the third -name request.

The `-R` option is mutually exclusive with the `-r` option.

- **-w**
  Does not remove files, but sends a warning to the owner of each old file (using `mailx`) that the file is a candidate for deletion.

  `days_old`
  Specifies the age of the files you want to remove. For example, if you specify 100 for `days_old`, all files that were last accessed 100 or more days ago are marked as candidates for deletion.

  `directory`
  Specifies the directory in which to look for files.

  By default, files are removed from the specified directory based on access time and their status as regular files, and are removed only from the directory specified (not from any subdirectories).

**Examples**

1. To remove all regular files from `/tmp` that were last accessed 100 or more days ago:

   ```bash
   skulker /tmp/ 100
   ```

   The trailing slash in `/tmp` is necessary if `/tmp` is a symbolic link and you want to list or remove files from the directory the link points to, rather than the symbolic link itself. If `/tmp` (or the directory specified) is not a symbolic link, the trailing slash has no effect.

2. To remove all regular files from `/tmp` that were last accessed 11 or more days ago:

   ```bash
   > ls -ll /tmp
   total 48
   -rw------- 1 BILLYJC SHUT 0 Nov 10 06:00 10.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 11 06:00 11.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 12 06:00 12.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 13 06:00 13.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 14 06:00 14.txt
   -rw------- 1 SUPERID SHUT 0 Nov 15 06:00 15.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 16 06:00 16.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 17 06:00 17.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 18 06:00 18.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 19 06:00 19.txt
   > date
   Mon Nov 29 11:17:20 EST 1999
   > skulker -i /tmp/ 11
   -rw------- 1 BILLYJC SHUT 0 Nov 10 06:00 10.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 11 06:00 11.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 12 06:00 12.txt
   -rw------- 1 BILLYJC SHUT 0 Nov 13 06:00 13.txt
   ```
Do you really want to delete these files? If yes, answer [y|Y]. Any other response cancels your request.

y
Deleting files...
>
ls -ll /tmp
total 48
-rw-------   1 SUPERID  SHUT           0 Nov 15 06:00 15.txt
-rw-------   1 BILLYJC  SHUT           0 Nov 19 06:00 19.txt
>
Note that non-superuser BILLYJC (who issued the skulker command) was not able to delete the superuser's (SUPERID) file (15.txt), even though the find command issued from skulker returned 15.txt as a file name to delete.

3. The skulker script can be run from a cron job. To use the cron daemon to run the skulker script at 3:15 a.m. every Monday through Friday:

> crontab
    15 3 * * 1-5 /etc/skulker -l /usr/spool/cron/skulker.log /tmp/ 100
<control-D>
>
This example removes all regular files from /tmp that were last accessed 100 or more days ago. By default, cron sends the stdout and stderr of the command in a mail message to the user who submitted the cron job.

4. To remove all files from /tmp, including subdirectories, that were last accessed 30 or more days ago:

    skulker -R /tmp/ 30

Exit values

0
  Successful completion

1
  Either skulker did not find any files that are candidates for deletion, or an error occurred.

2
  There was a usage error.

Usage notes

skulker only uses the objects access time to determine which objects are candidates for deletion. If you want to use other criteria such as the modify time or the change time, you can modify your own version of skulker to include that criteria. For instance, to modify your version to include the modify time (which, for example, is updated when the object is written to), add the primary

-mtime +days_old

to the find requests within skulker. To include the change time, add the primary

-ctime +days_old

to the find requests within skulker.

Messages

Possible messages include:

directory is not a directory
  The object that is specified for directory is not a directory object.

find command returned nonzero exit status: return code
The find command returned a nonzero exit status: return code.

Error occurred during remove of file. Return code=return code.
The rm or rmdir command failed with return code while attempting to delete file.

file is in use, not removed.
Some other process was using this file. file cannot be removed.

sort - Start the sort-merge utility

Format
sort [-cmu] [-o outfile] [-t char] [-y[n]] [-zn] [-bdfiMnr] [-k startpos[,endpos]] ... [file ...]
sort [-cmu] [-o outfile] [-tchar] [-yn] [-zn] [-bdfiMnr] [+startposition [- -endposition]] ... [file ...]

Description
sort implements a full sort-and-merge utility. By default, it sorts according to all the information in the record, in the order given in the record.

sort operates on input files containing records that are separated by the newline character. When you do not specify either the -c or -m option, sort sorts the concatenation of all input files and produces the output on standard output. If you do not specify any files, sort reads from the standard input (stdin). If you specify - as one of the file names, sort reads from the standard input (stdin).

The following options select particular operations:

-c
Checks input files to ensure that they are correctly ordered according to the key position and sort ordering options that are specified, but does not modify or output the files. This option affects only the exit code.

-m
Merges files into one sorted output stream. This option assumes that each input file is correctly ordered according to the other options specified on the command line; you can check this with the -c option.

-u
Ensures that output records are unique. If two or more input records have equal sort keys, sort writes only the first record to the output. When you use -u with -c, sort prints a diagnostic message if the input records have any duplicates.

When you do not specify either the -c or the -m option, sort sorts the concatenation of all input files and produces the output on standard output.

Options
-o outfile
Writes output to the file outfile. By default, sort writes output to the standard output. The output file can be one of the input files. In this case, sort makes a copy of the data to allow the (potential) overwriting of the input file.

-t char
Indicates that the character char separates input fields. When you do not specify the -t option, sort assumes that any number of white space (blank or tab) characters separate fields.

-y[n]
Restricts the amount of memory available for sorting to n KB of memory (where a KB of memory is 1024 bytes). If n is missing, sort chooses a reasonable maximum amount of memory for sorting, dependent on the system configuration. sort needs at least enough memory to hold five records simultaneously. If you try to request less, sort automatically takes enough. When the input files
overflow the amount of memory available, `sort` automatically does a polyphase merge (external sorting) algorithm, which is, of necessity, much slower than internal sorting. \( n \) must be at least 2. \( n \) has a maximum value of 1024 and a default value of 56.

When you use `-u` with `-c`, `sort` prints a diagnostic message if the input records have any duplicates. Using the `-y` option might therefore improve sorting performance substantially for medium to large input files.

- `zn`  
  Indicates that the longest input record (including the newline character) is \( n \) bytes in length. By default, record length is limited to `LINE_MAX`.

The following options control the way in which `sort` does comparisons between records in order to determine the order in which the records are placed on the output. The ordering options apply globally to all sorting keys except those keys for which you individually specify the ordering option. For more information about sorting keys, see “Sorting keys” on page 629.

- `-b`  
  Skips, for comparison purposes, any leading white space (blank or tab) in any field (or key specification).

- `-d`  
  Uses dictionary ordering. With this option, `sort` examines only blanks, uppercase and lowercase letters, and numbers when making comparisons.

- `-f`  
  Converts lowercase letters to uppercase for comparison purposes.

- `-i`  
  Ignores, for comparison purposes, nonprintable characters.

- `-k [startpos [,endpos]]`  
  Specifies a sorting key. For more information, see “Sorting keys” on page 629.

- `-M`  
  Assumes that the field contains a month name for comparison purposes. Any leading white space is ignored. If the field starts with the first three letters of a month name in uppercase or lowercase, the comparisons are in month-in-year order. Anything that is not a recognizable month name compares less than `JAN`.

- `-n`  
  Assumes that the field contains an initial numeric value. `sort` sorts first by numeric value and then by the remaining text in the field according to options.

  Numeric fields can contain leading optional blanks or optional minus (-) signs. `sort` does not recognize the plus (+) sign.

  This option treats a field which contains no digits as if it had a value of zero. If more than one line contains no digits, the lines are sorted alphanumerically.

- `-r`  
  Reverses the order of all comparisons so that `sort` writes output from largest to smallest rather than smallest to largest.

**Sorting keys**

By default, `sort` examines entire input records to determine ordering. By specifying sorting keys on the command line, you can tell `sort` to restrict its attention to one or more parts of each record.

You can indicate the start of a sorting key with:

```
-k m[.n][options]
```

where \( m \) and the optional \( n \) are positive integers. You can choose `options` from the set `bdfiMnr` (described previously) to specify the way in which `sort` does comparisons for that sorting key. Ordering
options set for a key override global ordering options. If you do not specify any options for the key, the global ordering options are used.

The number \( m \) specifies which field in the input record contains the start of the sorting key. The character given with the \(-t\) option separates input fields; if this option is not specified, spaces or tabs separate the fields. The resulting sort key is from the \( m \)th field to the end of the record. The number \( n \) specifies which character in the \( m \)th field marks the start of the sorting key; if you do not specify \( n \), the sorting key starts at the first character of the \( m \)th field.

If an ending position for a key is not specified, the sorting key extends from the starting position to the end of the input record. You can also specify an ending position for a key, with:

\[-k\ m[.n][options],p[.q][options]\]

where \( p \) and \( q \) are positive integers, indicating that the sort key ends with the \( q \)th character of the \( p \)th field. If you do not specify \( q \) or if you specify a value of 0 for \( q \), the sorting key ends at the last character of the \( p \)th field. For example:

\[-k\ 2.3,4.6\]

defines a sorting key that extends from the third character of the second field to the sixth character of the fourth field. The \( b \) option applies only the key start or key end for which it is specified;

\[-k\ 2\]

defines a sorting key that extends from the first character of the second field to the end of the record;

\[-k\ 2,2\]

defines a sorting key that extends from the first character of the second field to the last character of the second field.

\( \text{sort} \) also supports a historical method of defining the sorting key. Using this method, you indicate the start of the sorting key with:

\[+m[.n][options]\]

which is equivalent to:

\[-k\ m+1[.n+1][options]\]

You can also indicate the end of a sorting key with:

\[-p[.q][options]\]

which when preceded with \(+m.n\) is equivalent to:

\[-k\ m+1[.n+1],p.0[options]\]

if \( q \) is specified and is zero. Otherwise,

\[-k\ m+1[.n+1],p+1[.q][options]\]

For example: \(+1.2\ -3.5\) defines a sorting key with a starting position that \( \text{sort} \) finds by skipping the first two characters of the next field and an ending position that \( \text{sort} \) finds by skipping the first three fields and then the first five characters of the next field. In other words, the sorting key extends from the third character of the second field to the sixth character of the fourth field. This is the same key as defined under the \(-k\) option, described earlier.

With either syntax, if the end of a sorting key is not a valid position after the beginning key position, the sorting key extends to the end of the input record.

You can specify multiple sort key positions by using several \(-k\) options or several \(+\) and \(-\) options. In this case, \( \text{sort} \) uses the second sorting key only for records where the first sorting keys are equal, the third
sorting key only when the first two are equal, and so on. If all key positions compare equal, sort determines ordering by using the entire record.

When you specify the -u option to determine the uniqueness of output records, sort looks only at the sorting keys, not the whole record. (Of course, if you specify no sorting keys, sort considers the whole record to be the sorting key.)

Examples

1. To sort an input file having lines consisting of the day of the month, white space, and the month, as in:

   30 December
   23 MAY
   25 June
   10 June

   use the command:

   sort -k 2M -k 1n

2. To merge two dictionaries, with one word per line:

   sort -m -dfi dict1 dict2 >newdict

Environment variables

sort uses the following environment variable:

TMPDIR
Contains the path name of the directory to be used for temporary files.

Files

sort uses the following file:

/tmp/stm*
Temporary files that are used for merging and -o option. You can specify a different directory for temporary files using the TMPDIR environment variable.

Localization

sort uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_NUMERIC
- LC_TIME
- NLSPATH

The -M option works only if LC_TIME identifies a locale that contains the same month names as the POSIX locale.


Exit values

0
Successful completion. Also returned if -c is specified and the file is in correctly sorted order.
1 Returned if you specified -c and the file is not correctly sorted. Also returned to indicate a non-unique record if you specified -cu.

2 Failure due to any of the following reasons:
   • Missing key description after -k
   • More than one -o option
   • Missing file name after -o
   • Missing character after -t
   • More than one character after -t
   • Missing number with -y or -z
   • endposition given before a startposition
   • Badly formed sort key
   • Incorrect command-line option
   • Too many key field positions specified
   • Insufficient memory
   • Inability to open the output file
   • Inability to open the input file
   • Error in writing to the output file
   • Inability to create a temporary file or temporary file name

Messages
Possible error messages include:

**Badly formed sort key position x**
   The key position was not specified correctly. Check the format and try again.

**File filename is binary**
   sort determined that filename is binary because it found a NULL (" ") character in a line.

**Insufficient memory for ...**
   This error normally occurs when you specify very large numbers for -y or -z and there is not enough memory available for sort to satisfy the request.

**Line too long: limit nn - truncated**
   Any input lines that are longer than the default number of bytes (LINE_MAX) or the number that is specified with the -z option are truncated.

**Missing key definition after -k**
   You specified -k, but did not specify a key definition after the -k.

**No newline at end of file**
   Any file not ending in a newline character has one added.

**Nonunique key in record ...**
   With the -c and -u options, a non-unique record was found.

**Not ordered properly at ...**
   With the -c option, an incorrect ordering was discovered.

**Tempfile error on ...**
   The named temporary (intermediate) file could not be created. Make sure that you have a directory that is named /tmp, and that this directory has space to create files. You can change the directory for temporary files using the TMPDIR environment variable.

**Tempnam() error**
   sort could not generate a name for a temporary working file. This should almost never happen.
Temporary file error (no space) for ...

Insufficient space was available for a temporary file. Make sure that you have a directory that is named /tmp, and that this directory has space to create files. You can change the directory for temporary files using the ROOTDIR and TMPDIR environment variables.

Too many key field positions specified

This implementation of sort has a limit of 64 key field positions.

Write error (no space) on output

Some error occurred in writing the standard output. Barring write-protected media and the like, this typically occurs when there is insufficient disk space to hold all of the intermediate data.

Portability

POSIX.2, X/Open Portability Guide.

Available on all UNIX systems, with only UNIX System V.2 or later having the full utility described here.

The -M, -y, and -z options are extensions of the POSIX standard.

Related information

awk, comm, cut, join, sort

The sortgen awk script is a useful way to handle complex sorting tasks. It originally appeared in The AWK Programming Language, by Aho, Weinberger, and Kernighan. The POSIX.2 standard regards the historical syntax for defining sorting keys as obsolete. Therefore, you should use only the -k option.

spell - Detect spelling errors in files

Format

spell [-biluvx] [-d hashfile] [-f local] [-h history] [+local] [file ...]

The spell utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, do not use this utility for applications that are intended to be portable to other UNIX-branded systems.

Description

spell checks for misspelled words in each specified file. If you do not specify a file, it checks the standard input. A list of potentially misspelled words is produced on standard output.

Words are checked against a local word list and then against a hashed word list. The hashed word list included in this distribution contains virtually no proper names or technical terms. It is assumed that you will enter these words into your local word list (or into your machine's word list). Any capitalized word in the hash list must be capitalized in the input document; all other words are matched either capitalized or not. All word forms, including plurals, must be explicitly included in the hash list. This approach prevents the acceptance of nonsense words that can result from the algorithmic combination of legal roots with legal suffixes or prefixes, a phenomenon common to many other spelling checkers.

Options

-b

Uses British spelling (such as “colour” instead of “color”). The dictionary file used is /usr/lib/hashb instead of /usr/lib/hashb.

-d hashfile

Uses hashfile as the dictionary. hashfile is a hash list produced from a list of words using the -i option of spell. To use a list other than the default /usr/lib/hashb, the -d option must be specified.
spell

- **f local**
  Uses the file `local` as a dictionary of local words, given one word per line. If you do not specify this option, the file `/usr/lib/lwords` is used as the local dictionary.

- **h history**
  Appends a history of all misspelled words to the file `history`. This file can be used by a system administrator for dictionary maintenance or generating a local dictionary.

- **i**
  Creates a new hash list file or add words to an existing file, instead of checking for spelling errors. Words to be entered into the dictionary should be specified one per line with no white space on the line. Lines beginning with the `#` character are ignored as comments. Be sure that the words you are entering into the hash list are correctly spelled.

- **l**
  Produces a longer form of output. For each misspelled word, `spell` prints three tab-separated columns containing the misspelled word, the line number, and the file name.

- **u**
  Forces `spell` to accept any word that is in all uppercase. `spell` assumes that such words are acronyms.

- **v**
  Writes to `stdout` all words not literally in the dictionary. This is the default for this implementation because it doesn't apply suffix/prefix rules to derive words.

- **x**
  Writes each plausible word stem to `stdout`. Because this implementation of `spell` doesn't derive words, all words are their own word stems.

+ **local**
  Uses the file `local` as a dictionary of local words, given one word per line. This is synonymous with `-f`.

**Examples**

By default, `spell` does not sort the output. This maintains the order and number of occurrences of spelling errors. The following command checks for spelling errors, puts them in dictionary order, removes duplicates, and print them in a multicolumn format:

```
spell file | sort -dfu | c
```

**Localization**

`spell` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_SYNTAX`
- `NLSPATH`


**Exit values**

0
   Successful completion

1
   Failure due to any of the following:
   - Missing `hashfile` name after `-d`. 
- Missing *history* file name after -h.
- Missing *local* file name after -f.
- Inability to open the *local* file
- Receipt of user interrupt
- An error reading the dictionary file
  A spelling mistake is not considered an error.

2
Incorrect command-line option

Files
spell uses the following files:

/usr/lib
- The default location of user hash files.
/usr/lib/hash
- The default dictionary file, in hashed form.
/usr/lib/hashb
- The British dictionary file, in hashed form.
/usr/lib/lwords
- The default location of the local words file. This need not exist.

Limits
Input lines in the text being checked are restricted to a maximum of 100 characters.

Portability
X/Open Portability Guide, UNIX systems.
The -d, -f, -h, -i, -l, and -u options are extensions of the POSIX standard.

Related information
sort, vi

**split - Split a file into manageable pieces**

**Format**

<table>
<thead>
<tr>
<th>Command</th>
<th>Options</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>split</code></td>
<td><code>[-a n] [-l n] [file [prefix]]</code></td>
<td><code>-b n[bkm] [-a n] [file [prefix]]</code></td>
</tr>
</tbody>
</table>

**Description**

`split` breaks up a file into a set of files. It starts a new file every time it has copied 1000 lines.

`split` names the files that it creates as a prefix followed by a suffix. *x* is the prefix unless you specify a different *prefix* on the command line. Unless altered by the following options, the suffix begins as aa and is incremented with each new file. By default, therefore, the first file is xaa followed by xab, and so on.

**Options**

- `-a n` Uses a suffix *n* letters long. The default is two.
-b n[bkm]
    Splits the file every n units. The default unit size is bytes. When you follow n with b, k, or m, split uses a corresponding unit size of 512 bytes, 1K (1024 bytes), or 1 megabyte (1 048 576 bytes).

-l n
    Splits the file every n lines.

-n
    Is an obsolete version of the -l option.

If the file is - (dash) or if no file is specified, split reads the standard input (stdin).

Localization
split uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH

Exit values
0
    Successful completion.
1
    Failure due to any of the following reasons:
    • Error opening input or output file.
    • Missing number after -a.
    • Incorrect -a option.
    • Missing byte count after -b.
    • Invalid byte count specification.
    • Invalid count specification.
    • Unknown option.
    • Out of memory for binary split buffer.
    • Read error on input file.
    • Write error on output file.
    • Too many names generated.

Portability
POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems,
The b suffix of the -b option is an extension to the POSIX.2 standard.

Related information
csplit
**stop - Suspend a process or job**

**Format**

```
stop [pid ...] [job-identifier ...]
tcsh shell: stop %job|pid ...
```

**Description**

`stop` is an alias for `kill –STOP`. Like `kill –STOP`, `stop` sends a SIGSTOP to the process you specify.

For more information, see “`kill - End a process or job, or send it a signal`” on page 364.

In the `tcsh` shell, `stop` stops the specified jobs or processes which are executing in the background. `job` can be a number, a string, '', %, + or -. There is no default `job`. Specifying `stop` alone does not stop the current job. See “`tcsh: Invoke a C shell`” on page 671.

**Options**

**job-identifier**

Is the job identifier reported by the shell when a process is started with &. It is one way to identify a process. It is also reported by the `jobs` command. When using the job identifier with the `stop` command, the job identifier must be prefaced with a percent (%) sign. For example, if the job identifier is 2, the `stop` command would be entered as follows:

```
stop %2
```

**pid**

Is the process ID that the shell reports when a process is started with &. You can also find it using the `ps` command. The `pid` argument is a number that can be specified as octal, decimal, or hexadecimal. Process IDs are reported in decimal. `stop` supports negative values for `pid`.

If `pid` is negative but not -1, the signal is sent to all processes whose process group ID is equal to the absolute value of `pid`. The negative `pid` is specified in this way:

```
stop -- --nn
```

where `nn` is the process group ID and can have a range of 2 to 7 digits (`nn` to `nnnnnnn`).

```
stop -- --9812753
```

The format must include the `--` before the `--nn` in order to specify the process group ID.

If `pid` is 0, the signal is sent to all processes in the process group of the invoker.

The process to be killed must belong to the current user, unless that user is the superuser.

**Related information**

`kill`, `jobs`, `sh`, `suspend`, `tcsh`

---

**strings - Display printable strings in binary files**

**Format**

```
strings [-aBoxz] [-n number] [-t format] [-W option[option] ... [file ... file ...]
strings [-] [-Boxz] [-t format] [-W option[option] ... [-number] [file ... file ...]
```
Description

If the command line specifies a file name of –, **strings** reads the standard input. **strings** finds pieces of information in binary files. It is frequently used for looking through executable files to uncover items such as copyright notices, error messages, and undocumented features.

The command displays strings of printable characters that are at least four characters in length. Strings must be terminated by a NUL character or by a newline.

Options

- **-a**
  This option has no effect in the z/OS environment. The entire file is examined, regardless of whether or not this option is specified.

- **-B**
  Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcode options (-W option) are specified.

- **-n number**
  Displays strings of printable characters that are at least number characters in length. If you do not specify the -n option, **strings** acts as if –n 4 had been specified.

- **-o**
  For each string, displays as an octal value its offset in bytes from the beginning of the file. This option is the same as -t o.

- **-t format**
  For each string, displays its offset in bytes from the beginning of the file. The base of the offset is set to decimal, octal, or hexadecimal by specifying format as d, o, or x, respectively.

- **-W option[,...]**
  Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

  - **filecodeset=codeset**
    Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command **iconv** -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcode options can be used on files with any file tag.
    
    If pgmcode is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcode is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see **Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993**.
    
    If filecodeset or pgmcode is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See **z/OS UNIX System Services Planning** for more information about automatic conversion.
    
    When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see **z/OS Unicode Services User's Guide and Reference**.

  - **pgmcode=codeset**
    Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command **iconv** -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcode options can be used on files with any file tag.
    
    If filecodeset is specified but pgmcode is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcode is specified, text conversion will not occur
unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User's Guide and Reference.

-x
For each string, displays as a hexadecimal value its offset in bytes from the beginning of the file. This option is the same as -t x.

-z
Ignores the POSIX definition of a string and searches for any group of printable characters greater than four in length.

- Is the obsolete version of -a.

- number
Is the obsolete version of -n number.

Examples
1. To display printable EBCDIC strings in a binary file to the standard output (stdout):
   strings myBinaryFile

2. To display printable ASCII strings in a binary file to the standard output (stdout):
   strings -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 myBinaryFile

3. To display printable EBCDIC strings in a binary file, assuming that automatic conversion has been enabled but the binary file is incorrectly tagged as an UTF-8 text file:
   strings -B myMisTaggedFile

Localization
strings uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Environment variables
strings uses the following environment variable:

_TEXT_CONV
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.
Exit values
0
   Successful completion.
1
   Failure due to any of the following reasons:
   • Incorrect command-line option.
   • Insufficient memory.
   • The code set is not valid.
   • Could not turn off automatic conversion.
   • Could not perform requested text conversion.

Portability
POSIX.2 User Portability Extension, X/Open Portability Guide.
A Berkeley addition to most UNIX systems. Most Berkeley versions do not require the terminating NUL or newline.
The -B, -o, -W, -x, and -z options are extensions of the POSIX standard.

strip - Remove unnecessary information from an executable file

Format
strip file

Description
On some UNIX systems, strip removes debug information from an executable. On z/OS, the debug information can only be removed by recompiling. strip does not modify the contents of any executable file; it is functionally equivalent to touch file.

Localization
strip uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH

Exit values
0
   Successful completion.
1
   Failure due to any of the following reasons:
   • file does not exist or could not be opened.
   • The user does not have write permission for file.
   • An error occurred while reading file.
• file is not an executable file.
• file is executable, but appears corrupted.

2
No file was specified on the command line.

Messages
Possible error messages include:

executable file file: No such file or directory
The input file does not exist. Check that the file name was entered correctly and that it exists.

file file1: Not an executable file
strip only operates on executable files.

Write permission required to strip file
The user does not have write permission on the file.

executable file file: Permission denied
The user does not have read permission on the file.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

stty - Set or display terminal options

Format
stty [-ag] [operand]

Description
stty sets or reports the terminal I/O characteristics for the standard input device. stty, entered without options or operands, reports only the terminal I/O characteristics that differ from the defaults. stty, entered with operands enables, disables, or selects the full range of terminal I/O characteristics.

The stty command affects whichever line discipline is in effect for your terminal.

Options
This command supports the following options:

-a
Displays all of the terminal I/O characteristics.

-g
Displays all of the terminal I/O characteristics in a format that can be used as input to the stty command.

The -a option gives you a clear readable description, whereas the -g option enables you to save and restore the terminal I/O characteristics.

stty entered with operands enables, disables, or selects the full range of terminal I/O characteristics.

Control mode operands
The valid operands for setting control modes are:

paxenb
Enable parity generation and detection. Not valid for z/OS line discipline. –paxenb is always used. See “Usage notes” on page 648.
-parenb
Disables parity generation and detection.

-parodd
Selects odd parity. Not valid for z/OS line discipline. -parodd is always used. See “Usage notes” on page 648.

-parodd
Selects even parity.

cs5
Selects character size CS5. Not valid for z/OS line discipline. CS8 is always used. See “Usage notes” on page 648.

cs6
Selects character size CS6. Not valid for z/OS line discipline. CS8 is always used. See “Usage notes” on page 648.

cs7
Selects character size CS7. Not valid for z/OS line discipline. CS8 is always used. See “Usage notes” on page 648.

cs8
Selects character size CS8.

-number
Sets the input and output baud rates to number. A number of zero hangs up the modem line.

-ispeed number
Sets the input baud rate to number. Not valid for z/OS line discipline. No special processing of zero is done. See “Usage notes” on page 648.

-ospeed number
Sets the output baud rate to number. Not valid for z/OS line discipline. No special processing of zero is done. See “Usage notes” on page 648.

-hupcl
Hangs up the modem line on the last close.

-hupcl
Do not hang up the modem line on the last close.

-hup
Hangs up the modem line on the last close.

-hup
Do not hang up the modem line on the last close.

-cstopb
Use two stop bits per character. Not valid for z/OS line discipline. -cstopb is always used. See “Usage notes” on page 648.

-cstopb
Use one stop bit per character.

cread
Enable the receiver.

cread
Disable the receiver. Not valid for z/OS line discipline. cread is always used. See “Usage notes” on page 648.

clocal
Assume a line without modem control.

clocal
Assume a line with modem control.

-columns number
Sets number of columns to number.
This should only be used if rlogin or telnet client does not support window size, or you are having trouble getting the correct size.

`row number`
Set number of rows to number.

This should only be used if rlogin or telnet client does not support window size, or you are having trouble getting the correct screen size.

**Input mode operands**
The valid operands for setting input modes are:

- **ignbrk**
  Ignore break on input.

- **-ignbrk**
  Do not ignore break on input.

- **brkint**
  Signal INTR on break.

- **-brkint**
  Do not signal INTR on break.

- **ignpar**
  Ignore parity errors.

- **-ignpar**
  Do not ignore parity errors.

- **parmrk**
  Mark parity errors.

- **-parmrk**
  Do not mark parity errors.

- **inpck**
  Enable input parity checking.

- **-inpck**
  Disable input parity checking.

- **istrip**
  Strip input characters to seven bits. This feature is required by the standards but do not use this setting. It makes it impossible to send EBCDIC alphanumerical characters to your shell session and you will have to take extreme measures to terminate the session.

- **-istrip**
  Do not strip input characters to seven bits. This is the default and should not be changed.

- **inlcr**
  Map newline to carriage return on input.

- **-inlcr**
  Do not map newline to carriage return on input.

- **igncr**
  Ignore carriage return on input.

- **-igncr**
  Do not ignore carriage return on input.

- **icrnl**
  Map carriage return to newline on input.

- **-icrnl**
  Do not map carriage return to newline on input.

- **iuclc**
  Map uppercase alphabetic characters to lowercase on input.
-iuclc
  Do not map uppercase alphabetic characters to lowercase on input.

ixon
  Enable START/STOP output control.

-ixon
  Disable START/STOP output control.

ixany
  Allow any character to restart input.

-ixany
  Do not allow any character to restart input.

ixoff
  Ask the system to send START/STOP characters to regulate the size of the input queue.

-ixoff
  Ask the system not to send START/STOP characters to regulate the size of the input queue.

Output mode operands
The valid operands for setting output modes are:

onlcr
  Converts newline characters to newline-carriage return sequences.

-onlcr
  Newline characters are displayed as newlines only.

opost
  Postprocess output.

-opost
  Do not postprocess output. Ignore all other output modes.

olcuc
  Map lowercase alphabetic characters to uppercase on output.

-olcuc
  Do not map lowercase alphabetic characters to uppercase on output.

ocrnl
  Map CR to NL on output.

-ocrnl
  Do not map CR to NL on output.

onocr
  Do not output CR at column 0.

-onocr
  Output CR at column 0.

onlret
  The terminal newline key performs the CR function.

-onlret
  The terminal newline key does not perform the CR function.

ofill
  Use fill characters for delays.

-ofill
  Use timing for delays.

ofdel
  Fill characters are DELs.

-ofdel
  Fill characters are NULs.
Sets the style of delay for CRs (CRDLY) to CR0.

Sets the style of delay for CRs (CRDLY) to CR1.

Sets the style of delay for CRs (CRDLY) to CR2.

Sets the style of delay for CRs (CRDLY) to CR3.

Sets the style of delay for NL (NDLY) to NL0.

Sets the style of delay for NL (NLDLY) to NL1.

Sets the style of delay for horizontal tabs (TABDLY) to TAB0.

Sets the style of delay for horizontal tabs (TABDLY) to TAB1.

Sets the style of delay for horizontal tabs (TABDLY) to TAB2.

Sets the style of delay for horizontal tabs (TABDLY) to TAB3.

Select the style of delay for backspace (BSDLY) to BS0.

Select the style of delay for backspace (BSDLY) to BS1.

Select the style of delay for form feeds (FFDLY) to FF0.

Select the style of delay for form feeds (FFDLY) to FF1.

Select the style of delay for vertical tabs (VTDLY) to VT0.

Select the style of delay for vertical tabs (VTDLY) to VT1.

Local mode operands

The valid operands for setting local modes are:

Enable character checking against the special control characters INTR, QUIT and SUSP.

Disable character checking against the special control characters INTR, QUIT and SUSP.

Enable canonical input mode.

Disable canonical input mode.

Set canonical uppercase or lowercase presentation.

Do not set canonical uppercase or lowercase presentation.

Enable any custom special control characters.
-iexten
   Disable any custom special control characters.

echo
   Echo every character typed.
-echo
   Do not echo every character typed.

echoe
   Enable the ERASE character to visibly erase the latest character.
-echoe
   Do not enable the ERASE character to visibly erase the latest character.
echok
   Echo newline after a KILL character.
-echok
   Do not echo newline after a KILL character.
echonl
   Echo newline (even when echo is disabled).
-echonl
   Do not echo newline when echo is disabled.

-noflush
   Disable flush after INTR, QUIT, and SUSP.
-noflush
   Enable flush after INTR, QUIT, and SUSP.
tostop
   Send the SIGTOU signal for background output.
-tostop
   Do not send the SIGTOU signal for background output.

Control character operands
In a double-byte environment, the char parameter to these operands must be a narrow (singlebyte) character.
The valid operands for assigning special control characters are:

min number
   Set min to number.
time number
   Set time to number.
eof char
   Set end of file character to char.
eol char
   Set end of line character to char.
erase char
   Set ERASE character to char.
intr char
   Set INTR character to char.
kill char
   Set KILL character to char.
quit char
   Set QUIT character to char.
susp char
   Set SUSP character to char.
**start char**
Set START character to `char`.

**stop char**
Set STOP character to `char`.

**Combination mode operands**
The valid operands for setting combination modes are:

**saved-settings**
Set the terminal I/O characteristics to the saved settings produced by the `-g` option.

**evenp**
Enable `parenb` and `cs7`; disable `parodd`. Not valid for z/OS line discipline. See “Usage notes” on page 648.

**parity**
Enable `parenb` and `cs7`; disable `parodd`. Not valid for z/OS line discipline. See “Usage notes” on page 648.

**oddp**
Enable `parenb`, `cs7` and `parodd`. Not valid for z/OS line discipline. See “Usage notes” on page 648.

**-parity**

**-evenp**

**-oddp**

**raw**
Enable raw input and output.

**-raw or cooked**
Disable raw input and output.

**nl**
Enable `icrlf`.

**-nl**
Disable `icrlf`; unset `inlcr` and `igncr`.

**lcase**
Set `xcase`, `iuccl`, and `olcuc`.

**-lcase**
Disable `xcase`, `iuccl`, and `olcuc`.

**LCASE**
Equivalent to `lcase`.

**-LCASE**
Equivalent to `-lcase`.

**tabs**
Preserve tabs when printing.

**-tabs or tab8**
Expand to spaces when printing.

**ek**
Reset ERASE and KILL characters to system defaults.

**sane**
Reset all modes to reasonable values.
Usage notes

1. \texttt{stty} will operate successfully even if it is unable to perform one or more actions in a group of requested actions. For example, if a valid z/OS operand is requested with an invalid one, \texttt{stty} will operate successfully because it can perform the valid operand. The valid operand will then be satisfied.

2. If \texttt{stty} is only used with invalid z/OS operands or invalid operands in combination with valid operands that have already been satisfied, \texttt{stty} will fail.

Localization

\texttt{stty} uses the following localization environment variables:

- \texttt{LANG}
- \texttt{LC_ALL}
- \texttt{LC_CTYPE}
- \texttt{LC_MESSAGES}
- \texttt{LC_SYNTAX}
- \texttt{NLSPATH}


Exit values

0

Successful completion.

1

Failure due to any of the following reasons:

- Error setting termios attributes.
- Unknown mode.
- Missing number after option.
- Argument out of range.
- Bad number after option.
- Internal error.
- Error reading termios attributes.
- Missing character after option.
- Badly formed argument option character
- Missing speed after \texttt{ispeed} or \texttt{ospeed}.
- Bad speed argument.

Portability

POSIX.2, UNIX System V.

\textbf{su - Change the user ID associated with a session}

Format

\texttt{su [-] [-s][userid [arg ...]]}

Description

\texttt{su} starts a new shell and lets you operate in it with the privileges of a superuser or another user.
If you do not specify a user ID, su changes your authorization to that of the superuser. The resulting MVS user ID can be any UID(0) user ID that is in the security product database. The security product returns the first UID(0) user ID that is found; this user ID can change over time as the cached information of the security product is updated.

If you specify a user ID, su changes your authorization to that of the specified user ID. The new environment is built and then a new session is initiated. The new session is run as a child shell of the shell issuing the su command.

Any arguments specified by arg are passed to the child shell, so must be valid invocation flags or arguments that are accepted by the child shell.

su performs these functions:

• Obtains your user profile information. After validating that you have an OMVS segment in the user profile, the OMVS segment information is obtained.

• Verifies authorization. If a user ID is not specified, you must have the appropriate authorization to obtain superuser authority. You must be permitted to the BPX.SUPERUSER resource in the FACILITY class.

If a user ID is specified, and you do not have read access to the SURROGAT class profile, BPX.SRV.uuuuuuuu (where uuuuuuuu is the MVS user ID associated with the target UID), you must enter the target user's password or password phrase when prompted. If a user ID is specified, and you have read access to the SURROGAT class profile for the target user, you can use the -s option, or press Enter at the password prompt.

• Changes the group ID. If a user ID is specified, the group ID is changed to that of the specified user's default group GID.

If a user ID is specified, the supplementary group list is changed to that of the specified user.

If the change of group ID or supplemental group list fails, the su command issues a message and continues.

• Changes the user ID. Your user ID might be changed to either the specified user ID or the superuser's user ID (UID 0).
  - When a user ID is specified, your MVS identity changes to the specified user ID, changing your access authority for MVS data sets in addition to changing to the new user's UID.
  - When a user ID is not specified, your MVS identity remains the same. This maintains your access authority to MVS data sets, while gaining superuser authority.
  - If you are already running under UID 0 and BPX.DAEMON is defined, issuing su with no userid will result in your UID being switched to BPXROOT. If BPX.DAEMON is not defined, and you issue su with the userid while running under UID 0, your UID will remain set to 0. In both cases, access to the BPX.SUPERUSER resource in the FACILITY class will not be checked.

• Sets up the shell environment. If the login shell ('-' flag) is specified, the OMVS segment of the new user is used to set up the shell environment, similar to user login processing. When a user ID is not specified, the new UID(0) user as found by the security product is used. This includes setting the SHELL, HOME, and LOGNAME environment variables. PATH is set to the system default (/bin), TERM is preserved from the current environment, and STEPLIB is set to "none". Other environment variables are not inherited by the new shell.

If the login shell is not specified, the OMVS segment of your user profile is used to set up the shell environment. The environment is set up to be as similar as possible to the environment of the shell issuing the su command. Existing values of HOME, LOGNAME, and PATH are preserved. If not set in the current shell environment, HOME and LOGNAME are set from the calling user's profile, and PATH is set to the system default (/bin). SHELL is set to calling user's profile value, or the default /bin/sh, if not defined.

• Executes the new shell. If login shell ('-' flag) is specified, prepend '-' to the shell's name. This indicates that the shell should read its login startup files (for example, /bin/sh will read /etc/profile and $HOME/.profile). The new shell is initialized to run as a child process of the shell issuing the su
command. If the `su` command is run from a restricted shell (such as a shell that was started with the `-r` option), you will exit from the restricted shell and leave the protection of the trusted environment.

**Note:**

1. The new shell is always run in a new address space, even if you have `_BPX_SHAREAS=YES` set.
2. If you use the OMVS interface when running a shell created by `su`, any attempt to execute TSO commands (PF6) results in the command running back in your TSO address space. When these TSO commands run, they run with your TSO identity, not the identity specified by `su`.

   If you are not using the OMVS interface (for example, you rlogin or telnet into the shell), you cannot use PF6 to execute a TSO command. As a result, there will be no TSO address space or identity. The alternative solution is to use `tso -t` or `tsocmd`, which allows you to run a TSO/E command with the current identity set by `su`.

To restore the previous session, enter `exit` or press `<EscChar-D>` (where EscChar is normally the cent sign). If you use rlogin or telnet to enter the shell, you hold down the Ctrl key while you press D. This action ends the child shell initiated by the `su` command and returns you to the previous shell, user ID, and environment. See *z/OS UNIX System Services User’s Guide* for more information about exiting the shell environment.

**Options**

-     
   Starts the new shell as a login shell. Sets the shell variables SHELL, HOME, and LOGNAME according to the new user's profile, and prepends a `-' to the shell name to indicate that the shell should read its login profiles. When a user ID is not specified, the new UID(0) user as found by the security product is used.

-`s`  
   Does not prompt for password or password phrase. If a user ID is specified, you must have read access to the SURROGAT class profile, BPX.SRV.uuuuuuuuu (where `uuuuuuuuuu` is the MVS userid associated with the target UID).

**Examples**

To switch to the admin user ID, but maintain the current user's shell environment:

```
su admin
```

To authorize a user to switch to another user without entering a password or password phrase, grant them RACF SURROGAT authority:

```
RDEFINE SURROGAT BPX.SRV.ADMIN UACC(NONE)
PERMIT BPX.SRV.ADMIN CLASS(SURROGAT) ID(FRED) ACCESS(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

Then, from Fred, issue:

```
su -s admin
```

To start a child shell with the login environment of the admin user ID:

```
su - admin
```

To run the `/usr/lib/backupall` script under the admin user ID and return to the parent shell environment when the script completes:

```
su admin /usr/lib/backupall
```
To run a remove shell command under the admin user ID and return to the parent shell environment when the command completes:

```
su admin -c "rm -rf /tmp/
```

**Usage notes**

1. The new shell inherits the standard file descriptors from the su command, so commands can be piped to the stdin of the new shell and run under the new user.
2. If the OMVS NOECHO option is in effect, your password or password phrase is displayed.
3. Because su starts a new interactive shell, it should not be used from a batch interface such as BPXBATCH, unless you provide the commands to be executed under superuser via stdin to the su command.
4. After issuing su -s in the shell to switch to another user, the new user will not have the authority to issue any commands that require an implicit open() of a tty. This restriction includes calls which invoke the Binder (such as cp -X and c89) as well as explicit attempts at opening a file descriptor (such as cat /dev/fd2). An ICH408I message is written to the console to alert the user of the access violation.

**Exit values**

0  
The command completed successfully.
1  
The user is not authorized to obtain superuser authority.
2  
Failure due to any of the following reasons:
   • Unable to execute the shell.
   • The OMVS segment of the user’s profile cannot be found.
   • Unable to set up the superuser environment.
3  
Failure due to any of the following reasons:
   • Incorrect command syntax.

**Limitations**

Only users who have RACF access permission to the superuser class can use su without specifying the user ID.

**Portability**

None. This command is an extension that comes with z/OS UNIX services.

**Related information**

sh, ISHELL

---

**Submit - Submit a batch job for background processing**

**Format**

```
submit [-jq] [filename ...]
```
**Description**

The `submit` command sends a batch job to the primary job entry subsystem (JES) for processing. The `filename` argument can either be a UNIX path name, data set, or input from standard input (stdin).

- If the `filename` argument is a sequential data set, `filename` is specified as:
  
  ```
  '//full.data.set.name'
  ```

- If the `filename` argument is a partitioned data set, `filename` is specified as
  
  ```
  '//full.pds.name(member)'
  ```

  where `member` is the member name of the partitioned data set `FULL.PDS.NAME`.

If no argument is given, `submit` reads from standard input. When standard input is being used, enter `''//`` or Ctrl-D to end the input.

If the `submit` command is successful, the job ID of the submitted job and the file name are displayed as part of the output written to standard output, unless modified by the `-j` option.

The JCL must have a job card. If a job card is not present, the `submit` command will fail.

**Options**

- `-j`
  
  Displays only the job ID when the `submit` command is successful.

- `-q`
  
  Quiet mode. In quiet mode, all error messages are suppressed and the exit status is preserved.

**Examples**

1. To submit a job that resides in a z/OS UNIX file:

   ```
   submit buildjcl.jcl
   ```

   Output to standard output:

   ```
   JOB JOB05990 submitted from path 'buildjcl.jcl'
   ```

2. To submit a job that resides in a partitioned data set:

   ```
   submit '//POSIX.RTL.UT13.JCL(TESTJCL)'
   ```

   Output to standard output:

   ```
   JOB JOB06075 submitted from data set 'POSIX.RTL.UT13.JCL(TESTJCL)'
   ```

3. To submit a job from standard input, you can specify the following command:

   ```
   cat test.jcl | submit
   ```

   Output to standard output:

   ```
   JOB JOB21532 submitted from stdin
   ```

**Exit values**

- `0`
  
  All jobs were submitted successfully.

- `1`
  
  One or more jobs were not submitted.
Incorrect command-line arguments or options.

Usage notes
1. The `submit` shell command uses the REXX submit function, which is described in *z/OS Using REXX and z/OS UNIX System Services*.

Localization
The `submit` command is not sensitive to a user's locale. It is up to the user to provide input that is acceptable to JES.

Limitations
1. JOBNAME can only be specified from within the JCL.
2. Only fully qualified data set names are accepted.
3. This version of `submit` cannot be run directly from TSO.
4. If multiple job cards are present in a file, `submit` only displays the job ID of the last job submitted within that file.
5. If a PDS member is specified, and the data set exists but the member does not, it produces the following messages to standard output:
   - IRX0250E System abend code 013, reason code 00000024.
   - IRX0255E Abend in host command execio or address environment routine MVS.
   - IRX0670E EXECIO error while trying to GET or PUT a record.
   `submit` then attempts to submit an empty job, and produces message FSUMB409 on stderr.
6. Do not include ANSI control characters in data sets (for example, RECFM=FBA) that are submitted to JES because they will not be accepted.

**sum - Calculate and display checksums and block counts**

Format

```
sum [-ciprtT] [file...]
```

The `sum` utility is fully supported for compatibility with older UNIX systems. However, use `cksum` utility because it may provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

Description

`sum` calculates and displays a checksum for each input file. A checksum is an error-checking technique used by many programs as a quick way to compare files that have been moved from one location to another to ensure that no data has been lost. It also displays the number of 512-byte blocks in each file. If you do not specify any files, or if you specify - as the file name, `sum` reads standard input (stdin).

`sum` differs from `cksum` only in the format of the output.

- When _UNIX03 is YES, the output of `sum` has the space-separated form:
  ```
  checksum  blockcount  filename
  ```
- When _UNIX03 is unset or not YES, the output of `sum` has the tab-separated form:
  ```
  checksum    blockcount    filename
  ```

If a file operand is not specified, the path name and its leading white space is omitted.
Read error messages are controlled by the _UNIX03 variable.

- If sum fails with a read error and _UNIX03 is YES, it sends a diagnostic message to standard error, and will not show a checksum for that file.
- If _UNIX03 is unset or not YES, it displays the checksum up to that point and marks the output line with FSUM6199 [read].

sum continues processing files in either case.

All other error messages are sent to standard error.

Options

sum can calculate checksums in a variety of ways. The default checksum algorithm produces a 16-bit unsigned integer resulting from the arithmetic addition of each input byte. This checksum algorithm is not sensitive to byte order.

- **c**
  Uses a standard 16-bit cyclic redundancy check (CRC-16).

- **i**
  Uses the CCITT standard cyclic redundancy check (CRC-CCITT). Data communications network protocols often use a cyclic redundancy check to ensure proper transmission. This algorithm is more likely to produce a different sum for inputs which differ only in byte order.

- **p**
  Uses the POSIX.2 checksum algorithm.

- **r**
  Enables the use of an alternate checksum algorithm which has the advantage of being sensitive to byte order.

- **t**
  Produces a line containing the total number of blocks of data read, as well as the checksum of the concatenation of the input files.

- **T**
  Enables the automatic conversion of tagged files.

Environment variables

- **_UNIX03**
  For more information about the effect of _UNIX03 on the sum command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Localization

sum uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_CTYPE**
- **LC_MESSAGES**
- **LC_TYPE**
- **NLSPATH**


Exit values

- **0**
  Successful completion.
Failure due to any of the following reasons:

- Inability to open input file.
- Error reading the input file.
- Error turning off the automatic conversion of the input file.

Unknown command-line option.

**Portability**

The default checksum algorithm is compatible with UNIX System V.2 and later. The -r algorithm is also available on UNIX System V.2 and is the default algorithm for Berkeley and Version 7. The -c, -i, and -t options are not available under UNIX.

**Related information**

cmp, cksum, diff, ls, wc

---

**suspend - Send a SIGSTOP to the current shell**

**Format**

suspend
tcsh shell: suspend

**Description**

suspend is an alias for stop $$, where stop is an alias of kill -STOP and $$ expands to the current process of the shell. suspend sends a SIGSTOP to the current shell.

See “kill - End a process or job, or send it a signal” on page 364 for more information.

In the tcsh shell, suspend causes the tcsh shell to stop in its tracks, much as if it had been sent a stop signal with ^Z. See “tcsh: Invoke a C shell” on page 671.

**Related information**

kill, sh, tcsh

---

**sysvar - Display static system symbols**

**Format**

sysvar var
sysvar 'var(start[:length])'

**Description**

The sysvar command allows users to obtain substitution text for system variables that are defined in the IEASYMxx member of SYS1.PARMLIB or in the system IPL parameters. The substitution text is printed to standard output (stdout). The substitution text can be used to substitute system variables in shell variables.
Substrings of system variables can be accessed by specifying the starting position and, optionally, the number of characters, which are enclosed in parentheses and separated by a colon. When \textit{length} is not specified, \texttt{sysvar \textquotesingle var\text{(start)}\textquotesingle} only prints the character in position \texttt{start}. For example, if the system symbol \texttt{SYSNAME} is assigned the substitution text \texttt{SYS1A}, \texttt{sysvar \textquotesingle SYSNAME(4)\textquotesingle} will print \texttt{1}.

\textbf{Notes:}

1. The name of the system symbol must be specified in uppercase letters.
2. Quoting or escaping must be applied if substring notation is used or if the symbol contains other special characters that are recognized by the shell.
3. Refer to \textit{z/OS MVS Initialization and Tuning Reference} for more information about system symbols and \texttt{sysvar \textquotesingle var\text{(start[:length])}\textquotesingle}.

\textbf{Examples}

1. Assuming that the system symbol is defined as \texttt{SYSA1}:

   \begin{verbatim}
   system_name=$(sysvar SYSNAME)
   echo $system_name
   ==> SYSA1
   
   system_name_short=$(sysvar 'SYSNAME(4:2)')
   echo $system_name_short
   ==> A1
   \end{verbatim}

\textbf{Exit values}

0  
Successful completion.

1  
Failure because \texttt{var} is not a valid system variable.

2  
Failure because no \texttt{var} was specified.

\section*{tabs - Set tab stops}

\textbf{Format}

\begin{verbatim}
  tabs [+m\[margin\]] [-T \text{term}] [- \text{number}]
  tabs [+m\[margin\]] [-T \text{term}] -t \text{tablist}
  tabs [+m\[margin\]] [-T \text{term}] \text{num1},\text{num2},...]
  tabs [+m\[margin\]] [-T \text{term}] \text{tabspec}
\end{verbatim}

\textbf{Description}

\texttt{tabs} sends a series of characters to the standard output, designed to clear the terminal hardware's tab stops and then set new ones. The characters that are sent depend on the type of terminal you are using. The first column of your terminal screen is column \texttt{1}. If you set a tab stop at position \texttt{N} and then tab to that position, the next character displayed on the screen appears in column \texttt{N+1} of the line (that is, after the tab stop).

\texttt{tabs} may not be able to set the tab stops on some types of terminals. In this situation, it issues an error message and then exits with a status greater than zero. \texttt{tabs} with no arguments sets tab stops every 8 positions.
Options

+m[margin]
Sets the left margin to margin. It defaults to 10 if you do not specify a value. All tab positions are relative to the left margin. To find the actual tab positions, you add the value of margin to each tab position.

-T type
Indicates the type of terminal you have. The term argument is a site-specific name for your terminal type.

If you do not specify -T, tabs looks for an environment variable that is named TERM and uses its value for type. If TERM is not defined, tabs assumes a default terminal type.

-t tablist
Sets tab stops as specified by tablist. tablist consists of one or more positive decimal integers, separated by commas; the numbers in the list should be in strictly increasing order.

If only one number N is given, tabs are set every N columns. If more than one number is given, tabs are set at those column numbers.

num1,[num2,...]
Sets tab stops to the given numbers. The numbers in the list should be positive decimal integers in strictly increasing order. Except for the first number, any number in the list may be preceded by a plus sign (+), in which case the number is considered to be an increment on the previous setting rather than a column position. For example,

tabs 4,8,12
   tabs 4,+4,+4

are equivalent.

tabspec
Can be one of -a, -a2, -c, -c2, -c3, -f, -p, -s or -u and sets tab stops at these positions:

-a
  1,10,16,36,72

-a2
  1,10,16,40,72

-c
  1,8,12,16,20,55

-c2
  1,6,10,14,49

-c3
  1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67

-f
  1,7,11,15,19,23

-p
  1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61

-s
  1,10,55

-u
  1,12,20,44

Each tabspec is designed for a particular programming language. Assembler uses -a, -a2, and -u. COBOL uses -c, -c2, and -c3. FORTRAN, PL/I, and SNOBOL use -f, -p, and -s, respectively.

-number
Sets tab stops every number positions along the line. number must be a single-digit decimal number. If number is zero (-o), tabs clears all the tab stops and does not set new ones.
Environment variables

**tabs** uses the following environment variables:

**TERM**
Contains the name of your terminal.

**TERMINFO**
Contains the path name of the terminfo database.

Localization

**tabs** uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_CTYPE**
- **LC_MESSAGES**
- **NLSPATH**


Exit values

0
Successful completion

1
Missing definition in the terminfo database.

2
Usage error.

3
Unknown terminal or cannot find the terminfo database.

4
Illegal tabs.

5
An error occurred.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
The +m, -t, and **tabspec** arguments are all extensions to the POSIX standard.
The -t argument is an extension to the X/Open standard.

Related information

**stty**

**tail: Display the last part of a file**

Format

```bash
    tail [-B] [-f] [-W option[option]] ... [-b|-c|-l|k|-m|-n [±]number] [file]
```

```bash
    tail [-B] [-W option[option]] ... ±[number][b | c | k | l | m | n][f] [file]
```
Description

tail without options displays the most recent entries in log files and any file where new information is added on the end.

The tail command is used with text files. To make a binary file input to the tail command, use the -c option. If a binary file is input without the -c option being specified, the entire file is sent to the screen.

If you do not specify a file, or if you specify - as the file name, tail reads from the standard input (stdin).

Options

+-number
   Is either of the following:

+number
   Skips to line number and then displays the rest of the file. For example, +100 prints from line 100 to the end of the file.

-number
   Prints number lines from the end of the file. For example, -20 prints the last 20 lines in the file.

You can precede or follow both +number and -number with one of the following letters to indicate the unit to be used:

• b-blocks
• c-bytes
• k-kilobytes
• l or n-lines
• m-megabytes

The default unit is lines.

-B
   Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-f
   Monitors a file as it grows. Every two seconds, tail wakes up and prints any new data at the end of the file. This option is ignored if tail read from the standard input, and standard input is a pipe.

-W option[,...]
   Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

filecodeset=odeset
   Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset.codeset can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command icnv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

   If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

   If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

   When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.
**pgmcodeset=codeSet**

Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is `codeSet`. `codeSet` can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The `filecodeset` and `pgmcodeset` options can be used on files with any file tag.

If `filecodeset` is specified but `pgmcodeset` is omitted, then the default program code set is IBM-1047. If neither `filecodeset` nor `pgmcodeset` is specified, text conversion will not occur unless automatic conversion is enabled or the `_TEXT_CONV` environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If `filecodeset` or `pgmcodeset` is specified, then automatic conversion is disabled for this command invocation and the `-B` option is ignored if it is also specified. See `z/OS UNIX System Services Planning` for more information about automatic conversion.

The only supported values for `pgmcodeset` are IBM-1047 and 1047.

**Examples**

1. To display the last 10 lines of a text file to the standard output (stdout):

   ```
   tail myTextFile
   ```

2. To display the last line of a text file and then monitor the file for updates (display new data that is written to the text file):

   ```
   tail -f -n -1 myTextFile
   ```

3. To display the last 200 bytes of a text file containing UTF-8 characters to the standard output (stdout), assuming that:
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file):

   ```
   tail -W filecodeset=UTF-8,pgmcodeset=IBM-1047 -c -200 myUtf8File
   ```

4. To display a text file containing EBCDIC characters starting at line 25, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as ASCII:

   ```
   tail -B -n +25 myMisTaggedFile
   ```

**Localization**

tail uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Environment variables**

tail uses the following environment variable:
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, "Controlling text conversion for z/OS UNIX shell commands," on page 993.

Exit values

0
Successful completion

1
Failure due to any of the following reasons:

- Insufficient memory.
- Write error on the standard output (stout).
- Badly formed line or character count.
- Missing number after an option.
- Error reopening a file descriptor.
- The code set is not valid.
- Could not turn off automatic conversion.
- Could not perform requested text conversion.

2
Failure due to an unknown command-line option.

Messages
Possible error messages include:

**Badly formed line/character count string**
In an option of the form –n number or –number, the number was not a valid number.

**Reopening file descriptor number**
–f was used to follow a file as it grew. tail closed the file that is associated with the given file descriptor number and then tried to open it 2 seconds later. At this point, tail found it could not reopen the file for reading, and therefore could not follow the file any longer.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

The POSIX standard included only the -f, -c+ -number, and -n±number options. The use of b, c and l as a + -number suffix is considered obsolete. All other options and suffixes are extensions of the POSIX standard.

Related information
cat, head, more

talk - Talk to another user

Format
talk address [terminal]
Description
Use the `talk` command to begin a two-way conversation with someone else logged in to the system. It reads from the standard input (stdin).

Options
`address`
Indicates the user with whom you want to talk. The most common form of address is the person's user name (as given by the `who` command), but other formats might be supported.

`terminal`
An optional identifier for use when the other user is logged in on more than one terminal. The format of the `terminal` identifier is the same as given by `who`.

Environment variables
`talk` uses the following environment variables:

- **TERM**
  Contains the name of your terminal.

- **TERMINFO**
  Contains the path name of the terminfo database.

Localization
`talk` uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_CTYPE**
- **LC_MESSAGES**
- **NLSPATH**


Usage notes
1. When you issue a `talk` command to request a conversation with another user, the other user receives a message of the form:

   Message from name
   talk: connection requested by your_address
   talk: respond with: talk your_address

   To set up the connection, your intended recipient must issue the system command

   talk your_address

   which establishes the two-way connection. After this connection has been established, both of you can type simultaneously. `talk` displays incoming messages from the other person in one part of the screen and outgoing messages in another part of the screen.

   Some terminals might not be able to split the screen into parts in this way. Depending on the terminal type, `talk` might try to simulate this effect. However, it may not be possible for both users to enter messages simultaneously. `talk` determines terminal type by looking for an environment variable named TERM. If this variable exists, `talk` uses its value as a site-specific name giving a terminal type. If TERM doesn't exist, `talk` assumes a default type.

2. The character-erase and line-kill characters work as usual. Typing <Ctrl-L> refreshes both parts of the screen (for example, if some unusual character messes up the display).
3. The interrupt character (for example, <Ctrl-C>) terminates your talk session and breaks the connection. When one side breaks the connection, talk notifies the other side and exits.

4. The msg command lets you refuse talk sessions. With:

```
msg n
```

you can tell the system that you don’t want to be interrupted by talk requests. If people try to establish a talk session with you, they are denied immediately; you are not informed about such requests. For more details, see msg.

**Exit values**

The following exit status values are possible:

- `0`
  - Successfully established and completed a transmission.
- `>0`
  - An error occurred, or you are trying to use talk on a terminal that cannot handle the way talk uses the screen.

**Portability**

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.

**Related information**

mail, msg, who, write

---

### tar: Manipulate the tar archive files to copy or back up a file

**Format**

```
tar [-cf[#sbvwlzOUXSG] tarfile [blocksize] [-V volpat] [file [-C pathname] ...]
tar -rf[#sbvwlzOUXSG] tarfile [blocksize] [-V volpat] [file [-C pathname] ...]
tar -tf[#sbvzE0UXSG] tarfile [blocksize] [-L type] [-V volpat] [file [-C pathname] ...]
tar -xf[#sAbvwpmozOUXSG] tarfile [blocksize] [-V volpat] [file [-C pathname] ...]
```

**Description**

tar reads, writes, and lists archive files. An archive file is a single file that contains one or more files, directories, or both. Archive files can be UNIX files or MVS data sets. A file that is stored inside an archive is called a component file. Similarly, a directory that is stored inside an archive is called a component directory.

**Restriction:** Note the following restrictions:

- MVS data sets cannot be specified for component files.
- **tar** supports the use of generation data groups (GDGs) and tape data sets.

Included with each component file and directory is recorded information such as owner and group name, permission bits, file attributes, and modification time. Therefore, you can use a single archive file to transfer file directories or to back up or restore groups of files and directories.

Archives that are created by **tar** are interchangeable with archives that are created with the pax utility. Both utilities can read and create archives in the default format of the other (USTAR for **pax** and TAR for **tar**). To save extended USTAR attributes, the USTAR format (-U) must be used with -X option. Also, the OS390 format can be used by using the -S option. In general, the USTAR format with -X option and OS390 format records the most information and is recommended. Archives are generally named with
suffixes such as .pax or .tar (or pax.Z and tar.Z for compressed files), but that naming convention is not required.

**tar** can read and write the GNU tar format archives. For the special entry types in GNU tar format, **tar** only supports the read and write long pathname or long linkname entries. Other special entry types in the GNU tar, such as archiving the sparse file, are not supported.

Table 31 on page 664 lists the recommended options for the USTAR format.

<table>
<thead>
<tr>
<th>Intent</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>To save only standard attributes</td>
<td>tar -U</td>
</tr>
<tr>
<td>To save all attributes to be restored on z/OS system</td>
<td>tar -S</td>
</tr>
<tr>
<td>To save all attributes to be restored on z/OS and non-z/OS systems</td>
<td>tar -UX</td>
</tr>
<tr>
<td>To save in GNU tar format</td>
<td>tar -G</td>
</tr>
</tbody>
</table>

To preserve information such as extended attributes, external links, ACLs, file tag information, and links whose targets exceed 100 characters, either the USTAR (-U) option or the OS390 format using the -S option must be used. See the -U option for selecting the USTAR format. The -0 and -X options and “z/OS-extended USTAR support” on page 532 contain information about enabling and disabling USTAR support.

You cannot use **tar** unless you specify -f.

### Options

The four forms of the command that is shown in the syntax represent the main functions of **tar** as follows:

- **-c**
  Creates an archive. Each named file is written into a newly created archive. Directories recursively include all components. Under the USTAR (-U) option, **tar** records directories and other special files in the tape archive; otherwise, it ignores such files. If - appears in place of any file name, **tar** reads the standard input for a list of files one per line. This allows other commands to generate lists of files for **tar** to archive. **Tip:** To preserve information about extended attributes and external links, the USTAR format (-U) must be used. Additionally, to preserve ACLs, file tag information, and link names greater than 100 characters, the USTAR format (-U) and -X option must be used. The OS390 archive format can also be used with the -S option to store all the file attributes.

- **-r**
  Writes the named files to the end of the archive. It is possible to have more than one copy of a file in a tape archive using this method. To use this form of the command with a tape, it must be possible to backspace the tape. Do not specify OS390 format to be appended to non-OS390 format archive or specify non-OS390 format to be appended to OS390 format archive.

  **Restriction:** You cannot specify both the -r and the -z option at the same time.

- **-t**
  Displays a table of contents. This option displays the names of all the files in the archive, one per line. If you specify one or more files on the command line, **tar** prints only those file names. The verbose (-v) option can be used to show the attributes of each component. For USTAR or OS390 format archives, the -L E option can be used to show the attributes and extended attributes of each component.
Extracts files from an archive. `tar` extracts each named file to a file of the same name. If you did not specify any files on the command line, all files in the archive are extracted. This extraction restores all file system attributes as controlled by other options.

You must specify one of the preceding basic options as the first character of an option string. You can add other characters to the option string. You can omit the leading dash on the first option string, but all subsequent options must be preceded with a dash. Other possible options in the option string are as follows:

- `-A`
  Restores ACL information when used with `-x` option.

- `-b`
  Sets the number of 512-byte blocks that are used for tape archive read/write operations to `blocksize`. The `blocksize` argument must be specified, and `blocksize` can be specified only when `b` is in the option string. When reading from the tape archive, `tar` automatically determines the blocking factor by trying to read the largest permitted blocking factor and using the actual number read to be the `blocksize`.

  For compatibility with z/OS UNIX, the largest valid block size is 20 blocks; in USTAR mode, it is 63 blocks.

- `-C pathname`
  Is an unusual option because it is specified in the middle of your file list. When `tar` encounters a `-C pathname` option while archiving files, it changes the working directory (for `tar` only) to `pathname` and treats all following entries in your file list (including another `-C`) as being relative to `pathname`.

- `-E`
  Although still supported for compatibility with previous versions of `tar`, this option was replaced by `-L E`.

- `-f`
  You must specify `-f`. The `-f` option uses the file `tarfile` for the tape archive rather than using the default. The `tarfile` argument must be specified, and `tarfile` can be specified only when `-f` is in the option string. The `tarfile` argument must precede the `blocksize` argument if both are present. If `tarfile` is the character `-`, then the archive format defaults to USTAR, standard input is used for reading archives, and the standard output is used for writing archives.

- `-G`
  Creates an archive in GNU tar format. For the special entry types in GNU tar format, `tar` only supports the ability to create the long pathname or long linkname entries. Other special entry types in the GNU tar, such as archiving the sparse file, are not supported. The ability to save the extended USTAR attributes is not supported.

- `-l`
  Writes an error message if all links are not resolved when files are added to the tape archive.

- `-L type`
  `-L` displays additional information when it lists the contents of an archive. Only one type can be specified per `-L` option. However, `-L` can be specified multiple times. The types that can be displayed are as follows:

  - `-A`
    Displays extended ACL (access control list) data.

    Specifying `tar` `-L A` does not automatically turn on the verbose table of contents format. You must also specify `-v` to display the chmod settings that are associated with the file.

    For more information about ACLs, see the section on Using access control lists (ACLs) in z/OS UNIX System Services Planning and “tar support for access control lists (ACLs)” on page 668.

  - `-E`
    Same as verbose (`-v`) output, but additionally displays extended attributes. See “Output” on page 667 for more information. `-L E` is equivalent to the `tar` `-E`
Displays file tag information. Does not automatically turn on the verbose -v option but can be used with -v or any other combination of table of contents display options. See “Output” on page 667 for more information.

-m
When the file is extracted from an archive, its modification stamp is not restored. The default behavior is to restore the time stamp from information contained in the archive.

-o
When files are written to archives, the owner and modes of directories in the archive are not recorded. If this is specified when extracting from an existing archive, tar does not restore any owner and group information in the archive. The default is to record this information when creating a tar archive, and to restore it when extracting from the archive.

-0
For USTAR formatted archive, this option turns off the extended USTAR support. -0 is the default and user needs to use -X option to turn on extended USTAR support for USTAR archive.

The -O option can be used to turn off the tar command to extract or list the GNU tar format with long file or linkname.

For more information, see “z/OS-extended USTAR support” on page 532.

-p
When extracting, restores the three high-order file permission bits, exactly as in the archive. They indicate the set-user-ID, set-group-ID, and sticky bit. For USTAR formatted archives, p also restores, if present, extended attributes and -A restores ACLs.

Tip: If -0 is specified, it overrides -p for extended attributes. They will not be restored. tar restores the modes exactly as stored in the archive and ignores the UMASK. To use -p on UNIX systems, you must have appropriate privileges; tar restores the modes exactly as in the archive and ignores the UMASK.

-#s
-#s is not supported by z/OS UNIX. The default archive file name that is used by tar is /dev/mt/0m. This option is the least general way to override this default. For a more general method, see the -f option. The file name generated by this option has the form /dev/mt/#s. The # can be any digit between 0 and 7, inclusive, to select the tape unit. The density selector s can be l (low), m (medium), or h (high).

-S
Forces tar to use the OS390 format, which provides support to save all files attributes by default.

-X
For USTAR formatted archives, -X enables extended USTAR support and GNU tar format to store long pathname or long linkname entries. This option has no effect for non-USTAR formatted archives except the GNU tar format. tar -X functions in the following manner:

• During archive writing, -X causes tar to preserve extended USTAR information. For the GNU tar format, -X enables tar to store long pathname or long linkname.
• During archive listing, -X causes tar to display extended USTAR information and long pathname or long linkname for the GNU tar format. This is the default; -0 can be used to disable extended USTAR support and long pathname or long linkname for the GNU tar format.
• During archive reading, -X enables tar to restore extended USTAR information and long pathname or long linkname for the GNU tar format. This is the default; -0 can be used to disable extended USTAR support and long pathname or long linkname for the GNU tar format.

The environment variable _OS390_USTAR=Y also turns on the extended USTAR information

To restore certain information, the user must also have the appropriate privileges and have specified the corresponding options. For example, you must specify -p to restore extended attributes and to restore ACLs.

For more information about extended attributes, see “z/OS-extended USTAR support” on page 532.
When a new tape archive is created with the -C option, forces `tar` to use the USTAR format. The default format that is used when a new archive is created is the original UNIX `tar` format. When you do not specify -C, `tar` can deduce whether the tape archive is in USTAR format by reading it, so you can use U to suppress a warning about USTAR format.

In order to save external links, extended attributes, and file tag information, and ACLs, the extended USTAR format must be used. To turn on the extended USTAR format, the -U and -X options must be specified. The OS390 format can also be used (-S option) to save all the file attributes by default.

**-v**
Displays each file name, along with the appropriate action key letter as it processes the archive. With the -t form of the command, this option gives more detail about each archive member being listed and shows information about the members in the same format used by the `ls -l` command. You can also use the -L type option which provides the ability to display additional information such as extended attributes and file tag information. See “Output” on page 667 for more information.

**-V volpat**
Provides automatic multivolume support. `tar` writes output to files, the names of which are formatted with `volpat`. Any occurrence of # in `volpat` is replaced by the current volume number. When you invoke `tar` with this option, it prompts for the first number in the archive set, and waits for you to type the number and a carriage return before proceeding with the operation. `tar` issues the same sort of message when a write error or read error occurs on the archive; this kind of error means that `tar` has reached the end of the volume and should go on to a new one.

**-w**
Is used to confirm each operation, such as replacing or extracting. `tar` displays the operation and the file involved. You can then confirm whether you want the operation to take place. Typing in an answer that begins with “y” tells `tar` to do the operation; anything else tells `tar` to go on to the next operation.

**-z**
Reads or writes, or both reads and writes, the tape archive by first passing through a compression algorithm compatible with that of `compress`.

**Restriction:** You cannot specify both the -r option and the -z option at the same time.

**Output**
When the -v or -L E (or -E) option is used with -t (table of contents), `tar` produces a verbose table of contents for the archive. The -L T option can also be used to additionally, or without the verbose output, display file tag information. The output for -v is similar to the output from the `ls -l` command with following exceptions:

- The following notation is used to represent hard, symbolic, and external links:
  - hlink external link to origfile
  - slink symbolic link to origfile
  - elink external link to ORIG.FILE

  indicates that hlink is a hard link to origfile.
  indicates that slink is a symbolic link to origfile.
  indicates that elink is an external link to ORIG.FILE.
- For symbolic and external links, `pax` output always shows a file size of 0.

Refer to the description of `ls` for an explanation of the `ls -v`. 

Shell command descriptions 667
The output from the `-L E` (or `-E`) option has the same format as `-v` and additionally displays a column showing the extended attributes:

- **a**
  - Program runs APF-authorized if linked AC=1.

- **p**
  - Program is considered program-controlled.

- **s**
  - Program runs in a shared address space.

- **l**
  - Program is loaded from the shared library region.

  **Note:** `l` is a lowercase `L`, not an uppercase `i`.

- **-**
  - Attribute not set.

The format of the `tar -L E` (or `-E`) output is variable in length and will be extended as necessary to display additional file characteristics that are not supported by `tar -v` (ls -l).

The format of the `tar -L T` output is similar to the output from `chtag -p`. If specified with `-v` or `-L E`, the output is displayed on the same line of and before the `-v` output. When used without `-v`, only the file tag information and file names are displayed. For example:

```
/tmp> tar -L T -tf asciitagged.tar
m ISO8859-1 T=off text_am
t ISO8859-1 T=on  text_at
- untagged T=off text_au
```

This option can be used with either `-v` or `-o E` (or both) to display additional verbose output. For example:

```
/tmp> tar -L T -tvf asciitagged.tar
m ISO8859-1 T=off -rw-r--r--  1 SteveS  Kings   9 Apr 30 22:31 text_am
mt ISO8859-1 T=on  -rw r--r--  1 SteveS  Kings   9 Apr 30 22:31 text_at
- untagged T=off -rw-r--r--  1 SteveS  Kings   9 Apr 30 22:06 text_au
```

### tar support for access control lists (ACLs)

**For archive writing or creating:** ACL data is stored in USTAR formatted archives, when `-X` option is used. The OS390 format (`-S` option) also stores the ACL information.

`tar -O` can be used to disable the creation of special headers. This prevents `tar` from storing ACL data and other nonstandard information such as file tag data and long link names. However, there is no option to disable storing of ACL data only.

**For archive reading or restoring:** By default, ACL data will not be restored when reading or restoring files from an archive. However, for USTAR and OS390 formatted archives, you can use `tar -A` to restore ACL data.

**For archive listing (table of contents):** For verbose output (tar), `+` is added to the end of the file permission bits for all files with extended ACLs. For example, `file2` and `dir1` have extended ACL entries:

```
> tar -tvf acldata.tar
  -rwx------  1 STIERT SHUT  294912 Nov  9 09:57 file1
  -rwx------  1 STIERT SHUT  294912 Nov  9 09:57 file2
  drwxr-xr-x+ 2 STIERT SHUT  8192 Mar 20 2000 dir1/
```

For more information about **Using access control lists (ACLs)**, see [z/OS UNIX System Services Planning](http://www.ibm.com/systems/z/os/ibm/Docs/prod/zos.doc/guide/).

### Usage notes

1. Use the `pax` command if you need to use multibyte patterns when searching for file names.
2. The POSIX 1003.1 standard defines formats for **pax** and **tar** archives that limit the length of the target of a link file to 100 characters or less.

**Note:** In the case of a hard link, the target is the first occurrence of the hard link which is archived. Subsequent hard links refer to the first instance.

**pax** and **tar** provide extended USTAR support and the OS390 format that allows these links to be preserved when creating an archive and restored when reading an archive. See “**z/OS-extended USTAR support**” on page 532 for more information.

3. The POSIX 1003.1 standard defines formats for **pax** and **tar** archives that limit the size of a file that can be stored in a pax and **tar** archive to less than 8 gigabytes in size. If a file being archived is 8 gigabytes or greater, an error message is issued, and the file is skipped. The command continues, but will end with a nonzero exit status.

4. On the z/OS system, superuser privileges or read access to the appropriate FACILITY classes are required to create character special files, to restore user and group names, and to set certain extended attributes.

5. Path names in the tape archive are normally restricted to a maximum length of 100 bytes. However, in USTAR (-U) and OS390(-O) format, path names can be up to 255 bytes long.

6. When transferring archives between z/OS systems and other UNIX systems, note the following:
   a. File transfers (for example, using OPUT/OGET or ftp put/get) must be done using binary or image format. This is true, even for archives consisting only of text files.
   b. You might need to convert text files from EBCDIC to ASCII (or some other character set). You can use the `iconv` utility to convert files before or after archiving. When text files are being created or extracted, you can use the `pax -o` option to convert them.

7. Automatic conversion on files with file tag information is disabled when reading files while archives are being created, and during writes while files are being extracted from archives. That is, the settings of system and environment variables that turn automatic conversion on and off do not affect the reading and writing of files by **tar**. **pax** supports file tag options that support conversion of files based on their file tag settings.

8. The POSIX 1003.1 standard defines formats for **tar** and USTAR archives that limit the modification time to a maximum value of 77777777777 (octal). Modification times that are larger than the maximum value are not properly saved. Modification times beyond 03:14:07 UTC on January 19, 2038 but less than or equal to the maximum value, are not properly saved or restored on z/OS V2R2 and earlier systems. They also might not be properly saved or restored on non-z/OS systems. However, on systems later than z/OS V2R2, such values are properly saved and restored. Due to these restrictions, the **pax** utility with the **pax** interchange format is the preferred method for processing time values beyond 03:14:07 UTC on January 19, 2038.

**Note:** The OS390 format has the same maximum value as the USTAR format.

### Examples

1. The following command takes a directory and places it in an archive in compressed format:

   ```bash
tar -cvzf archive directory
   ```

2. To identify all files that were changed in the last week (7 days), and to archive them to the `/tmp/posix/testpgm` file, enter:

   ```bash
   find /tmp/posix/testpgm -type f -mtime -7 | tar -cvf testpgm.tar -
   
type -f tells `find` to select only files. This avoids duplicate input to `tar`.

3. In the following examples, archive `acldata.tar` contains `file1`, `file2`, and `dir1`. `file1` has no ACL data, `file2` has an access ACL, and `dir1` contains a file default, a directory default, and an access ACL. If you only specify option `-f`, your output will be:

   ```bash
   > tar -f acldata.tar
   file1
   ```
If you also specify -L A, ACL information will be displayed:

```bash
> tar -L A -f acldata.tar
file1
file2
user:WELLIE2:rw-
group:SYS1:rwx
```

Finally, if you add the verbose option, -v, you will see the chmod settings associated with the file:

```bash
> tar -L A -vf acldata.tar
-rwx------   1 STIERT  SHUT  294912 Nov  9 09:57 file1
-rwx------+  1 STIERT  SHUT  294912 Nov  9 09:57 file2
user:WELLIE2:rw-
group:SYS1:rwx
drwxr-xr-x+  2 STIERT  SHUT  8192  Mar 20  2000 dir1/
user:RRAND:rwx
user:WELLIE2:rw-
group:SHUT:rwx
fdefault:user:RRAND:rwx
fdefault:group:SHUT:r-x
default:user:ANGIEH:rwx
default:group:SYS1:r--
```

**Localization**

tar uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Exit values**

0  Successful completion.

1  Failure due to any of the following:
- Incorrect option.
- Incorrect command-line arguments.
- Out of memory.
- Compression error.
- Failure on extraction.
- Failure on creation.

**Portability**

4.2BSD

The -U option is an extension to provide POSIX USTAR format compatibility. The -p option is a common extension on BSD UNIX systems that is not available on UNIX System V systems. The -O, -X, and -S options are also extensions of POSIX standard.

**Related information**

cpio, pax
Also, see the pax file format description in Appendix H, “File formats,” on page 969 for more information.

**tcsh: Invoke a C shell**

**Format**

\texttt{tcsh [-bcdeFimnqstvVxX]}
\texttt{tcsh -l}

**Note:** -l is a lowercase L, not an uppercase i.

**Description**

The tcsh shell is an enhanced but completely compatible version of the Berkeley UNIX C shell, tcsh. It is a command language interpreter usable both as an interactive login shell and a shell script command processor. It includes a command-line editor, programmable word completion, spelling correction, a history mechanism, job control, and a C-like syntax.

You can invoke the shell by typing an explicit \texttt{tcsh} command. A login shell can also be specified by invoking the shell with the -l option as the only argument.

A login shell begins by executing commands from the system files \texttt{/etc/csh.cshrc} and \texttt{/etc/csh.login}. It then executes commands from files in the user's home directory: first \texttt{~/.tcshrc}, then \texttt{~/.history} (or the value of the \texttt{histfile} shell variable), then \texttt{~/.login}, and finally \texttt{~/.cshdirs} (or the value of the \texttt{dirsfile} shell variable). The shell reads \texttt{/etc/csh.login} after \texttt{/etc/csh.cshrc}.

Non-login shells read only \texttt{/etc/csh.cshrc} and \texttt{~/.tcshrc} or \texttt{~/.cshrc} on invocation.

Commands like \texttt{stty}, which need be run only once per login, typically go in the user's \texttt{~/.login} file.

In the normal case, the shell begins reading commands from the terminal, prompting with >. The shell repeatedly reads a line of command input, breaks it into words, places it on the command history list, and then parses and executes each command in the line. See “Command execution” on page 687.

A user can log out of a tcsh shell session by typing ^D, \texttt{logout}, or \texttt{login} on an empty line (see \texttt{ignoreeof} shell variable), or via the shell's autologout mechanism. When a login shell terminates, it sets the \texttt{logout} shell variable to normal or automatic as appropriate, then executes commands from the files \texttt{/etc/csh.logout} and \texttt{~/.logout}.

The names of the system login and logout files vary from system to system for compatibility with different csh variants; see “tcsh files” on page 713.

**Restriction:** If the tagged script is being run with automatic conversion enabled, the code page of the locale must be SBCS.

**Options**

If the first argument (argument 0) to the tcsh shell is - (hyphen), then it is a login shell. You can also specify the login shell by invoking the tcsh shell with the -l as the only argument.

The z/OS UNIX System Services tcsh shell accepts the following options on the command line:

- **-b**
  Forces a break from option processing, causing any further shell arguments to be treated as non-option arguments. The remaining arguments will not be interpreted as shell options. This can be used to pass options to a shell script without confusion or possible subterfuge.

- **-c**
  Reads and executes commands stored in the command shell (this option must be present and must be a single argument). Any remaining arguments are placed in the \texttt{argv} shell variable.
-d
Loads the directory stack from ~/.cshdirs whether or not it is a login shell.

-e
Terminates shell if any invoked command terminates abnormally or yields a nonzero exit status.

-i
Invokes an interactive shell and prompts for its top-level input, even if it appears to not be a terminal. Shells are interactive without this option if their inputs and outputs are terminals.

-l
Invokes a login shell. Only applicable if -l is the only option specified.

Note: -l is a lowercase L not an uppercase i.

-m
Loads ~/.tcshrc even if it does not belong to the effective user.

-n
Parses commands but does not execute them. This aids in debugging shell scripts.

-q
Accepts SIGQUIT and behaves when it is used under a debugger. Job control is disabled.

-s
Take command input from the standard input.

-t
Reads and executes a single line of input. A \ (backslash) can be used to escape the newline at the end of this line and continue onto another line.

-v
Sets the verbose shell variable so command input is echoed after history substitution.

-V
Sets the verbose shell variable even before executing ~/.tcshrc.

-x
Sets the echo shell variable so commands are echoed immediately before execution.

-X
Is to -x as -V is to -v.

After processing of option arguments, if arguments remain but none of the -c, -i, -s, or -t were given, the first argument is taken as the name of a file of commands, or script, to be executed. The shell opens this file and saves its name for possible resubstitution by $0. Since many systems use shells whose shell scripts are not compatible with this shell, the tcsh shell uses such a standard shell to execute a script whose character is not a #, that is, which does not start with a comment.

Remaining arguments are placed in the argv shell variable.

tcsh shell editing

The command-line editor is described. We then discuss Completion and Listing and Spelling Correction. They are two sets of functionality that are implemented as editor commands. Finally, the Editor Commands topic lists and describes the editor commands specific to the tcsh shell and their default bindings.

Command-line editor

Command-line input can be edited by using key sequences much like those used in GNU Emacs or vi. The editor is active only when the edit shell variable is set, which it is by default in interactive shells. The bindkey built-in command can display and change key bindings. Emacs-style key bindings are used by default, but bindkey can change the key bindings to vi-style bindings.
The shell always binds the arrow keys to:

down
   down-history
up
   up-history
left
   backward-char
right
   forward-char

unless doing so would alter another single-character binding. To prevent these bindings, set the arrow key escape sequences to the empty string with `settc`.

Other key bindings are, for the most part, what `emacs` and `vi` users would expect and can easily be displayed by `bindkey`, so there is no need to list them here. Likewise, `bindkey` can list the editor commands with a short description of each.

**Note:** Editor commands do not have the same notion of a word as does the tcsh shell. The editor delimits words with any nonalphanumeric characters not in the shell variable `wordchars`. The tcsh shell recognizes only white space and some of the characters with special meanings to it, listed in “Command syntax” on page 679.

### Completion and listing

The tcsh shell is often able to complete words when given a unique abbreviation. Type part of a word (for example `ls /usr/lost`) and press the tab key to run the `complete-word` editor command. The shell completes the file name `/usr/lost` to `/usr/lost+found`, replacing the incomplete word with the complete word in the input buffer. (To speed up typing and to provide a visual indicator of successful completion, the terminal `/` (forward slash); completion adds a `/` to the end of completed directories and a space to the end of other completed words. The `addsuffix` shell variable can be unset to prevent this action.) If no match is found (for example, `/usr/lost+found` does not exist), the terminal bell rings. If the word is already complete (for example, there is a `/usr/lost` on your system, or you were thinking too far ahead and typed the whole thing), a `/` or space is added to the end if it is not already there.

Completion works anywhere in the line, not just at the end; completed text pushes the rest of the line to the right. Completion in the middle of a word often results in leftover characters to the right of the cursor. Those leftover characters need to be deleted.

Commands and variables can be completed in much the same way. For example, typing `em [tab]` would complete `em` to `emacs` if `emacs` were the only command on your system beginning with `em`. Completion can find a command in any directory in the path or if given a full path name. Typing `echo $ar[tab]` would complete `$ar` to `$argv` if no other variable began with `ar`.

The shell parses the input buffer to determine whether the word you want to complete should be completed as a file name, command or variable. The first word in the buffer and the first word following `;`, `|`, `|&`, `&|` or `||` is considered to be a command. A word beginning with `$` is considered to be a variable. Anything else is a file name. An empty line is completed as a file name.

You can list the possible completions of a word at any time by typing `^D` to run the `delete-char-or-list-or-eof` editor command. The tcsh shell lists the possible completions using the `ls-F` built-in and reprints the prompt and unfinished command line, for example:

```
> ls /usr/l[^D]
  lib/  lib/ local/ lost+found/
> ls /usr/l
```

Shell command descriptions 673
If the `autolist` shell variable is set, the tcsh shell lists the remaining choices (if any) whenever completion fails:

```
> set autolist
> nm /usr/lib/libt[tab]
libtermcap.a@ libtermlib.a@
> nm /usr/lib/libterm
```

If `autolist` is set to `ambiguous`, choices are listed only if multiple matches are possible, and if the completion adds no new characters to the name to be matched.

A file name to be completed can contain variables, your own or others' home directories that are abbreviated with ~ (tilde; see File name substitution) and directory stack entries that are abbreviated with = (equal; see “Directory stack substitution” on page 686). For example:

```
> ls ~k[^D]  
kahn kas kellogg
> ls ~ke[tab]
> ls ~kellogg/
```

```
or
> set local = /usr/local
> ls $lo[tab]
> ls $local/~[D]
bin/ etc/ lib/ man/ src/
> ls $local/
```

Variables can also be expanded explicitly with the expand-variables editor command.

delete-char-or-list-or-eof only lists at the end of the line; in the middle of a line it deletes the character under the cursor and on an empty line it logs one out or, if ignoreeof is set, does nothing. M-^D, bound to the editor command list-choices, lists completion possibilities anywhere on a line, and list-choices (or any one of the related editor commands which do or don't delete, list and log out, listed under delete-char-or-list-or-eof) can be bound to ^D with the bindkey built-in command if so desired.

The complete-word-fwd and complete-word-back editor commands (not bound to any keys by default) can be used to cycle up and down through the list of possible completions, replacing the current word with the next or previous word in the list.

The tcsh shell variable `fignore` can be set to a list of suffixes to be ignored by completion. Consider the following:

```
> ls
Makefile condiments.h~ main.o side.c
README main.c meal side.o
condiments.h main.c-
> set fignore = (.o ~)
> emacs ma[^D]
> emacs ma[tab]
> emacs main.c
> emacs main.c~
```

`main.c~` and `main.o` are ignored by completion (but not listing), because they end in suffixes in `fignore`. \ is needed in front of ~ to prevent it from being expanded to home as described in File name substitution. `fignore` is ignored if only one completion is possible.

If the complete shell variable is set to `enhance`, completion: 1.) ignores case and 2.) considers periods, hyphens and underscores (\, `-` and `_`) to be word separators and hyphens and underscores to be equivalent.

If you had the following files:

```
comp.lang.c comp.lang.perl comp.std.cpp
comp.lang.c++ comp.std.c
```
and typed `mail -f c.l.c[tab]`, it would be completed to `mail -f comp.lang.c`, and `^[D` would list comp.lang.c and comp.lang.c++. `mail -f c.c++[^[D]` would list comp.lang.c++ and comp.std.c++. Typing `rm a--file[^D]` in the following directory

```
A_silly_file a-hyphenated-file another_silly_file
```

would list all three files, because case is ignored and hyphens and underscores are equivalent. Periods, however, are not equivalent to hyphens or underscores.

Completion and listing are affected by several other tcsh shell variables: `recexact` can be set to complete on the shortest possible unique match, even if more typing might result in a longer match. For example:

```
> ls
fodder foo food foonly
> set recexact
> rm fo[tab]
```

just beeps, because 'fo' could expand to 'fod' or 'foo', but if we type another 'o',

```
> rm foo[tab]
> rm foo
```

the completion completes on 'foo', even though 'food' and 'foonly' also match. `autoexpand` can be set to run the expand-history editor command before each completion attempt, and `correct` can be set to complete commands automatically after one hits 'return'. `matchbeep` can be set to make completion beep or not beep in a variety of situations, and `nobeep` can be set to never beep at all. `nostat` can be set to a list of directories and patterns which match directories to prevent the completion mechanism from stat(2)ing those directories.

**Note:** The completion operation succeeds, but faster. The setting of `nostat` is evident when using the `listflags` variable. For example:

```
>set listflags=x>
ls-F /u/pluto
Dirl/exe1*
>set nostat=(/u/pluto/)
>ls-F /u/pluto
Dirlexe1
```

Although, you must be careful when setting `nostat` to keep the trailing / (forward slash).

`listmax` and `listmaxrows` can be set to limit the number of items and rows (respectively) that are listed without asking first. `recognize_only_executables` can be set to make the shell list only executables when listing commands, but it is quite slow.

Finally, the complete built-in command can be used to tell the shell how to complete words other than file names, commands and variables. Completion and listing do not work on glob-patterns (see File name substitution), but the `list-glob` and `expand-glob` editor commands perform equivalent functions for glob-patterns.

**Spelling correction**

The tcsh shell can sometimes correct the spelling of file names, commands and variable names as well as completing and listing them.

Individual words can be corrected for spelling with the `spell-word` editor command (typically bound to M-s and M-S where M = Meta Key or escape (ESC) key) and the entire input buffer with `spell-line` (typically bound to M-$). The correct shell variable can be set to 'cmd' to correct the command name or 'all' to correct the entire line each time return is typed.
When spelling correction is invoked in any of these ways and the shell thinks that any part of the command line is misspelled, it prompts with the corrected line:

```
> set correct = cmd
> lz /usr/bin
CORRECT>ls /usr/bin (y|n|e|a)?
```

where one can answer y or space to execute the corrected line, e to leave the uncorrected command in the input buffer, a to abort the command as if ^C had been pressed, and anything else to execute the original line unchanged.

Spelling correction recognizes user-defined completions (see the complete built-in command). If an input word in a position for which a completion is defined resembles a word in the completion list, spelling correction registers a misspelling and suggests the latter word as a correction. However, if the input word does not match any of the possible completions for that position, spelling correction does not register a misspelling.

Like completion, spelling correction works anywhere in the line, pushing the rest of the line to the right and possibly leaving extra characters to the right of the cursor.

Spelling correction is not guaranteed to work the way one intends, and is provided mostly as an experimental feature.

**Editor commands**

bindkey lists key bindings and bindkey -l lists and briefly describes editor commands. Only new or especially interesting editor commands are described here. See emacs and vi for descriptions of each editor's key bindings.

The character or characters to which each command is bound by default is given in parentheses. ^character means a control character and M-character a meta character, typed as escape-character on terminals without a meta key. Case counts, but commands that are bound to letters by default are bound to both lowercase and uppercase letters for convenience.

**complete-word**
Completes a word as described in Completion and listing.

**complete-word-back**
Like complete-word-fwd, but steps up from the end of the list.

**complete-word-fwd**
Replaces the current word with the first word in the list of possible completions. can be repeated to step down through the list. At the end of the list, beeps and reverts to the incomplete word.

**complete-word-raw**
Like complete-word, but ignores user-defined completions.

**copy-prev-word**
Copies the previous word in the current line into the input buffer. See also insert-last-word.

**dabbrev-expand**
Expands the current word to the most recent preceding one for which the current is a leading substring, wrapping around the history list (once) if necessary. Repeating dabbrev-expand without any intervening typing changes to the next previous word etc., skipping identical matches much like history-search-backward does.

**delete-char (not bound)**
Deletes the character under the cursor. See also delete-char-or-list-or-eof.

**delete-char-or-eof (not bound)**
Does delete-char if there is a character under the cursor or end-of-file on an empty file. See also delete-char-or-list-or-eof.
delete-char-or-list (not bound)
Does delete-char if there is a character under the cursor or list-choices at the end of the line. See also delete-char-or-list-or-eof.

delete-char-or-list-or-eof (^D)
Does delete-char if there is a character under the cursor, list-choices at the end of the line or end-of-file on an empty line. See also delete-char-or-eof, delete-char-or-list and list-or-eof.

down-history
Like up-history, but steps down, stopping at the original input line.

down-of-file
Signals an end of file, causing the tcsh shell to exit unless the ignoreeof shell variable is set to prevent this. See also delete-char-or-list-or-eof.

expand-history (M-space)
Expands history substitutions in the current word. See History substitution. See also magic-space, toggle-literal-history, and the autoexpand shell variable.

expand-glob(^X-*)
Expands the glob-pattern to the left of the cursor. For example:

```
>ls test*[^X-]*
```

would expand to

```
>ls test1.c test2.c
```

if those were the only two files in your directory that begin with 'test'. See File name substitution.

expand-line (not bound)
Like expand-history, but expands history substitutions in each word in the input buffer.

expand-variables (^X-$)
Expands the variable to the left of the cursor. See Variable substitution.

history-search-backward (M-p, M-P)
Searches backwards through the history list for a command beginning with the current contents of the input buffer up to the cursor and copies it into the input buffer. The search string can be a glob-pattern (see File name substitution) containing '*', '?', '[' or ']'. up-history and down-history will proceed from the appropriate point in the history list. Emacs mode only. See also history-search-forward and i-search-back.

history-search-forward(M-n, M-N)
Like history-search-backward, but searches forward.

i-search-back (not bound)
Searches backward like history-search-backward, copies the first match into the input buffer with the cursor positioned at the end of the pattern, and prompts with 'bck: ' and the first match. Additional characters can be typed to extend the search. i-search-back can be typed to continue searching with the same pattern, wrapping around the history list if necessary, (i-search-back must be bound to a single character for this to work) or one of the following special characters can be typed:

^W
Appends the rest of the word under the cursor to the search pattern.

delete (or any character bound to backward-delete-char)
Undoes the effect of the last character and deletes a character from the search pattern if appropriate.

^G
If the previous search was successful, aborts the entire search. If not, goes back to the last successful search.

escape
Ends the search, leaving the current line in the input buffer.
Any other character not bound to **self-insert-command** terminates the search, leaving the current line in the input buffer, and is then interpreted as normal input. In particular, a carriage return causes the current line to be executed. Emacs mode only. See also **i-search-fwd** and **history-search-backward**.

**i-search-fwd**
Like i-search-back, but searches forward.

**insert-last-word (M-_)**
Inserts the last word of the previous line (!$) into the input buffer. See also **copy-prev-word**.

**list-choices (M-D)**
Lists completion possibilities as described under **Completion and listing**. See also **delete-char-or-list-or-eof**.

**list-choices-raw (^X-^D)**
Like **list-choices**, but ignores user-defined completions.

**list-glob (^X-g, ^X-G)**
Lists (via the 1s -F) matches to the glob-pattern (see **File name substitution**) to the left of the cursor.

**list-or-eof (not bound)**
Does **list-choices** or **end-of-file** on an empty line. See also **delete-char-or-list-or-eof**.

**magic-space (not bound)**
Expands history substitutions in the current line, like **expand-history**, and appends a space. **magic-space** is designed to be bound to the spacebar, but is not bound by default.

**normalize-command (^X-?)**
Searches for the current word in PATH and, if it is found, replaces it with the full path to the executable. Special characters are quoted. Aliases are expanded and quoted but commands within aliases are not. This command is useful with commands which take commands as arguments, for example, dbx and sh -x.

**normalize-path (^X-n, ^X-N)**
Expands the current word as described under the expand setting of the **symlinks** shell variable.

**overwrite-mode (unbound)**
Toggles between input and overwrite modes.

**run-fg-editor (M-^Z)**
Saves the current input line and looks for a stopped job with a name equal to the last component of the file name part of the EDITOR or VISUAL environment variables, or, if neither is set, ed or vi. If such a job is found, it is restarted as if fg % job had been typed. This is used to toggle back and forth between an editor and the shell easily. Some people bind this command to ^Z so they can do this even more easily.

**run-help (M-h, M-H)**
Searches for documentation on the current command, using the same notion of **current command** as the completion routines, and prints it. There is no way to use a pager; **run-help** is designed for short help files. Documentation should be in a file named **command**.help, **command**.1, **command**.6, **command**.8 or **command**, which should be in one of the directories listed in the HPATH environment variable. If there is more than one help file only the first is printed.

**self-insert-command (text characters)**
In insert mode (the default), inserts the typed character into the input line after the character under the cursor. In overwrite mode, replaces the character under the cursor with the typed character. The input mode is normally preserved between lines, but the **inputmode** shell variable can be set to insert or overwrite to put the editor in that mode at the beginning of each line. See also **overwrite-mode**.

**sequence-lead-in (arrow prefix, meta prefix, ^X)**
Indicates that the following characters are part of a multi-key sequence. Binding a command to a multi-key sequence really creates two bindings: the first character to **sequence-lead-in** and the whole sequence to the command. All sequences beginning with a character bound to **sequence-lead-in** are effectively bound to **undefined-key** unless bound to another command.
spell-line (M-$)
Attempts to correct the spelling of each word in the input buffer, like spell-word, but ignores words whose first character is one of ' - ', ' ! ', ' ^ ' or ' % ', or which contain '\', '* ' or '? ', to avoid problems with switches, substitutions and the like. See Spelling correction.

spell-word (M-s, M-S)
Attempts to correct the spelling of the current word as described under Spelling correction. Checks each component of a word which appears to be a path name.

toggle-literal-history (M-r, M-R)
Expands or unexpands history substitutions in the input buffer. See also expand-history and the autoexpand shell variable.

undefined-key (any unbound key)
Beeps.

up-history (up-arrow, ^P)
Copies the previous entry in the history list into the input buffer. If histlit is set, uses the literal form of the entry. Can be repeated to step up through the history list, stopping at the top.

vi-search-back (?)
Prompts with ? for a search string (which can be a glob-pattern, as with history-search-backward), searches for it and copies it into the input buffer. The bell rings if no match is found. Hitting return ends the search and leaves the last match in the input buffer. Hitting escape ends the search and executes the match. vi mode only.

vi-search-fwd (/)
Like vi-search-back, but searches forward.

which-command (M-?)
Does a which (built-in command) on the first word of the input buffer. which displays the command that is executed by the shell after substitutions and path searching. The displayed command has passed access checks by the security product based on the effective ids of the user.

Command syntax

The tcsh shell splits input lines into words at blanks and tabs. The special characters '&', '|', ';', '<', '>', '(' and ')' and the doubled characters '&&', '||', '<<' and '>>' are always separate words, whether or not they are surrounded by white space.

When the tcsh shell's input is not a terminal, the character '#' is taken to begin a comment. Each # and the rest of the input line on which it appears is discarded before further parsing.

A special character (including a blank or tab) can be prevented from having its special meaning, and possibly made part of another word, by preceding it with a backslash (\) or enclosing it in single ('), double ("), or backward (`) quotation marks. When not otherwise quoted a newline preceded by a \ is equivalent to a blank, but inside quotes this sequence results in a newline.

Furthermore, all substitutions (see “Substitutions” on page 680) except history substitution can be prevented by enclosing the strings (or parts of strings) in which they appear with single quotation marks or by quoting the crucial characters (for example, '$' or ' ' for variable substitution or command substitution respectively) with \, (alias substitution is no exception: quoting in any way any character of a word for which an alias has been defined prevents substitution of the alias. The usual way of quoting an alias is to precede it with a backslash.) History substitution is prevented by backslashes but not by single quotation marks. Strings quoted with double or backward quotation marks undergo Variable substitution and Command substitution, but other substitutions are prevented.

Text inside single or double quotation marks becomes a single word (or part of one). Metacharacters in these strings, including blanks and tabs, do not form separate words. Only in one special case (see Command substitution) can a double-quoted string yield parts of more than one word; single-quoted strings never do. Backward quotes are special: they signal command substitution, which might result in more than one word.
Quoting complex strings, particularly strings which themselves contain quoting characters, can be confusing. Remember that quotes need not be used as they are in human writing. It might be easier to quote not an entire string, but only those parts of the string which need quoting, using different types of quoting to do so if appropriate.

The `backslash_quote` shell variable can be set to make backslashes always quote `\`, `,`, and `"`. This action might make complex quoting tasks easier but can cause syntax errors in csh (or tcsh) scripts.

**Substitutions**

The tcsh shell performs various transformations on input in the order in which they occur. The topic covers data structures and the commands and variables that affect them. Remember that substitutions can be prevented by quoting as described in “Command syntax” on page 679.

**History substitution**

Each command, or event, input from the terminal is saved in the history list. The previous command is always saved, and the `history` shell variable can be set to a number to save that many commands. The `histdup` shell variable can be set to not save duplicate events or consecutive duplicate events.

Saved commands are numbered sequentially from 1 and stamped with the time. It is not typically necessary to use event numbers, but the current event number can be made part of the prompt by placing an exclamation point (!) in the `prompt` shell variable.

The shell actually saves history in expanded and literal (unexpanded) forms. If the `histlit` shell variable is set, commands that display and store history use the literal form.

The `history` built-in command can print, store in a file, restore and clear the history list at any time, and the `savehist` and `histfile` shell variables can be set to store the history list automatically on logout and restore it on login.

History substitutions introduce words from the history list into the input stream, making it easy to repeat commands, repeat arguments of a previous command in the current command, or fix spelling mistakes in the previous command with little typing and a high degree of confidence.

History substitutions begin with the character `!`. They can begin anywhere in the input stream, but they do not nest. The `!` can be preceded by a `\` to prevent its special meaning; for convenience, a `!` is passed unchanged when it is followed by a blank, tab, newline, = or (. History substitutions also occur when an input line begins with `^`. The characters used to signal history substitution (`!` and `^` (caret)) can be changed by setting the `histchars` shell variable. Any input line which contains a history substitution is printed before it is executed.

A history substitution can have an **event specification**, which indicates the event from which words are to be taken, a **word designator**, which selects particular words from the chosen event, and a **modifier**, which manipulates the selected words.

An event specification can be

- `n`  
  A number, referring to a particular event.

- `-n`  
  An offset, referring to the even n before the current event.

- `#`  
  The current event. This should be used carefully in csh, where there is no check for recursion. tcsh allows 10 levels of recursion.

- `!`  
  The previous event, which is equivalent to -1.
s
The most recent event whose first word begins with the string s.

?s?
The most recent event which contains the string s. The second ? can be omitted if it is immediately followed by a newline.

For example, consider this bit of someone's history list:

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>8:30</td>
<td>nroff -man wumpus.man</td>
</tr>
<tr>
<td>10</td>
<td>8:31</td>
<td>cp wumpus.man wumpus.man old</td>
</tr>
<tr>
<td>11</td>
<td>8:36</td>
<td>vi wumpus.man</td>
</tr>
<tr>
<td>12</td>
<td>8:37</td>
<td>diff wumpus.man.old wumpus.man</td>
</tr>
</tbody>
</table>

The commands are shown with their event numbers and time stamps. The current event, which we have not typed in yet, is event 13. !11 and !-2 refer to event 11. !! refers to the previous event, 12. !! can be abbreviated ! if it is followed by a : (colon). !n refers to event 9, which begins with n. !?old? also refers to event 12, which contains old. Without word designators or modifiers history references simply expand to the entire event, so we might type !cp to redo the copy command or !!|more if the diff output scrolled off the top of the screen.

History references can be insulated from the surrounding text with braces if necessary. For example, !vdoc would look for a command beginning with vdoc, and, in this example, not find one, but !(v)doc would expand unambiguously to vi wumpus.mandoc. Even in braces, history substitutions do not nest.

While csh expands, for example, !3d to event 3 with the letter d appended to it, tcsh expands it to the last event beginning with 3d; only completely numeric arguments are treated as event numbers. This makes it possible to recall events beginning with numbers. To expand !3d as in csh say !\3d.

To select words from an event we can follow the event specification by a : (colon) and a designator for the desired words. The words of an input line are numbered from 0, the first (typically command) word being 0, the second word (first argument) being 1, etc. The basic word designators are:

0
The first command word

n
The nth argument

^ The first argument, equivalent to 1

$ The last argument

% The word matched by an ?s? search

x-y A range of words

-y Equivalent to 0-y

* Equivalent to ^-$, but returns nothing if the event contains only 1 word

x* Equivalent to x-$

x- Equivalent to x*, but omitting the last word ($)
current event. In:- hurkle.man would reuse the first two words from the nroff command to say nroff -man hurkle.man.

The : separating the event specification from the word designator can be omitted if the argument selector begins with a '^', '$', '*', '%' or '-'. For example, our diff command might have been diff !!^old !!^ or, equivalently, diff !$.old !$. However, if !! is abbreviated !, an argument selector beginning with - (hyphen) will be interpreted as an event specification.

A history reference can have a word designator but no event specification. It then references the previous command. Continuing our diff example, we could have said simply diff !^old !^ or, to get the arguments in the opposite order, just diff !*.

The word or words in a history reference can be edited, or modified, by following it with one or more modifiers, each preceded by a : (colon):

- **h** Remove a trailing path name component, leaving the head.
- **t** Remove all leading path name components, leaving the tail.
- **r** Remove a file name extension .xxx, leaving the root name.
- **e** Remove all but the extension
- **u** Uppercase the first lowercase letter.
- **l** Lowercase the first uppercase letter.
- **s/l/r** Substitute l for r. l is simply a string like r, not a regular expression as in the eponymous ed command. Any character can be used as the delimiter in place of /. a \ can be used to quote the delimiter inside l and r. The character & in the r is replaced by l; \ also quotes &. If l is empty (""), the l from a previous substitution or the s from a previous ?s? event specification is used. The trailing delimiter can be omitted if it is immediately followed by a newline.
- **&** Repeat the previous substitution
- **g** Apply the following modifier once to each word.
- **a** Apply the following modifier as many times as possible to a single word. 'a' and 'g' can be used together to apply a modifier globally. In the current implementation, using the 'a' and 's' modifiers together can lead to an infinite loop. For example, :as/l/ff/ will never terminate. This behavior might change in the future.
- **p** Print the new command line but do not execute it.
- **q** Quote the substituted words, preventing further substitutions.
- **x** Like q, but break into words at blanks, tabs and newlines.

Modifiers are applied only to the first modifiable word (unless 'g' is used). It is an error for no word to be modifiable.

For example, the diff command might have been written as diff wumpus.man.old !#^:r, using :r to remove .old from the first argument on the same line (!#^). We could say echo hello out there, then echo ! *:u to capitalize 'hello', echo !*:au to say it out loud, or echo !*:agu to really shout. We might follow mail -s "I forgot my password" rot with !:s/rot/root to correct the spelling of 'root' (but see Spelling correction for a different approach).
There is a special abbreviation for substitutions. ^, when it is the first character on an input line, is equivalent to !s^s. Thus, we might have said ^rot^root to make the spelling correction in the previous example. This is the only history substitution which does not explicitly begin with !.

In csh as such, only one modifier can be applied to each history or variable expansion. In tcsh, more than one can be used, for example

| % mv wumpus.man /usr/man/man1/wumpus.1 |
| % man !$:t:r |
| man wumpus |

In csh, the result would be wumpus.1:r. A substitution followed by a colon might need to be insulated from it with braces:

| > mv a.out /usr/games/wumpus |
| > setenv PATH !$:h:$PATH |
| Bad ! modifier: $ |
| > setenv PATH !{2:$h}:$PATH |
| setenv PATH /usr/games:/bin:/usr/bin: |

The first attempt would succeed in csh but fails in tcsh, because tcsh expects another modifier after the second colon instead of $.

Finally, history can be accessed through the editor and through the substitutions just described. The following commands search for events in the history list and compile them into the input buffer:

- up-history
- down-history
- history-search-backward
- history-search-forward
- i-search-back
- i-search-fwd
- vi-search-back
- vi-search-fwd
- copy-prev-word
- insert-last-word

The toggle-literal-history editor command switches between the expanded and literal forms of history lines in the input buffer. expand-history and expand-line expand history substitutions in the current word and in the entire input buffer respectively.

**Alias substitution**

The shell maintains a list of aliases which can be set, unset, and printed by the alias and unalias commands. After a command line is parsed into simple commands (see “Command execution” on page 687) the first word of each command, left-to-right, is checked to see if it has an alias. If so, the first word is replaced by the alias. If the alias contains a history reference, it undergoes history substitution as though the original command were the previous input line. If the alias does not contain a history reference, the argument list is left untouched.

Thus if the alias for *ls* were *ls* -1 the command *ls /usr* would become *ls -1 /usr*, the argument list here being undisturbed. If the alias for *lookup* were *grep !^ /etc/passwd* then *lookup bill* would become *grep bill /etc/passwd*. Aliases can be used to introduce parser metasyntax. For example, alias print ‘pr !$ | lpr’ defines a command (print) which prints its arguments to the line printer.

Alias substitution is repeated until the first word of the command has no alias. If an alias substitution does not change the first word (as in the previous example) it is flagged to prevent a loop. Other loops are detected and cause an error.
Some aliases are referred to by the shell; see “tcsh built-in commands” on page 695.

Variable substitution

The tcsh shell maintains a list of variables, each of which has as value a list of zero or more words. The values of tcsh shell variables can be displayed and changed with the set and unset commands. The system maintains its own list of "environment" variables. These can be displayed and changed with printenv, setenv and unsetenv.

Variables can be made read-only with set -r. Read-only variables cannot be modified or unset; attempting to do so will cause an error. Once made read-only, a variable cannot be made writable, so set -r should be used with caution. Environment variables cannot be made read-only.

Some variables are set by the tcsh shell or referred to by it. For instance, the argv variable is an image of the shell’s argument list, and words of this variable's value are referred to in special ways. Some of the variables referred to by the tcsh shell are toggles; the shell does not care what their value is, only whether they are set or not. For instance, the verbose variable is a toggle which causes command input to be echoed. The -v command line option sets this variable. Special shell variables lists all variables which are referred to by the shell.

Other operations treat variables numerically. The @ (at) command permits numeric calculations to be performed and the result assigned to a variable. Variable values are, however, always represented as (zero or more) strings. For the purposes of numeric operations, the null string is considered to be zero, and the second and subsequent words of multiword values are ignored.

After the input line is aliased and parsed, and before each command is executed, variable substitution is performed keyed by $ characters. This expansion can be prevented by preceding the $ with a \ except within double quotation marks (") where it always occurs, and within single quotation marks (’) where it never occurs. Strings quoted by backward quotation marks or accents (´) are interpreted later (see Command substitution) so $ substitution does not occur there until later, if at all. A $ is passed unchanged if followed by a blank, tab, or end-of-line.

Input/output redirections are recognized before variable expansion, and are variable expanded separately. Otherwise, the command name and entire argument list are expanded together. It is thus possible for the first (command) word (to this point) to generate more than one word, the first of which becomes the command name, and the rest of which become arguments.

Unless enclosed in double quotation marks (") or given the :q modifier the results of variable substitution can eventually be command and file name substituted. Within ", a variable whose value consists of multiple words expands to a (portion of a) single word, with the words of the variable's value separated by blanks. When the :q modifier is applied to a substitution the variable will expand to multiple words with each word separated by a blank and quoted to prevent later command or file name substitution.

The following metasequences are provided for introducing variable values into the shell input. Except as noted, it is an error to reference a variable which is not set.

$\{name[selector]\}

Substitutes only the selected words from the value of name. The selector is subjected to $ substitution and can consist of a single number or two numbers separated by a - (hyphen). The first word of a variable's value is numbered 1. If the first number of a range is omitted it defaults to 1. If the last member of a range is omitted it defaults to $#name. The selector * selects all words. It is not an error for a range to be empty if the second argument is omitted or in range.

$0

Substitutes the name of the file from which command input is being read. An error occurs if the name is not known.

$\{number\}

Equivalent to $argv[number].
$*
   Equivalent to $argv, which is equivalent to $argv[\ast].

The : (colon) modifiers described in History substitution, except for :p, can be applied to the Variable substitution. More than one can be used. Braces might be needed to insulate a variable substitution from a literal colon just as with history substitution; any modifiers must appear within the braces. The following substitutions cannot be modified with : modifiers.

$?name
${?name}
   Substitutes the string 1 if name is set, 0 if it is not.

$0
   Substitutes the name of the file from which command input is being read. An error occurs if the name is not known.

$?0
   Substitutes 1 if the current input file name is known, 0 if it is not. Always 0 in interactive shells.

$#name or ${#name}
   Substitutes the number of words in name.

$#
   Equivalent to '$#argv'.

$%name
${%name}
   Substitutes the number of characters in name.

$%number
${%number}
   Substitutes the number of characters in $argv[number].

$?
   Equivalent to $status.

$$
   Substitutes the (decimal) process number of the (parent) shell.

$!
   Substitutes the (decimal) process number of the last background process started by this shell.

$<
   Substitutes a line from the standard input, with no further interpretation thereafter. It can be used to read from the keyboard in a shell script. While csh always quotation marks $<, as if it were equivalent to $<q, tcsh does not. Furthermore, when tcsh is waiting for a line to be typed the user can type an interrupt to interrupt the sequence into which the line is to be substituted, but csh does not allow this.

The editor command expand-variables, normally bound to ^X-$, can be used to interactively expand individual variables.

The remaining substitutions are applied selectively to the arguments of tcsh built-in commands. This means that portions of expressions that are not evaluated are not subjected to these expansions. For commands that are not internal to the tcsh shell, the command name is substituted separately from the argument list. This occurs very late, after input-output redirection is performed, and in a child of the main shell.

Command substitution

Command substitution is indicated by a command enclosed in ''. The output from such a command is broken into separate words at blanks, tabs and newlines, and null words are discarded. The output is variable and command substituted and put in place of the original string.
Command substitutions inside double quotation marks ("") retain blanks and tabs; only newlines force new words. The single final newline does not force a new word in any case. It is thus possible for a command substitution to yield only part of a word, even if the command outputs a complete line.

**File name substitution**

If a word contains any of the characters `*`, `?`, `[` or `{` or begins with the character `~` it is a candidate for file name substitution, also known as globbing. This word is then regarded as a pattern (glob-pattern), and replaced with an alphabetically sorted list of file names which match the pattern.

In matching file names, the character . (period) at the beginning of a file name or immediately following a / (forward slash), as well as the character / must be matched explicitly. The character * matches any string of characters, including the null string. The character ? matches any single character. The sequence [...] matches any one of the characters enclosed. Within [...], a pair of characters that are separated by - matches any character lexically between the two.

Some glob-patterns can be negated: The sequence [^...] matches any single character that is not specified by the characters and ranges of characters in the braces.

An entire glob-pattern can also be negated with ^:

```bash
> echo *
  bang crash crunch ouch
> echo ^cr*
  bang ouch
```

Glob-patterns which do not use `?`, `*` or `[]` or which use `{}` or `^` are not negated correctly.

The metanotation a{b,c,d}e is a shorthand for abe ace ade.

- Left-to-right order is preserved: `/usr/source/s1/{oldls,ls}s.c` expands to `/usr/source/s1/oldls.c /usr/source/s1/ls.s.c`. The results of matches are sorted separately at a low level to preserve this order, such as, like the following example, where `../{memo,*box}` might expand to `../memo ..../box ../mbox`. (Note that 'memo' was not sorted with the results of matching 'box'.) It is not an error when this construct expands to files that do not exist, but it is possible to get an error from a command to which the expanded list is passed. This construct can be nested. As a special case the words {}, and {} are passed undisturbed.

- The character ~ at the beginning of a file name refers to home directories. Standing alone, for example, ~, it expands to the invoker’s home directory as reflected in the value of the home shell variable. When followed by a name consisting of letters, digits and - (hyphen) characters the shell searches for a user with that name and substitutes their home directory; thus ~ken might expand to `/usr/ken` and ~ken/chmach to `/usr/ken/chmach`.

- If the character ~ is followed by a character other than a letter or / or appears elsewhere than at the beginning of a word, it is left undisturbed. A command like `setenv MANPATH /usr/man:/usr/local/man:-~/lib/man` does not, therefore, do home directory substitution as one might hope. It is an error for a glob-pattern containing `*`, `?`, `[` or `~`, with or without `^`, not to match any files. However, only one pattern in a list of glob-patterns must match a file (so that, for example, `rm *a *.c *.o` would fail only if there were no files in the current directory ending in `'a'`, `'c'`, or `'o'`), and if the nonomatch shell variable is set a pattern (or list of patterns) which matches nothing is left unchanged instead of causing an error.

The `noglob` shell variable can be set to prevent file name substitution, and the `expand-glob` editor command, normally bound to `^X-*`, can be used to interactively expand individual file name substitutions.

**Directory stack substitution**

The directory stack is a list of directories numbered from zero that is used by the pushd, popd and dirs built-in commands for tcsd. dirs can print, store in a file, restore, and clear the directory stack at any time. The savedirs and dirsfile shell variables can be set to store the directory stack automatically on
logout and restore it on login. The `dirstack` shell variable can be examined to see the directory stack and set to put arbitrary directories into the directory stack.

The character `=` (equal) followed by one or more digits expands to an entry in the directory stack. The special case `=-` expands to the last directory in the stack. For example,

```bash
> dirs -v
0 /usr/bin
1 /usr/spool/uucp
2 /usr/accts/sys
> echo =1
/usr/spool/uucp
> echo =0/calendar
/usr/bin/calendar
> echo =-
/usr/accts/sys
```

The `noglob` and `nonomatch` shell variables and the `expand-glob` editor command apply to directory stack as well as file name substitutions.

**Other substitutions**

There are several more transformations involving file names, not strictly related to the “Directory stack substitution” on page 686, but mentioned here for completeness. Any file name can be expanded to a full path when the `symlinks` variable is set to expand. Quoting prevents this expansion, and the `normalize-path` editor command does it on demand. The `normalize-command` editor command expands commands in `PATH` into full paths on demand. Finally, `cd` and `pushd` interpret `-` (hyphen) as the old working directory (equivalent to the tcsh shell variable `owd`). This is not a substitution, but an abbreviation recognized only by those commands. Nonetheless, it too can be prevented by quoting.

**Command execution**

The next three topics describe how the shell executes commands and deals with their input and output.

**Built-in and non-built-in command execution**

Built-in commands for tcsh are executed within the shell. If any component of a pipeline except the last is a built-in command, the pipeline is executed in a subshell.

Parenthesized commands are always executed in a subshell:

```bash
(cd; pwd); pwd
```

which prints the home directory, leaving you where you were (printing this after the home directory), while

```bash
cd; pwd
```

leaves you in the home directory. Parenthesized commands are most often used to prevent `cd` from affecting the current shell.

When a command to be executed is found not to be a built-in command the tcsh shell attempts to execute the command via `execve`. Each word in the variable path names a directory in which the tcsh shell will look for the command. If it is given neither a `-c` nor a `-t` option, the shell hashes the names in these directories into an internal table so that it will only try an `execve` in a directory if there is a possibility that the command resides there. This greatly speeds command location when a large number of directories are present in the search path. If this mechanism has been turned off (via `unhash`), if the shell was given a `-c` or `-t` argument or in any case for each directory component of path which does not begin with a `/`, the shell concatenates the current working directory with the given command name to form a path name of a file which it then attempts to execute.
If the file has execute permissions but is not an executable to the system (that is, it is neither an executable binary nor a script which specifies its interpreter), then it is assumed to be a file containing shell commands and a new shell is spawned to read it. The `shell` special alias can be set to specify an interpreter other than the shell itself.

**Input or output**

The standard input and standard output of a command can be redirected with the following syntax listed in Table 32 on page 688.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt; name</code></td>
<td>Open file name (which is first variable, command and file name expanded) as the standard input.</td>
</tr>
<tr>
<td><code>&lt;&lt;word</code></td>
<td>Read the shell input up to a line which is identical to <code>word</code>. <code>word</code> is not subjected to variable, file name or command substitution, and each input line is compared to <code>word</code> before any substitutions are done on this input line. Unless a quoting <code>, &quot;</code>, <code>' </code> or <code>' </code> appears in <code>word</code> variable and command substitution is performed on the intervening lines, allowing <code>\</code> to quote <code>$</code>, <code>\</code> and ` ' (single quotation mark). Commands that are substituted have all blanks, tabs, and newlines preserved, except for the final newline which is dropped. The resultant text is placed in an anonymous temporary file which is given to the command as standard input.</td>
</tr>
<tr>
<td><code>&gt; name</code></td>
<td>The file <code>name</code> is used as standard output. If the file does not exist then it is created; if the file exists, its is overwritten and, therefore, the previous contents are lost.</td>
</tr>
<tr>
<td><code>! name&gt;</code></td>
<td>If the shell variable <code>noclobber</code> is set, then the file must not exist or be a character special file (for example, a terminal or <code>/dev/null</code>) or an error results. This helps prevent accidental destruction of files. In this case the <code>!</code> forms can be used to suppress this check.</td>
</tr>
<tr>
<td><code>&amp; name</code></td>
<td>The forms involving <code>&amp;</code> (ampersand) route the diagnostic output into the specified file as well as the standard output. <code>name</code> is expanded in the same way as `&lt; input file names are.</td>
</tr>
<tr>
<td><code>&gt;&amp; name</code></td>
<td>Like <code>&gt;</code>, but appends output to the end of <code>name</code>. If the shell variable <code>noclobber</code> is set, then it is an error for the file not to exist, unless one of the <code>!</code> forms is given.</td>
</tr>
<tr>
<td><code>&gt;&gt; name</code></td>
<td>Like <code>&gt;</code>, but appends output to the end of <code>name</code>. If the shell variable <code>noclobber</code> is set, then it is an error for the file not to exist, unless one of the <code>!</code> forms is given.</td>
</tr>
<tr>
<td><code>! name&gt;&gt;</code></td>
<td>Like <code>&gt;</code>, but appends output to the end of <code>name</code>. If the shell variable <code>noclobber</code> is set, then it is an error for the file not to exist, unless one of the <code>!</code> forms is given.</td>
</tr>
<tr>
<td><code>&amp;! name</code></td>
<td>The forms involving <code>&amp;</code> (ampersand) route the diagnostic output into the specified file as well as the standard output. <code>name</code> is expanded in the same way as `&lt; input file names are.</td>
</tr>
</tbody>
</table>

A command receives the environment in which the shell was invoked as modified by the input-output parameters and the presence of the command in a pipeline. Thus, unlike some previous shells, commands run from a file of shell commands have no access to the text of the commands by default; instead they receive the original standard input of the shell. The `<<` mechanism should be used to present inline data. This permits shell command scripts to function as components of pipelines and allows the shell to block read its input. The default standard input for a command run detached is not the empty file `/dev/null`, but the original standard input of the shell. If this is a terminal and if the process attempts to read from the terminal, then the process will block and the user is notified (see “Jobs” on page 693).

Diagnostic output can be directed through a pipe with the standard output. Simply use the form `|&` instead of just `|`.
The shell cannot presently redirect diagnostic output without also redirecting standard output, but (command > output-file) >& error-file is often an acceptable workaround. Either output-file or error-file can be /dev/tty to send output to the terminal.

Features
Having described how the shell accepts, parses and executes command lines, we now turn to a variety of its useful features.

Control flow
The tcsh shell contains a number of commands that can be used to regulate the flow of control in command files (shell scripts) and (in limited by useful ways) from terminal output. These commands all operate by forcing the shell to reread or skip in its input and, due to the implementation, restrict the placement of some of the commands.

The foreach, switch, and while statements, as well as the if-then-else form of the if statement, require that the major keywords appear in a single simple command on an input line.

If the shell's input is not seekable, the shell buffers up input whenever a loop is being read and performs seeks in this internal buffer to accomplish the rereading implied by the loop. (To the extent that this allows, backward gotos will succeed on non-seekable inputs.)

Expressions
The if, while, and exit built-in commands use expressions with a common syntax. The expressions can include any of the operators described in the next three topics. Note that the @ built-in command has its own separate syntax.

Logical, arithmetical, and comparison operators
These operators are similar to those of C and have the same precedence. They include:

| | & & ^ == != =~ !~ <= >= < > << >> + - * / % ~ ( ) |

Here the precedence increases to the right, '==' '!=' '~=~=' and '!~='; '<=' '>=~=' '<' and '>'; '<<<' and '>>>'; '+' and '-' being in groups, at the same level. The '==' '!=~=' '~=~=' '!~=' operators compare their arguments as strings; all others operate on numbers. The operators '~=~=' and '!~=' are like '!=' and '==' except that the right hand side is a glob-pattern (see File name substitution) against which the left hand operand is matched. This reduces the need for use of the switch built-in command in shell scripts when all that is really needed is pattern matching.

Strings that begin with 0 are considered octal numbers. Null or missing arguments are considered 0. The results of all expressions are strings, which represent decimal numbers. No two components of an expression can appear in the same word; except when adjacent to components of expressions that are syntactically significant to the parser ('$' '|' '<' '>' '(' ')') they must be surrounded by spaces.

Command exit status
Commands can be executed in expressions and their exit status returned by enclosing them in braces ({}). The braces must be separated from the words of the command by spaces. Command executions succeed, returning true, that is, 1, if the command exits with status 0, otherwise they fail, returning false (0). If more detailed status information is required, then the command must be executed outside of an expression and the status shell variable examined.

Shell command descriptions 689
**File inquiry operators**

Some of these operators perform true/false tests on files and related objects. They are of the form `-op file`, where `op` is one of:

- `ac`: An extended ACL of type `c` exists. Character `c` represents the type of ACL:
  - `a`: Access ACL
  - `d`: Directory default ACL
  - `f`: File default ACL
  Testing with suboptions `d` and `f` will always return false for files (files do not have default ACLs).
- `r`: Read access (as determined by security product and effective IDs)
- `w`: Write access (as determined by security product and effective IDs)
- `x`: Execute access (as determined by security product and effective IDs)
- `X`: Executable in the path or shell built-in. For example, `-X ls` and `-X ls-F` are generally true, but `-X /bin/ls` is not. (This is determined by security product and effective ids.)
- `e`: Existence
- `Ea`: File has the APF extended attribute
- `Ep`: File has the program
- `Es`: File has the shared address space extended attribute
- `El`: File has the shared library extended attribute
- `o`: Ownership
- `x`: Zero size
- `s`: Nonzero size
- `f`: Plain file
- `d`: Directory
- `l`: Symbolic link
- `b`: Block special file
- `c`: Character special file
- `p`: Named pipe (fifo)
S
  Socket special file
u
  Set-user ID bit is set
g
  Set-group-ID bit is set
k
  Sticky bit is set
t
  t file_descriptor (which must be a digit) is an open file descriptor for a terminal device
L
  Applies subsequent operators in a multiple-operator test to a symbolic link instead of to the file to
  which the link points

file is command and file name expanded and then tested to see if it has the specified relationship to the
real user. If file does not exist or is inaccessible or, for the operators indicated by *, if the specified file
type does not exist on the current system, then all inquiries return false (0).

These operators can be combined for conciseness: -xy file is equivalent to -x file & & -y file. For example,
- fx is true (returns 1) for plain executable files, but not for directories.

L can be used in a multiple-operator test to apply subsequent operators to a symbolic link instead of to
the file to which the link points. For example, - lLo is true for links owned by the invoking user. Lr, Lw,
and Lx are always true for links and false for non-links. L has a different meaning when it is the last
operator in a multiple-operator test.

It is possible but not useful, and sometimes misleading, to combine operators which expect file to be a
file with operators which do not (for example, X and t). Following L with a non-file operator can lead to
particularly strange results.

Other operators return other information, that is not just 0 or 1. They have the same format as before
where op can be one of the following choices:

A
  Last file access time, as the number of seconds since epoch.
A:
  Like A, but in time stamp format, that is, 'Fri May 14 16:36:10 1993'.
M
  Last file modification time.
M:
  Like M, but in time stamp format.
C
  Last inode modification time.
C:
  Like C, but in time stamp format.
D
  Device number.
I
  Inode number.
F
  Composite file identifier, in the form device : inode.
L
  The name of the file pointed to by a symbolic link.
N
  Number of (hard) links.
P
Permissions, in octal, without leading zeros.

P: Like P, with leading zeros.

P mode
Equivalent to -P mode & file, that is, -P22 file returns 22 if file is writable by group and other, 20 if by group only, and 0 if by neither.

P mode:
Like P mode, with leading zeros.

U
Numeric user ID.

U:
Username, or the numeric user ID if the username is unknown.

G
Numeric group ID.

G:
Group name, or the numeric group ID if the group name is unknown.

Z
Size in bytes.

m file
Returns the security label of the file if one exists. Otherwise, returns false.

Only one of these operators can appear in a multiple-operator test, and it must be the last. L has a different meaning at the end of and elsewhere in a multiple-operator test. Because 0 is a valid return value for many of these operators, they do not return 0 when they fail: most return -1, and F returns : (colon).

File inquiry operators can also be evaluated with the filetest built-in command.

File inquiry operators for use with file tagging and the filetest built-in command

-B file
• True if the file is tagged as binary.
• False if the file is not tagged or tagged as text.
• Returns codeset if the file is tagged as mixed text and binary, that is, txtflag = OFF and codeset stored in file tag.

-T file
• False if the file is not tagged or if it is tagged as txtflag = OFF.
• Returns codeset if the file is tagged as text.

Either -B file or -T file will allow a tcsh "if test" to evaluate to true when the file is tagged as indicated. These two operators will also allow tcsh to test for a specific code set. For example,

```bash
if ( -T file == IBM-1047 ) # True if tagged as IBM-1047 text
if ( -B file ) # True if tagged as binary
```

Note: Code sets that are aliases of each other exist, which might cause the test to fail, because the file inquiry operator might return an alias of the code set that you are testing.
Jobs

The shell associates a job with each pipeline. It keeps a table of current jobs, printed by the jobs command, and assigns them small integer numbers. When a job is started asynchronously with & (ampersand), the shell prints a line which looks like

```
[1] 1234
```

indicating that the job, which was started asynchronously, was job number 1 and had one (top-level) process, whose process ID was 1234.

If you are running a job and want to do something else you can press the suspend key (typically ^Z), which sends a STOP signal to the current job. The shell will then normally indicate that the job has been 'suspended and print another prompt. If the listjobs shell variable is set, all jobs is listed like the jobs built-in command; if it is set to 'long' the listing is in long format, like jobs -l. You can then manipulate the state of the suspended job. You can put it in the background with the bg command or run some other commands and eventually bring the job back into the foreground with fg. (See also the run-fg-editor editor command.) A ^Z takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is typed. The wait built-in command causes the shell to wait for all background jobs to complete.

The ^] key sends a delayed suspend signal, which does not generate a STOP signal until a program attempts to read it, to the current job. This can be typed ahead when you have prepared some commands for a job that you want to stop after it has read them. The ^Y key performs this function in csh; in tcsh, ^Y is an editing command.

A job being run in the background stops if it tries to read from the terminal. Background jobs are typically allowed to produce output, but this can be disabled by giving the command stty tostop. If you set the stty option, then background jobs will stop when they try to produce output like they do when they try to read input.

There are several ways to refer to jobs in the shell. The character % introduces a job name. If you want to refer to job number 1, you can name it as %1. Just naming a job brings it to the foreground; thus %' is a synonym for fg %1, bringing job 1 back into the foreground. Similarly, saying %1 & resumes job 1 in the background, just like bg %1. A job can also be named by an unambiguous prefix of the string typed in to start it: %ex would normally restart a suspended ex job, if there were only one suspended job whose name began with the string 'ex'. It is also possible to say %? string to specify a job whose text contains string, if there is only one such job.

The shell maintains a notion of the current and previous jobs. In output pertaining to jobs, the current job is marked with a + (plus) and the previous job with a - (hyphen). The abbreviations %+, %, and (by analogy with the syntax of the history mechanism) %% all refer to the current job, and %- refers to the previous job.

The job control mechanism requires that the stty option new be set on some systems. It is an artifact from a new implementation of the tty driver that allows generation of interrupt characters from the keyboard to tell jobs to stop. See stty and the setty tcsh built-in command for details on setting options in the new tty driver.

Status reporting

The tcsh shell learns immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work. If, however, you set the shell variable notify, the shell will notify you immediately of changes of status in background jobs. There is also a shell command notify which marks a single process so that its status changes are immediately reported. By default notify marks the current process; simply say 'notify' after starting a background job to mark it.

When you try to leave the shell while jobs are stopped, you are warned that 'You have stopped jobs.' You can use the jobs command to see what they are. If you do this or immediately try to exit again, the shell will not warn you a second time and the suspended jobs are terminated.
Automatic, periodic, and timed events

There are various ways to run commands and take other actions automatically at various times in the life cycle of the shell.

- The `sched` built-in command puts commands in a scheduled-event list, to be executed by the shell at a given time.
- The `beepcmd`, `cwdcmd`, `periodic` and `precmd` special aliases can be set, respectively, to execute commands when the shell wants to ring the bell, when the working directory changes, every t-period minutes and before each prompt.
- The `autologout` shell variable can be set to log out of the shell after a given number of minutes of inactivity.
- The `mail` shell variable can be set to check for new mail periodically.
- The `printexitvalue` shell variable can be set to print the exit status of commands which exit with a status other than zero.
- The `rmstar` shell variable can be set to ask the user, when `rm *` is typed, if that is really what was meant.
- The `time` shell variable can be set to execute the `time` built-in command after the completion of any process that takes more than a given number of CPU seconds.
- The `watch` and `who` shell variables can be set to report when selected users log in or out, and the `log` built-in command reports on those users at any time.

National language system report

When using the system's multicultural support, the `setlocale` function is called to determine appropriate character classification and sorting. This function typically examines the `LANG` and `LC_CTYPE` environment variables; refer to the system documentation for further details.

Unknown characters (those that are neither printable nor control characters) are printed in the format `\nnn`.

The version shell variable indicates what options were chosen when the shell was compiled. Note also the `newgrp` built-in and `echo_style` shell variable and the locations of the shell's input files (see “tcsh files” on page 713).

**Restriction:** The tcsh shell does not support three locales. They are IBM-1388 (Chinese), IBM-933 (Korean) and IBM-937 (Traditional Chinese).

Handling signals

Login shells ignore interrupts when reading the file `~/.logout`. The shell ignores quit signals unless started with `-q`. Login shells catch the terminate signal, but non-login shells inherit the terminate behavior from their parents. Other signals have the values that the shell inherited from its parent.

In shell scripts, the shell's handling of interrupt and terminate signals can be controlled with `onintr`, and its handling of hangups can be controlled with `hup` and `nohup`.

The shell exits on a hangup (see also the `logout` shell variable). By default, the shell's children do too, but the shell does not send them a hangup when it exits. `hup` arranges for the shell to send a hangup to a child when it exits, and `nohup` sets a child to ignore hangups.
Managing terminals

The shell uses three different sets of terminal (tty) modes: edit, used when editing, quote, used when quoting literal characters, and execute, used when executing commands. The shell holds some settings in each mode constant, so commands which leave the tty in a confused state do not interfere with the shell. The shell also matches changes in the speed and padding of the tty. The list of tty modes that are kept constant can be examined and modified with the setty built-in. Although the editor uses CBREAK mode (or its equivalent), it takes typed-ahead characters anyway.

The echotc, settc and telltc commands can be used to manipulate and debug terminal capabilities from the command line.

The tcsh shell adapts to window resizing automatically and adjusts the environment variables LINES and COLUMNS if set.

tcsh built-in commands

The following table lists the tcsh built-in commands, which are not /bin/sh built-ins.

<table>
<thead>
<tr>
<th>@ (at)</th>
<th>filetest</th>
<th>notify</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>glob</td>
<td>onintr</td>
<td>telltc</td>
</tr>
<tr>
<td>alloc</td>
<td>hashstat</td>
<td>popd</td>
<td>uncomplete</td>
</tr>
<tr>
<td>bindkey</td>
<td>hup</td>
<td>pushd</td>
<td>(unhash</td>
</tr>
<tr>
<td>builtins</td>
<td>limit</td>
<td>rehash</td>
<td>unlimited</td>
</tr>
<tr>
<td>bye</td>
<td>log</td>
<td>repeat</td>
<td>unsetenv</td>
</tr>
<tr>
<td>chdir</td>
<td>login</td>
<td>sched</td>
<td>watchlog</td>
</tr>
<tr>
<td>complete</td>
<td>logout</td>
<td>setenv</td>
<td>where</td>
</tr>
<tr>
<td>dirs</td>
<td>ls-F</td>
<td>settc</td>
<td>which</td>
</tr>
<tr>
<td>echotc</td>
<td>notify</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other tcsh built-in commands are also found in the z/OS shell. In some cases, they might differ in function; see the specific command description for a discussion of the tcsh version of the command.

<table>
<thead>
<tr>
<th>: (colon)</th>
<th>continue</th>
<th>fg</th>
<th>nice</th>
<th>stop</th>
<th>unset</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>echo</td>
<td>history</td>
<td>nohup</td>
<td>suspend</td>
<td>wait</td>
</tr>
<tr>
<td>bg</td>
<td>eval</td>
<td>jobs</td>
<td>printenv</td>
<td>time</td>
<td>writedown</td>
</tr>
<tr>
<td>break</td>
<td>exec</td>
<td>kill</td>
<td>set</td>
<td>umask</td>
<td></td>
</tr>
<tr>
<td>cd</td>
<td>exit</td>
<td>newgrp</td>
<td>shift</td>
<td>unalias</td>
<td></td>
</tr>
</tbody>
</table>

As well as built-in commands, the tcsh shell has a set of special aliases:

beepcmd
cwdcmd
periodic
precmd
shell

If set, each of these aliases executes automatically at the indicated time. They are initially undefined. For more information about aliases, see Alias substitution.

Descriptions of these aliases are as follows:
beepcmd
   Runs when the shell wants to ring the terminal bell.

cwdcmd
   Runs after every change of working directory. For example, if the user is working on an X window
   system using xterm and a re-parenting window manager that supports title bars such as twm and
does

   > alias cwdcmd 'echo -n "^[]2;${HOST}:$cwd ^G"'

   then the shell will change the title of the running xterm to be the name of the host, a colon, and the
   full current working directory. A fancier way to do that is

   > alias cwdcmd 'echo -n "^[]2;${HOST}:$cwd^G^[]1;${HOST}^G"'

   This will put the host name and working directory on the title bar but only the host name in the icon
   manager menu. Putting a cd, pushd or popd in cwdcmd might cause an infinite loop.

periodic
   Runs every tperiod minutes. This provides a convenient means for checking on common but
   infrequent changes such as new mail. For example, if one does

   > set tperiod = 30
   > alias periodic checknews

   then the checknews program runs every 30 minutes. If periodic is set but tperiod is unset or set to
   0, periodic behaves like precmd.

precmd
   Runs just before each prompt is printed. For example, if one does

   > alias precmd date

   then date runs just before the shell prompts for each command. There are no limits on what precmd
   can be set to do, but discretion should be used.

shell
   Specifies the interpreter for executable scripts that do not themselves specify an interpreter. The first
   word should be a full path name to the interpreter. For example: /bin/tcsh or /usr/local/bin/
   tcsh (by default, this is set to /bin/tcsh).

Programming constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>breaksw</td>
<td>Causes a break from a switch, resuming after the endsw.</td>
</tr>
<tr>
<td>case label</td>
<td>A label in a switch. See the switch built-in description.</td>
</tr>
<tr>
<td>continue</td>
<td>Continues execution of the nearest enclosing while or foreach. The rest of</td>
</tr>
<tr>
<td></td>
<td>the commands on the current line are executed.</td>
</tr>
<tr>
<td>default</td>
<td>Labels the default case in a switch statement. It should come after all case</td>
</tr>
<tr>
<td></td>
<td>labels.</td>
</tr>
<tr>
<td>else</td>
<td>See the description of the foreach, if, switch, and while statements</td>
</tr>
<tr>
<td>end</td>
<td>that follow.</td>
</tr>
<tr>
<td>endif</td>
<td></td>
</tr>
<tr>
<td>endsw</td>
<td></td>
</tr>
<tr>
<td>Construct</td>
<td>Action</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>goto word</strong></td>
<td>With goto, <em>word</em> is file name and command substituted to yield a string of the form label. The tcsh shell rewinds its input as much as possible, searches for a line of the form <em>label</em>, possible preceded by blanks or tabs, and continues execution after that line.</td>
</tr>
<tr>
<td><strong>foreach</strong></td>
<td>Successively sets the variable name to each member of wordlist and executes the sequence of commands between this command and the matching end. (Both foreach and end must appear alone on separate lines.) The built-in command <em>continue</em> can be used to continue the loop prematurely and the built-in command <em>break</em> to terminate it prematurely. When this command is read from the terminal, the loop is read once prompting with <strong>foreach</strong>? (or <strong>prompt2</strong>) before any statements in the loop are executed. If you make a mistake typing in a loop at the terminal you can rub it out.</td>
</tr>
<tr>
<td><strong>if (expr) then</strong></td>
<td>If the specified <em>expr</em> is true then the commands to the first else are executed; otherwise if <em>expr2</em> is true then the commands to the second else are executed. Any number of <strong>else-if</strong> pairs are possible; only one <strong>endif</strong> is needed. The else part is optional. (The words else and endif must appear at the beginning of input lines; the <strong>if</strong> must appear alone on its input line or after an <strong>else</strong>.)</td>
</tr>
<tr>
<td><strong>switch (string)</strong></td>
<td>Each case label is successively matched, against the specified <em>string</em> which is first command and file name expanded. The file metacharacters *, ?, and [...] can be used in the case labels, which are variable expanded. If none of the labels match before a default label is found, then the execution begins after the default label. Each case label and the default label must appear at the beginning of a line. The command <strong>breaksw</strong> causes execution to continue after the endsw. Otherwise control might fall through case labels and default labels as in C. If no label matches and there is no default, execution continues after the endsw.</td>
</tr>
<tr>
<td><strong>while (expr)</strong></td>
<td>Executes the commands between the while and the matching end while <em>expr</em> (expression) evaluates nonzero. while and end must appear alone on their input lines. <strong>break</strong> and <strong>continue</strong> can be used to terminate or continue the loop prematurely. If the input is a terminal, the user is prompted the first time through the loop as with foreach.</td>
</tr>
</tbody>
</table>

**tcsh shell and environment variables**

The variables described in this topic have special meaning to the tcsh shell. The tcsh shell sets **addsuffix**, **argv**, **autologout**, **command**, **echo_style**, **edit**, **gid**, **group**, **home**, **loginsh**, **path**, **prompt**, **prompt2**, **prompt3**, **shell**, **shlvl**, **tcsh**, **term**, **tty**, **uid**, **user**, and **version** at startup. They do not change thereafter, unless changed by the user. The tcsh shell updates **cwd**, **dirstack**, **owd**, and **status** when necessary, and sets **logout** on logout.

The shell synchronizes **group**, **home**, **path**, **shlvl**, **term**, and **user** with the environment variables of the same names: whenever the environment variable changes the shell changes the corresponding shell variable to match (unless the shell variable is read-only) and vice versa. Although **cwd** and PWD have identical meanings, they are not synchronized in this manner.

The shell automatically interconverts the different formats of path and PATH.
Table 33: tcsh built-in shell variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>addsuffix</td>
<td>If set, file name completion adds / to the end of directories and a</td>
</tr>
<tr>
<td></td>
<td>space to the end of normal files.</td>
</tr>
<tr>
<td>ampm</td>
<td>This variable gives a user the ability to alter the time format in their</td>
</tr>
<tr>
<td></td>
<td>tcsh prompt. Specifically, <code>ampm</code> overrides the <code>%T</code> and <code>%P</code> formatting</td>
</tr>
<tr>
<td></td>
<td>sequences in a user’s prompt. If set, all times are shown in 12-hour</td>
</tr>
<tr>
<td></td>
<td>AM/PM format.</td>
</tr>
<tr>
<td>argv</td>
<td>The arguments to the shell. Positional parameters are taken from</td>
</tr>
<tr>
<td></td>
<td><code>argv</code>. For example, <code>$1</code> is replaced by <code>$argv</code>. Set by default, but</td>
</tr>
<tr>
<td></td>
<td>typically empty in interactive shells.</td>
</tr>
<tr>
<td>autocorrect</td>
<td>If set, the <code>spell-word</code> editor command is invoked automatically</td>
</tr>
<tr>
<td></td>
<td>before each completion. (This variable is not implemented.)</td>
</tr>
<tr>
<td>autoexpand</td>
<td>If set, the <code>expand-history</code> editor command is invoked automatically</td>
</tr>
<tr>
<td></td>
<td>before each completion attempt.</td>
</tr>
<tr>
<td>autolist</td>
<td>If set, possibilities are listed after an ambiguous completion. If set to</td>
</tr>
<tr>
<td></td>
<td><code>ambiguous</code>, possibilities are listed only when no new characters are</td>
</tr>
<tr>
<td></td>
<td>added by completion.</td>
</tr>
<tr>
<td>autologout</td>
<td>Set to the number of minutes of inactivity before automatic logout.</td>
</tr>
<tr>
<td></td>
<td>Automatic locking is an unsupported feature on the z/OS platform. If</td>
</tr>
<tr>
<td></td>
<td>you specify a second parameter on the <code>autologout</code> statement (intending</td>
</tr>
<tr>
<td></td>
<td>it to be for autolock), this parameter will be assigned to</td>
</tr>
<tr>
<td></td>
<td><code>autologout</code>. When the shell automatically logs out, it prints</td>
</tr>
<tr>
<td></td>
<td><code>autologout</code>, sets the variable logout to automatic and exits. Set to</td>
</tr>
<tr>
<td></td>
<td>60 (automatic logout after 60 minutes) by default in login and</td>
</tr>
<tr>
<td></td>
<td>superuser shells, but not if the shell thinks it is running under a</td>
</tr>
<tr>
<td></td>
<td>window system (the DISPLAY environment variable is set), or the tty is</td>
</tr>
<tr>
<td></td>
<td>a pseudo-tty (pty). See also the <code>logout</code> shell variable.</td>
</tr>
<tr>
<td>backslash_quote</td>
<td>If set, backslashes () always quote , ' (single quotation mark) and</td>
</tr>
<tr>
<td></td>
<td>&quot; (double quotation mark). This might make complex quoting tasks</td>
</tr>
<tr>
<td></td>
<td>easier, but it can cause syntax errors in csh scripts.</td>
</tr>
<tr>
<td>cdpath</td>
<td>A list of directories in which <code>cd</code> should search for subdirectories if</td>
</tr>
<tr>
<td></td>
<td>they aren't found in the current directory.</td>
</tr>
<tr>
<td>command</td>
<td>If set, the command that was passed to the shell with the <code>-c</code> flag.</td>
</tr>
<tr>
<td>complete</td>
<td>If set to <code>enhance</code>, completion first ignores case and then considers</td>
</tr>
<tr>
<td></td>
<td>periods, hyphens and underscores (’,’-‘ and ‘_’) to be word separators</td>
</tr>
<tr>
<td></td>
<td>and hyphens and underscores to be equivalent.</td>
</tr>
<tr>
<td>correct</td>
<td>If set to <code>cmd</code>, commands are automatically spelling-corrected. If set to</td>
</tr>
<tr>
<td></td>
<td><code>complete</code>, commands are automatically completed. If set to <code>all</code>, the</td>
</tr>
<tr>
<td></td>
<td>entire command line is corrected.</td>
</tr>
<tr>
<td>cwd</td>
<td>The full path name of the current directory. See also the <code>dirstack</code></td>
</tr>
<tr>
<td></td>
<td>and <code>cwd</code> shell variables.</td>
</tr>
<tr>
<td>dextract</td>
<td>If set, <code>pushd +n</code> extracts the <code>n</code>th directory from the directory stack</td>
</tr>
<tr>
<td></td>
<td>instead of rotating it to the top.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| **dirsfile** | The default location in which `dirs -S` and `dirs -L` look for a history file. If unset, `~/.cshdirs` is used. Because only `~/.tcshrc` is normally sourced before `~/.cshdirs`, `dirsfile` should be set in `~/.tcshrc` instead of `~/.login`. For example: 

```bash
set dirsfile = ~/.cshdirs
```

<p>| <strong>dirstack</strong> | An array of all the directories on the directory stack. <code>$dirstack[1]</code> is the current working directory, <code>$dirstack[2]</code> the first directory on the stack, etc. Note that the current working directory is <code>$dirstack[1]</code> but <code>=0</code> in directory stack substitutions, etc. One can change the stack arbitrarily by setting <code>dirstack</code>, but the first element (the current working directory) is always correct. See also the <code>cwd</code> and <code>owd</code> shell variables. |
| <strong>dunique</strong> | If set, <code>pushd</code> removes any instances of <code>name</code> from the stack before pushing it onto the stack. |
| <strong>echo</strong> | If set, each command with its arguments is echoed just before it is executed. For non-built-in commands all expansions occur before echoing. Built-in commands are echoed before command and file name substitution, since these substitutions are then done selectively. Set by the <code>-x</code> command line option. |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>echo_style</strong></td>
<td>The style of the echo built-in. Can be set to:</td>
</tr>
<tr>
<td></td>
<td><strong>bsd</strong></td>
</tr>
<tr>
<td></td>
<td><strong>sysv</strong></td>
</tr>
<tr>
<td></td>
<td><strong>both</strong></td>
</tr>
<tr>
<td></td>
<td><strong>none</strong></td>
</tr>
<tr>
<td></td>
<td>Set to both by default to the local system default.</td>
</tr>
<tr>
<td></td>
<td>The following is an example of this variable's use:</td>
</tr>
<tr>
<td></td>
<td>&gt; echo $echo_style</td>
</tr>
<tr>
<td></td>
<td>bsd</td>
</tr>
<tr>
<td></td>
<td>&gt; echo &quot;\n&quot;</td>
</tr>
<tr>
<td></td>
<td>&gt; echo -n &quot;test&quot;</td>
</tr>
<tr>
<td></td>
<td>test&gt;</td>
</tr>
<tr>
<td></td>
<td>&gt; set echo_style=sysv</td>
</tr>
<tr>
<td></td>
<td>echo $echo_style</td>
</tr>
<tr>
<td></td>
<td>sysv</td>
</tr>
<tr>
<td></td>
<td>echo &quot;\n&quot;</td>
</tr>
<tr>
<td></td>
<td>&gt; echo -n &quot;test&quot;</td>
</tr>
<tr>
<td></td>
<td>-n test</td>
</tr>
<tr>
<td></td>
<td>&gt; set echo_style=both</td>
</tr>
<tr>
<td></td>
<td>echo $echo_style</td>
</tr>
<tr>
<td></td>
<td>both</td>
</tr>
<tr>
<td></td>
<td>&gt; echo -n &quot;test&quot;</td>
</tr>
<tr>
<td></td>
<td>test&gt; echo &quot;\n&quot;</td>
</tr>
<tr>
<td></td>
<td>&gt;set echo_style=none</td>
</tr>
<tr>
<td></td>
<td>echo $echo_style</td>
</tr>
<tr>
<td></td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>&gt; echo -n &quot;test&quot;</td>
</tr>
<tr>
<td></td>
<td>-n test</td>
</tr>
<tr>
<td></td>
<td>&gt; echo &quot;\n&quot;</td>
</tr>
<tr>
<td></td>
<td>&gt;</td>
</tr>
<tr>
<td><strong>edit</strong></td>
<td>If set, the command-line editor is used. Set by default in interactive shells.</td>
</tr>
<tr>
<td><strong>ellipsis</strong></td>
<td>If set, the %c'/%. and %C prompt sequences (see the prompt shell variable) indicate skipped directories with an ellipsis (...) instead of /.</td>
</tr>
<tr>
<td><strong>fignore</strong></td>
<td>Lists file name suffixes to be ignored by completion.</td>
</tr>
<tr>
<td><strong>filec</strong></td>
<td>In the tcsh shell, completion is always used and this variable is ignored.</td>
</tr>
<tr>
<td><strong>gid</strong></td>
<td>The user's real group ID.</td>
</tr>
<tr>
<td><strong>group</strong></td>
<td>The user's group name.</td>
</tr>
<tr>
<td><strong>histchars</strong></td>
<td>A string value determining the characters used in history substitution. The first character of its value is used as the history substitution character, replacing the default character ! (exclamation point). The second character of its value replaces the character ^ (caret) in quick substitutions.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>histdup</td>
<td>Controls handling of duplicate entries in the history list. If set to all only unique history events are entered in the history list. If set to prev and the last history event is the same as the current command, then the current command is not entered in the history. If set to erase and the same event is found in the history list, that old event gets erased and the current one gets inserted. The prev and all options renumber history events so there are no gaps.</td>
</tr>
<tr>
<td>histfile</td>
<td>The default location in which history -S and history -L look for a history file. If unset, ~/.history is used. histfile is useful when sharing the same home directory between different machines, or when saving separate histories on different terminals. Because only ~/.tcshrc is normally sourced before ~/.history, histfile should be set in ~/.tcshrc instead of ~/.login. An example:</td>
</tr>
<tr>
<td></td>
<td>set histfile = ~/.history</td>
</tr>
<tr>
<td>histlit</td>
<td>If set, built-in and editor commands and the savehist mechanism use the literal (unexpanded) form of lines in the history list. See also the toggle-literal-history editor command.</td>
</tr>
<tr>
<td>history</td>
<td>The first word indicates the number of history events to save. The optional second word indicates the format in which history is printed; if not given, %h\t%T\t%R\n is used. The format sequences are described under prompt. (Note that %R has a variable meaning). Set to 100 by default.</td>
</tr>
<tr>
<td>home</td>
<td>Initialized to the home directory of the invoker. The file name expansion of ~ refers to this variable.</td>
</tr>
<tr>
<td>ignoreeof</td>
<td>If set to the empty string or 0 and the input device is a terminal, the end-of-file command (typically generated by the user by typing ^D on an empty line) causes the shell to print 'Use &quot;logout&quot; to leave tcsh.' instead of exiting. This prevents the shell from accidentally being killed. If set to a number n, the shell ignores n - 1 consecutive end-of-files and exits on the nth. If unset, 1 is used. That is, the shell exits on a single ^D.</td>
</tr>
<tr>
<td>implicitcd</td>
<td>If set, the shell treats a directory name that is typed as a command as though it were a request to change to that directory. If set to verbose, the change of directory is echoed to the standard output. This behavior is inhibited in non-interactive shell scripts, or for command strings with more than one word. Changing directory takes precedence over executing a like-named command, but it is done after alias substitutions. Tilde and variable expansions work as expected.</td>
</tr>
<tr>
<td>inputmode</td>
<td>If set to insert or overwrite, puts the editor into that input mode at the beginning of each line.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>listflags</td>
<td>If set to x, a or A, or any combination thereof (for example, xA), they are used as flags to <code>ls -F</code>, making it act like <code>ls -xF</code>, <code>ls -Fa</code>, <code>ls -FA</code> or a combination (for example, <code>ls -FxA</code>): <code>a</code> shows all files (even if they start with a <code>.</code>), <code>A</code> shows all files but <code>.</code> and '..', and <code>x</code> sorts across instead of down. If the second word of listflags is set, it is used as the path to <code>ls(1)</code>.</td>
</tr>
<tr>
<td>listjobs</td>
<td>If set, all jobs are listed when a job is suspended. If set to <code>long</code>, the listing is in long format.</td>
</tr>
<tr>
<td>listlinks</td>
<td>If set, the <code>ls -F</code> built-in command shows the type of file to which each symbolic link points. For an example of its use, see “<code>ls -F</code> built-in command for tcsh: List files” on page 727.</td>
</tr>
<tr>
<td>listmax</td>
<td>The maximum number of items that the <code>list-choices</code> editor command will list without asking first.</td>
</tr>
<tr>
<td>listmaxrows</td>
<td>The maximum number of rows of items that the list-choices editor command will list without asking first.</td>
</tr>
<tr>
<td>loginsh</td>
<td>Set by the shell if is a login shell. Setting or unsetting it within a shell has no effect. See also <code>shlvl</code>.</td>
</tr>
<tr>
<td>logout</td>
<td>Set by the shell to <code>normal</code> before a normal logout, automatic before an automatic logout, and <code>hangup</code> if the shell was killed by a <code>hangup</code> signal. See also the <code>autologout</code> shell variable.</td>
</tr>
<tr>
<td>mail</td>
<td>The names of the files or directories to check for incoming mail, separated by white space, and optionally preceded by a numeric word. Before each prompt, if 10 minutes have passed since the last check, the shell checks each file and says 'You have new mail.' (or, if mail contains multiple files, 'You have new mail in name.') if the <code>filesize</code> is greater than zero in size and has a modification time greater than its access time. If you are in a login shell, then no mail file is reported unless it has been modified after the time the shell has started up, in order to prevent redundant notifications. Most login programs will tell you whether or not you have mail when you log in. If a file specified in mail is a directory, the shell will count each file within that directory as a separate message, and will report 'You have n mails.' or 'You have n mails in name.' as appropriate. This functionality is provided primarily for those systems that store mail in this manner, such as the Andrew Mail System. If the first word of mail is numeric it is taken as a different mail checking interval, in seconds. Under rare circumstances, the shell might report 'You have mail.' instead of 'You have new mail.'</td>
</tr>
<tr>
<td>matchbeep</td>
<td>If set to never, completion never beeps. If set to nomatch, it beeps only when there is no match. If set to ambiguous, it beeps when there are multiple matches. If set to notunique, it beeps when there is one exact and other longer matches. If unset, ambiguous is used.</td>
</tr>
<tr>
<td>nobeep</td>
<td>Beeping is completely disabled.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>noclobber</td>
<td>Restrictions are placed on output redirection to ensure that files are not accidentally destroyed and that &gt;&gt; redirections refer to existing files, as described in Input or output.</td>
</tr>
<tr>
<td>noglob</td>
<td>File name substitution and directory stack substitution are inhibited. This option is most useful in shell scripts that do not deal with file names, or after a list of file names has been obtained and further expansions are not wanted.</td>
</tr>
<tr>
<td>nokanji</td>
<td>If set and the shell supports Kanji (see the version shell variable), it is disabled so that the meta key can be used.</td>
</tr>
<tr>
<td>nonomatch</td>
<td>A file name substitution or directory stack substitution that does not match any existing files is left untouched instead of causing an error. It is still an error for the substitution to be malformed, that is, echo [] still gives an error.</td>
</tr>
<tr>
<td>nostat</td>
<td>A list of directories (or glob-patterns that match directories; see File name substitution) that should not be stat(2)ed during a completion operation. This is typically used to exclude directories which take too much time to stat(2), for example /afs.</td>
</tr>
<tr>
<td>notify</td>
<td>The shell announces job completions asynchronously. The default is to present job completions just before printing a prompt.</td>
</tr>
<tr>
<td>owd</td>
<td>The old working directory, equivalent to the - (hyphen) used by cd and pushd. See also the cwd and dirstack shell variables.</td>
</tr>
<tr>
<td>path</td>
<td>A list of directories in which to look for executable commands. A null word specifies the current directory. If there is no path variable then only full path names will be executed. path is set by the shell at startup from the PATH environment variable or, if PATH does not exist, to a system-dependent default something like (/usr/local/bin /usr/bsd/bin /usr/bin .). The shell might put '.' first or last in path or omit it entirely depending on how it was compiled; see the version shell variable. A shell which is given neither the -c nor the -t option hashes the contents of the directories in path after reading ~/.tcshrc and each time path is reset. If you add a new command to a directory in path while the shell is active, you might need to do a rehash for the shell to find it.</td>
</tr>
<tr>
<td>printexit-value</td>
<td>If set and an interactive program exits with a nonzero status, the shell prints 'Exit status'.</td>
</tr>
<tr>
<td>prompt2</td>
<td>The string with which to prompt in while and foreach loops and after lines ending in \ (backslash). The same format sequences can be used as in prompt (note the variable meaning of %R). Set by default to %R? in interactive shells.</td>
</tr>
<tr>
<td>prompt3</td>
<td>The string with which to prompt when confirming automatic spelling correction. The same format sequences can be used as in prompt (note the variable meaning of %R). Set by default to CORRECT&gt;%R (y</td>
</tr>
<tr>
<td>promptchars</td>
<td>If set to a two-character string, the %# formatting sequence in the prompt shell variable is replaced with the first character for normal users and the second character for the superuser.</td>
</tr>
<tr>
<td>pushdtohome</td>
<td>pushd without arguments does pushd ^, like cd.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pushdsilent</td>
<td>pushd and popd do not print the directory stack.</td>
</tr>
<tr>
<td>recexact</td>
<td>Completion completes on an exact match even if a longer match is possible.</td>
</tr>
<tr>
<td>recognize_only_executables</td>
<td>Command listing displays only files in the path that are executable.</td>
</tr>
<tr>
<td>rmstar</td>
<td>The user is prompted before rm * is executed.</td>
</tr>
<tr>
<td>rprompt</td>
<td>The string to print on the right side of the screen (after the command input) when the prompt is being displayed on the left. It recognizes the same formatting characters as prompt. It will automatically disappear and reappear as necessary, to ensure that command input isn’t obscured, and will only appear if the prompt, command input, and itself will fit together on the first line. If edit isn’t set, then rprompt will be printed after the prompt and before the command input.</td>
</tr>
<tr>
<td>savedirs</td>
<td>The shell does dirs -S before exiting.</td>
</tr>
<tr>
<td>savehist</td>
<td>The shell does history -S before exiting. If the first word is set to a number, at most that many lines are saved. (The number must be less than or equal to history.) If the second word is set to merge, the history list is merged with the existing history file instead of replacing it (if there is one) and sorted by time stamp and the most recent events are retained. An example: set savehist = ( 15 merge )</td>
</tr>
<tr>
<td>sched</td>
<td>The format in which the sched built-in command prints scheduled events. If not given, %h%T%R\n is used. The format sequences are described under prompt; note the variable meaning of %R.</td>
</tr>
<tr>
<td>shell</td>
<td>The file in which the shell resides. This is used in forking shells to interpret files which have execute bits set, but which are not executable by the system (see “Built-in and non-built-in command execution” on page 687. Initialized to the (system-dependent) home of the shell.</td>
</tr>
<tr>
<td>shlvl</td>
<td>The number of nested shells. Reset to 1 in login shells. See also loginsh.</td>
</tr>
<tr>
<td>status</td>
<td>The status returned by the last command. If it terminated abnormally, then 0200 is added to the status. tcsh built-in commands which fail return exit status 1, all other built-in commands return status 0.</td>
</tr>
<tr>
<td>tcsh</td>
<td>The version number of the shell in the format R.VV.PP, where R is the major release number, VV the current version and PP the patch level.</td>
</tr>
<tr>
<td>term</td>
<td>The terminal type. Typically set in ~/.login.</td>
</tr>
<tr>
<td>tperiod</td>
<td>The period, in minutes, between executions of the periodic special alias.</td>
</tr>
<tr>
<td>tty</td>
<td>The name of the tty, or empty if not attached to one.</td>
</tr>
<tr>
<td>uid</td>
<td>The user’s login name.</td>
</tr>
<tr>
<td>user</td>
<td>The user’s login name.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>verbose</td>
<td>The words of each command are printed after history substitution (if any). Set by the -v command line option.</td>
</tr>
</tbody>
</table>
### Table 33: tcsh built-in shell variables (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **version** | The version ID stamp. It contains the shell's version number (see `tcsh`), origin, release date, vendor, operating system and machine (see `VENDOR`, `OSTYPE`, and `MACHTYPE` environment variables) and a comma-separated list of options which were set at compile time. Options which are set by default in the distribution are noted.  
  8b  
  The shell is eight bit clean; default.  
  7b  
  The shell is not eight bit clean.  
  nls  
  The system’s multicultural support is used; default for systems with multicultural support.  
  If  
  Login shells execute `/etc/csh.login` before instead of after `/etc/csh.cshrc` and `~/.login` before instead of after `~/.tcshrc` and `~/.history`.  
  dl  
  ‘.’ is put last in path for security; default.  
  nd  
  ‘.’ is omitted from path for security.  
  vi  
  vi-style editing is the default instead of emacs.  
  dtr  
  Login shells drop DTR when exiting.  
  bye  
  `bye` is a synonym for logout and `log` is an alternative name for `watchlog`.  
  al  
  `autologout` is enabled; default.  
  kan  
  Kanji is used and the ISO character set is ignored, unless the nokanji shell variable is set.  
  sm  
  The system’s malloc is used.  
  hb  
  The `#!<program> <args>` convention is emulated when executing shell scripts.  
  ng  
  The `newgrp` built-in is available.  
  rh  
  The shell attempts to set the `REMOTEHOST` environment variable.  
  afs  
  The shell verifies your password or password phrase with the Kerberos server if local authentication fails. The `afsuser` shell variable or the `AFSUSER` environment variable will override your local username.  
| An administrator can enter additional strings to indicate differences in the local version. |
### Table 33: tcsh built-in shell variables (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>visiblebell</strong></td>
<td>If set, a screen flash is used instead of the audible bell. See <strong>nobeep</strong>. (Currently not implemented.)</td>
</tr>
<tr>
<td><strong>watch</strong></td>
<td>A list of user/terminal pairs to watch for logins and logouts. If either the user is <em>any</em> all terminals are watched for the given user and vice versa. Setting watch to <em>(any any)</em> watches all users and terminals. For example,</td>
</tr>
</tbody>
</table>

```plaintext
set watch = (george ttyd 1 any console $user any)
```

reports activity of the user george on ttyd1, any user on the console, and oneself (or a trespasser) on any terminal.

Logins and logouts are checked every 10 minutes by default, but the first word of watch can be set to a number to check every so many minutes. For example,

```plaintext
set watch = (1 any any)
```

reports any logins or logouts once every minute. For the impatient, the **log** built-in command triggers a watch report at any time. All current logins are reported (as with the **log** built-in) when watch is first set. The **who** shell variable controls the format of watch reports.

<table>
<thead>
<tr>
<th><strong>who</strong></th>
<th>The format string for watch messages. The following sequences are replaced by the given information:</th>
</tr>
</thead>
<tbody>
<tr>
<td>%n</td>
<td>The name of the user who logged in/out.</td>
</tr>
<tr>
<td>%a</td>
<td>The observed action, such as 'logged on', 'logged off', or 'replaced olduser on'.</td>
</tr>
<tr>
<td>%l</td>
<td>The terminal (tty) on which the user logged in/out.</td>
</tr>
<tr>
<td>%M</td>
<td>The full host name of the remote host, or 'local' if the login/logout was from the local host.</td>
</tr>
<tr>
<td>%m</td>
<td>The host name of the remote host up to the first '.' (period). The full name is printed if it is an IP address or an X Window System display.</td>
</tr>
</tbody>
</table>

%M and %m are available only on systems that store the remote host name in `/etc/utmp`. If unset, %n has %a %l from %m. is used, or %n has %a %l. on systems which do not store the remote hostname.

| **wordchars** | A list of non-alphanumeric characters to be considered part of a word by the forward-word, backward word, etc. editor commands. If unset, *?_-[]~= is used. |

---

**tcsh** shell variables not described in the Table 33 on page 698 are described as follows:

**prompt**

The string which is printed before reading each command from the terminal. prompt can include any of the following formatting sequences, which are replaced by the given information:
%/
The current working directory.

%~
The current working directory, but with one's home directory represented by ~ and other users' home directories represented by ~user as per file name substitution. ~user substitution happens only if the shell has already used ~user in a path name in the current session.

%c[[0]n], %.[[0]n]
The trailing component of the current working directory, or n trailing components if a digit n is given. If n begins with 0, the number of skipped components precede the trailing components in the format /trailing. If the ellipsis shell variable is set, skipped components are represented by an ellipsis so the whole becomes ...trailing. ~ substitution is done as in %~~, but the ~ component is ignored when counting trailing components.

%c
Like %c, but without ^ substitution.

%h, %!, !
The current history event number.

%M
The full host name.

%m
The host name up to the first '.' (period).

%S (%s)
Start (stop) standout mode.

%B (%b)
Start (stop) boldfacing mode.

%U (%u)
Start (stop) underline mode.

%t, %@
The time of day in 12-hour AM/PM format.

%T
Like %t, but in 24-hour format (but see the ampm shell variable).

%p
The precise time of day in 12-hour AM/PM format, with seconds.

%P
Like %p, but in 24-hour format (but see the ampm shell variable).

\c
\c is parsed as in bindkey.

^c
\c is parsed as in bindkey.

%%
A single %.

%n
The user name.

%d
The weekday in Day format.

%D
The day in dd format.

%w
The month in Mon format.

%W
The month in mm format.
%y
  The year in yy format.

%Y
  The year in yyyy format.

%l
  The tcsh shell’s tty.

%L
  Clears from the end of the prompt to end of the display or the end of the line.

%$
  Expands the shell or environment variable name immediately after the $.

%#
  > (or the first character of the promptchars shell variable) for normal users, # (or the second character of promptchars) for the superuser.

{%string%}
  Includes string as a literal escape sequence. It should be used only to change terminal attributes and should not move the cursor location. This cannot be the last sequence in prompt.

%?
  The return code of the command executed just before the prompt.

%R
  In prompt2, the status of the parser. In prompt3, the corrected string. In history, the history string.

The bold, standout and underline sequences are often used to distinguish a superuser shell. For example,

> set prompt = "%m [%h] %B[%@%b [%/] you rang?"

Set by default to %# in interactive shells.

cymlinks
Can be set to several different values to control symbolic link resolution:

- If set to chase, whenever the current directory changes to a directory containing a symbolic link, it is expanded to the real name of the directory to which the link points. This does not work for the user’s home directory.

- If set to ignore, the shell tries to construct a current directory relative to the current directory before the link was crossed. This means that cd'ing through a symbolic link and then cd.. 'ing returns one to the original directory. This only affects built-in commands and file name completion.

- If set to expand, the shell tries to fix symbolic links by actually expanding arguments which look like path names. This affects any command, not just built-ins. Unfortunately, this does not work for hard-to-recognize file names, such as those embedded in command options. Expansion can be prevented by quoting. While this setting is typically the most convenient, it is sometimes misleading and sometimes confusing when it fails to recognize an argument which should be expanded. A compromise is to use ignore and use the editor command normalize-path (bound by default to ^X-n) when necessary.

Some examples are in order. First, set up some play directories:

> cd /tmp
> mkdir from from/src to
> ln -s from/src to/dist

Here’s the behavior with symbolic links unset,

> cd /tmp/to/dist; echo $cwd
/tmp/to/dist
> cd ..; echo $cwd
/tmp/from

Shell command descriptions  709
here's the behavior with symbolic links set to chase,

```
> cd /tmp/to/dst; echo $cwd
/tmp/from/src
> cd ..; echo $cwd
/tmp/from
```

displays `/tmp/from/src` with symbolic links set to chase.

And here's the behavior with symbolic links set to ignore,

```
> cd /tmp/to/dist; echo $cwd
/tmp/to/dist
> cd ..; echo $cwd
/tmp/to
```

displays `/tmp/to/dist` with symbolic links set to ignore.

And here's the behavior with symbolic links set to expand.

```
> cd /tmp/to/dist; echo $cwd
/tmp/to/dist
> cd ..; echo $cwd
/tmp/to
> cd /tmp/to/dist; echo $cwd
/tmp/to/dist
> cd "."; echo $cwd
/tmp/from
> /bin/echo ..
/tmp/to
> /bin/echo ".."
.. 
```

`expand` expansion works just like `ignore` for built-ins like `cd`, is prevented by quoting, and happens before file names are passed to non-built-in commands.

### time

If set to a number, then the `time` built-in command executes automatically after each command that takes more than that many CPU seconds. If there is a second word, it is used as a format string for the output of the time built-in. The following sequences can be used in the format string:

- `%U` - The time the process that is spent in user mode in CPU seconds.
- `%S` - The time the process that is spent in kernel mode in CPU seconds.
- `%E` - The elapsed (wall clock) time in seconds.
- `%P` - The CPU percentage that is computed as (%U + %S) / %E.
- `%W` - The number of times the process was swapped.
- `%X` - The average amount in (shared) text space that is used in Kbytes.
- `%D` - The average amount in (unshared) data/stack space that is used in Kbytes.
- `%K` - The total space used (%X + %D) in Kbytes.
- `%M` - The maximum memory the process had in use at any time in Kbytes.
- `%F` - The number of major page faults (page that is needed to be brought from disk).
- `%R` - The number of minor page faults.
- `%I` - The number of input operations.
The number of output operations.

The number of socket messages received.

The number of socket messages sent.

The number of signals received.

The number of voluntary context switches (waits).

The number of involuntary context switches.

Only the first four sequences are supported on systems without BSD resource limit functions. The default time format is

```
Uu   %Ss %E %P %X+%Dk %I+%OiOio   %Fpf+%Ww
```

for systems that support resource usage reporting.

Table 34 on page 711 contains a list of tcsh environment variables.

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMNS</td>
<td>A list of directories in which cd should search for subdirectories if they aren't found in the current directory.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>Used by X Window System. If set, the shell does not set AUTOLOGOUT.</td>
</tr>
<tr>
<td>EDITOR</td>
<td>The path name to a default editor. See also the VISUAL environment variable and the run-fg-editor editor command.</td>
</tr>
<tr>
<td>GROUP</td>
<td>Equivalent to the group shell variable.</td>
</tr>
<tr>
<td>HOME</td>
<td>Equivalent to the HOME shell variable.</td>
</tr>
<tr>
<td>HOST</td>
<td>Initialized to the name of the machine of the machine on which the shell is running, as determined by the gethostname system call.</td>
</tr>
<tr>
<td>HOSTTYPE</td>
<td>Initialized to the type of the machine on which the shell is running, as determined at compile time. This variable is obsolete and will be removed in a future version.</td>
</tr>
<tr>
<td>HPATH</td>
<td>A colon-separated list of directories in which the run-help editor command looks for a command documentation.</td>
</tr>
<tr>
<td>LANG</td>
<td>Gives the preferred character environment. See National language system report.</td>
</tr>
<tr>
<td>LC_CTYPE</td>
<td>If set, only CTYPE character handling is changed. See National language system report.</td>
</tr>
<tr>
<td>LINES</td>
<td>The number of lines in the terminal. See “Managing terminals” on page 695.</td>
</tr>
<tr>
<td>MACHTYPE</td>
<td>The machine type (microprocessor class or machine model), as determined at compile time.</td>
</tr>
</tbody>
</table>
### Table 34: tcsh environment variables (continued)

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOREBIND</strong></td>
<td>If set, printable characters are not rebound to SELF-INSERT-COMMAND. After a user sets NOREBIND, a new shell must be started. See National language system report.</td>
</tr>
<tr>
<td><strong>OSTYPE</strong></td>
<td>The operating system, as determined at compile time.</td>
</tr>
<tr>
<td><strong>PATH</strong></td>
<td>A colon-separated list of directories in which to look for executables. Equivalent to the path shell variable, but in a different format.</td>
</tr>
<tr>
<td><strong>PWD</strong></td>
<td>Equivalent to the cwd shell variable, but not synchronized to it; updated only after an actual directory change.</td>
</tr>
<tr>
<td><strong>REMOTE-HOST</strong></td>
<td>The host from which the user has logged in remotely, if this is the case and the shell is able to determine it. (The z/OS tcsh shell is not currently compiled with REMOTEHOST defined; see the version shell variable.)</td>
</tr>
<tr>
<td><strong>SHLVL</strong></td>
<td>Equivalent to the shlvl shell variable.</td>
</tr>
<tr>
<td><strong>TERM</strong></td>
<td>Equivalent to the term shell variable.</td>
</tr>
<tr>
<td><strong>USER</strong></td>
<td>Equivalent to the user shell variable.</td>
</tr>
<tr>
<td><strong>VENDOR</strong></td>
<td>The vendor, as determined at compile time.</td>
</tr>
<tr>
<td><strong>VISUAL</strong></td>
<td>The path name to a default full-screen editor. See the editor environment variable and the run-fg-editor editor command.</td>
</tr>
</tbody>
</table>

### Using tcsh shell variables to control automatic conversion

When the tcsh shell is redirecting stdin, stdout, or stderr, it will default to no automatic conversion of tagged files, and no tagging of files created by the redirection. The following tcsh shell variables will override this behavior:

- **_TAG_REDIR_IN=TXT**
  - Redirected stdin will override the file's TXTFLAG, treating it as if it were tagged as:
    - TXTFLAG = ON, CCSID = existing file tag CCSID
  - This has no effect if CCSID = 0.

- **_TAG_REDIR_IN=BIN**
  - Redirected stdin will override the file's TXTFLAG, treating it as if it were tagged as:
    - TXTFLAG = OFF, CCSID = existing file tag CCSID
  - This effectively disables automatic conversion.

- **_TAG_REDIR_OUT=TXT**
  - Redirected stdout is tagged as:
    - TXTFLAG = ON, CCSID = program CCSID at the time of the first write (if not already tagged)

- **_TAG_REDIR_OUT=BIN**
  - Redirected stdout is tagged as:
    - TXTFLAG = OFF, CCSID = program CCSID at the time of the first write (if not already tagged)

- **_TAG_REDIR_ERR=TXT**
  - Redirected stderr is tagged as:
TXTFLAG = ON, CCSID = program CCSID at the time of the first write (if not already tagged)

_TAG_REDIR_ERR=BIN
  Redirected stderr is tagged as:
  TXTFLAG = OFF, CCSID = program CCSID at the time of the first write (if not already tagged)

The automatic conversion shell variable can be specified for one command, or for multiple commands within a tcsh shell session or shell script. If the variable is set in a user's .tcshrc file, then it will affect child shells, that is, nested shell scripts.

**Note:** Because the standard tcsh shell execution performs redirection before variable assignment, the syntax for specifying the shell variable for one command is set `var=value`. For example:

```
(set _TAG_REDIR_OUT=TXT; command >file)
```

You can also use these shell variables for commands in a pipeline. For example, they can be used to tag the standard output of each command that is writing to a pipeline or to tag the standard input of each command that is reading from a pipeline.

tcsh files

/etc/csh.cshrc
  Read first by every shell.

/etc/csh.login
  Read by login shells after /etc/csh.cshrc.

~/.tcshrc
  Read by every shell after /etc/csh.cshrc or its equivalent.

~/.history
  Read by login shells after ~/.tcshrc if `savehist` is set. See also histfile.

~/.login
  The shell reads ~/.login after ~/.tcshrc and ~/.history. See the version shell variable.

~/.cshdirs
  Read by login shells after ~/.login if `savedirs` is set. See also dirsfile.

~/.logout
  Read by login shells at logout.

/bin/sh
  Used to interpret shell scripts not starting with a #.

/tmp/sh*
  Temporary file for < <.

tcsh shell: problems and limitations

Some limitations of the tcsh shell are:

- Words can be no longer than 1024 characters.
- The system limits argument lists to 10240 characters.
- The number of arguments to a command which involves file name expansion is limited to 1/6th the number of characters allowed in an argument list.
- Command substitutions can substitute no more characters than are allowed in an argument list.
- To detect looping, the shell restricts the number of alias substitutions on a single line to 20.
When a suspended command is restarted, the tcsh shell prints the directory that it started in if this is different from the current directory. This can be misleading (that is, wrong) as the job might have changed directories internally.

Shell built-in functions are not stoppable/restartable. Command sequences of the form 'a ; b ; c' are also not handled gracefully when stopping is attempted. If you suspend 'b', the tcsh shell will then immediately execute 'c'. This is especially noticeable if this expansion results from an alias. It suffices to place the sequence of commands in ()'s to force it to a subshell, for example, ( a ; b ; c ).

Control over tty output after processes are started is primitive. In a virtual terminal interface, more options are available for output control.

Alias substitution is most often used to clumsily simulate shell procedures; shell procedures should be provided instead of aliases.

Commands within loops are not placed in the history list. Control structures should be parsed instead of being recognized as built-in commands. This would allow control commands to be placed anywhere, to be combined with |, and to be used with & and ; (semicolon) metasyntax.

**foreach** does not ignore here-documents when looking for its end.

It should be possible to use the : (colon) modifiers on the output of command substitutions.

The screen update for lines that are longer than the screen width is very poor if the terminal cannot move up the cursor up (terminal type 'dumb').

It is not necessary for HPATH and NOREBIND to be environment variables.

Glob-patterns that do not use '?', '*' or '[]' or which use '{'} or '~' are not negated correctly.

The single-command form of **if** does output redirection even if the expression is false and the command is not executed.

**ls -F** includes file identification characters when file names are being sorted and does not handle control characters in file names well. It cannot be interrupted.

The **visiblebell** shell variable is currently not implemented.

In file name and programmed completion, the C completion rule word list type does not correctly select completion from the given directory.

There are three locales (code pages) which the tcsh shell will not correctly support: IBM-1388 (Chinese), IBM-933 (Korean) and IBM-937 (Traditional Chinese).

If you want to help maintain and test tcsh, send mail to listserv@mx.gw.com with the text 'subscribe tcsh'.

**Related information**

: (colon), @ (at), alias, bg, break, cd, continue, echo, eval, exec, exit, fg, history, jobs, kill, newgrp, nice, nohup, printenv, set, shift, stop, suspend, time, umask, unalias, unset, wait

---

**@ (at) built-in command for tcsh: Print the value of tcsh shell variables**

**Format**

- @
- @ name = expr
- @ name[index] = expr
- @ name++ + |– –
- @ name[index]++ + |– –
Description
@ (at) in the tcsh shell prints the value of tcsh shell variables.

Options
@ in the tcsh shell supports the following options:

```
name = expr
```
Assigns the value of expr to name.

```
name[index] = expr
```
Assigns the value of expr to the index'th component of name. Both name and its index'th component must exist.

For both name = expr and name[index] = expr, expr might contain the operators such as *, and +, as in C. If expr contains <, >, &, or " then at least part of expr must be placed within (). The syntax of expr has nothing to do with that described under Expressions.

expr must evaluate to a numeric expression. Therefore, use set instead of @ to assign array variables.

```
name++ | --
```
Increments (++) or decrements (--) name.

```
name[index]++ | --
```
Increments (++) or decrements (--) name's index'th component.

Usage notes
1. The space between @ and name is required.
2. The spaces between name and = and between = and expr are optional.
3. Components of expr must be separated by spaces.

Related information
tcsh

% (percent) built-in command for tcsh: Move jobs to the foreground or background

Format

% [job] [&]

Description
%
, is a synonym for the fg built-in command.

• % (percent) without arguments will bring the current job to the foreground.
• % specified with a job number attempts to bring that particular job to the foreground.
• % job & will move the specified job to the background. This syntax works the same as the bg built-in command. If no job is specified, the current job is moved to the background.

Note: Jobs that are current will have a + next to the status column in jobs command output. See “Jobs” on page 693.

Related information
jobs, tcsh
alloc built-in command for tcsh: Show the amount of dynamic memory acquired

**Format**

```
alloc argument
```

**Description**

`alloc` shows the amount of dynamic memory acquired, broken down into used and free memory. `alloc` used with an argument, shows the number of free and used blocks in each size category. The categories start at size 8 and double at each step.

**Note:** `alloc` is supported, but the output is not meaningful on z/OS.

**Related information**

`tcsh`

bindkey built-in command for tcsh: List all bound keys

**Format**

```
bindkey [-l|-d|-e|-v|-u] 
bindkey [-a] [-b] [-k] [-i] [-] [-] key 
bindkey [-a] [-b] [-k] [-c|-s] [-] [-] key command
```

**Description**

`bindkey` specified alone (without options, `key`, or `key command`) lists all bound keys and the editor command to which each is bound.

`bindkey` specified with `key` (with or without options) lists the editor command to which key is bound.

`bindkey` specified with `key command` (with or without options) binds the editor command to `key`.

**Options**

- `-l`
  
  Lists all commands and a short description of each.

- `-d`
  
  Binds all keys to the standard bindings for the default editor.

- `-e`
  
  Binds all keys to the standard GNU Emacs-like bindings.

- `-v`
  
  Binds all keys to the standard vi-like bindings.

- `-a`
  
  Lists or changes key-bindings in the alternative key map. This is the key map used in vi command mode.

- `-b`
  
  `key` is interpreted as a control character written `^character` (`^A`) or `C-character` (`C-A`), a meta character written `M-character` (`M-A`), or an extended prefix key written `X-character` (`X-A`).

- `-k`
  
  `key` is interpreted as a symbolic arrow key name, which can be one of 'down', 'up', 'left' or 'right'.

716 z/OS: UNIX System Services Command Reference
-r
   Removes key's binding. Be careful: bindkey -r does not bind key to self-insert-command, it unbinds key completely.

-c
   command is interpreted as a built-in or external command instead of an editor command.

-s
   command is taken as a literal string and treated as terminal input when key is typed. Bound keys in command are themselves reinterpreted, and this continues for ten levels of interpretation.

--
   Forces a break from option processing, so the next word is taken as key even if it begins with `-'.

Usage notes

1. key can be a single character or a string. If a command is bound to a string, the first character of the string is bound to sequence-lead-in and the entire string is bound to the command.

2. Control characters in key can be literal (they can be typed by preceding them with the editor command quoted-insert, normally bound to ^V) or written caret-character style, for example, ^A. Delete is written ^? (caret-question mark). key and command can contain backslashed escape sequences (in the style of System V echo) as follows:

   \a
   Bell

   \b
   Backspace

   \e
   Escape

   \f
   Form feed

   \n
   Newline

   \r
   Carriage return

   \t
   Horizontal tab

   \v
   Vertical tab

   \nnnn
   The EBCDIC character corresponding to the octal number nnn

   The \ character nullifies the special meaning of the following characters, notably \ and ^.

Related information

tcsh

builtins built-in command for tcsh: Prints the names of all built-in commands

Format

builtins

Description

builtins prints the names of all built-in commands.
bye built-in command for tcsh: Terminate the login shell

Format
bye

Description
A synonym for the logout built-in command. (See the version shell variable.)

Related information
logout

chdir built-in shell command for tcsh: Change the working directory

Format
chdir

Description
A synonym for the cd built-in command.

Related information
cd, tcsh

complete built-in command for tcsh: List completions

Format
complete [command [word/pattern/list[:select]/[|suffix]/]]...

Description
complete, without arguments, lists all completions. With command, complete lists completions for command. With command and word etc., complete defines completions.

Arguments
command
    command can be a full command name or a glob-pattern. See File name substitution. It can begin with – to indicate that completion should be used only when command is ambiguous.

word
    word specifies which word relative to the current word is to be completed, and can be one of the following:

    c
        Current-word completion. pattern is a glob-pattern which must match the beginning of the current word on the command line. pattern is ignored when completing the current word.

    C
        Like c, but includes pattern when completing the current word.
n
Next-word completion. pattern is a glob-pattern which must match the beginning of the previous word on the command line.

N
Like n, but must match the beginning of the word two before the current word.

p
Position-dependent completion. pattern is a numeric range, with the same syntax used to index shell variables, which must include the current word.

list
The list of possible completions, which can be one of the following:

a
Aliases

b
Bindings (editor commands)

d
Directories

D
Directories which begin with the supplied path prefix

e
Environment variables

f
File names

F
File names which begin with the supplied path prefix

g
Group names

j
Jobs

l
Limits

n
Nothing

s
Shell variables

S
Signals

t
Plain (text) files

T
Plain (text) files which begin with the supplied path prefix

v
Any variables

u
User names

x
Like n, but prints select when list-choices is used

X
Completions

$var
Words from the variable var
(...)
Words from the given list
...
Words from the output of command

**select**

*select* is an optional glob-pattern. If given, only words from *list* which match *select* are considered and the *fignore* shell variable is ignored. The last three types of completion might not have a *select* pattern, and *x* uses *select* as an explanatory message when the *list-choices* editor command is used.

**suffix**

*suffix* is a single character to be appended to a successful completion. If null, no character is appended. If omitted (in which case the fourth delimiter can also be omitted), a slash is appended to directories and a space to other words.

**Examples**

1. Some commands take only directories as arguments, so there is no point in completing plain files. For example:

   ```
   > complete cd 'p/1/d/
   ```

   completes only the first word following *cd* (*p/1*) with a directory.

2. *p*-type completion can be used to narrow down command completion. For example:

   ```
   > co[0]
   complete compress
   > complete co* 'p/0/(compress)/'
   > co[0]
   > compress
   ```

   This completion completes commands (words in position *0*, *p/0*) which begin with *co* (thus matching *co* *) to *compress* (the only word in the list). The leading - indicates that this completion is to be used only with ambiguous commands.

3. This is an example of *n*-type completion. Any word following *find* and immediately following `-user` is completed from the list of users.

   ```
   > complete find 'n/-user/u/
   ```

4. This demonstrates *c*-type completion. Any word following *cc* and beginning with `-I` is completed as a directory. `-I` is not taken as part of the directory because we used lowercase *c*.

   ```
   > complete cc 'c/-I/d/
   ```

5. Different *lists* are useful with different commands:

   ```
   > complete alias 'p/1/a/
   > complete man 'p/*/c/
   > complete set 'p/1/s/
   > complete true 'p/1/x:Truth has no options./
   ```

   These complete words following *alias* with aliases, *man* with commands, and *set* with shell variables. *true* doesn't have any options, so *x* does nothing when completion is attempted and prints 'Truth has no options.' when completion choices are listed.

   The *man* example, and several other examples that follow, could just as well have used *c/* or *n/* as *p/*.

6. Words can be completed from a variable evaluated at completion time,

   ```
   > complete ftp 'p/1/$hostnames/
   > set hostnames = ('rtfm.mit.edu tesla.ee.cornell.edu)
   > ftp [0]
   rtfm.mit.edu tesla.ee.cornell.edu
   ```
or from a command run at completion time:

```
> complete kill 'p/*/ps | awk \{print \$1\}'
> kill -9 [D]
23113 23377 23380 23406 23429 23529 23530 PID
```

The `complete` command does not itself quote its arguments, so the braces, space and $ in `{print $1}` must be quoted explicitly.

7. One command can have multiple completions:

```
> complete dbx 'p/2/(core)/' 'p/*/c/
```

This example completes the second argument to `dbx` with the word `core` and all other arguments with commands. The positional completion is specified before the next-word completion. Since completions are evaluated from left to right, if the next-word completion were specified first it would always match and the positional completion would never be executed. This is a common mistake when defining a completion.

8. The `select` pattern is useful when a command takes only files with particular forms as arguments. For example,

```
> complete cc 'p/*/f:.*.[cao]/'
```

completes `cc` arguments only to files ending in `.c`, `.a`, or `.o`. `select` can also exclude files, using negation of a glob-pattern as described under File name substitution.

9. One might use

```
> complete rm 'p/*/f:.*.{c,h,cc,C,tex,1,man,l,y}/'
```

to exclude precious source code from `rm` completion. Of course, one could still type excluded names manually or override the completion mechanism using the `complete-word-raw` or `list-choices-raw` editor command.

10. The `D`, `F` and `T` lists are like `d`, `f` and `t` respectively, but they use the `select` argument in a different way: to restrict completion to files beginning with a particular path prefix. For example, the Elm mail program uses `=` as an abbreviation for one's mail directory. One might use

```
> complete elm c@=@F:$HOME/Mail/@
```

to complete `elm -f =` as if it were `elm -f ~/Mail/`. We used `@` instead of `/` to avoid confusion with the `select` argument, and we used `$HOME` instead of `~` because home directory substitution only works at the beginning of a word.

11. `suffix` is used to add a nonstandard suffix (not space or `/` for directories) to completed words. For example,

```
> complete finger 'c/*@/$hostnames/' 'p/1/u/@'
```

completes arguments to `finger` from the list of users, appends an @, and then completes after the @ from the `hostnames` variable. Note the order in which the completions are specified.

12. A more complex example:

```
complete find \n 'n/-name/f/' 'n/-newer/f/' 'n/-{,n}cpio/f/' \n 'n/-exec/c/' 'n/-ok/c/' 'n/-user/u/' \n 'n/-group/g/' 'n/-fstype/(nfs 4.2)/' \n 'n/-type/(b c d f l p s)/' \n 'c/-/(name newer cpio ncpio exec ok user \n group fstype type atime ctime depth inum \n ls mtime nogroup nouser perm print prune \n```

Shell command descriptions 721
complete is an experimental feature, and the syntax might change in future versions of the shell. See also the uncomplete built-in command.

Related information

tcsh, uncomplete

dirs built-in command for tcsh: Print the directory stack

Format

dirs [-l] [-n] [-v]
dirs -S[-L] [file name]
dirs -c

Description

dirs used alone prints the directory stack in the following format: The top of the stack is at the left and the first directory in the stack is the current directory. For example:

```
> cd <<<<<< # Change to home dir
> pushd /bin << # Change dir to /bin and add /bin to dir stack
/bin ~
> pushd /tmp << # Change dir to /tmp and add /tmp to dir stack
/tmp /bin ~
> dirs <<<<<< # Display current dir stack
/tmp /bin ~
> dirs -l <<<<< # Display in expanded (long) format
/tmp /bin /u/erinf
> dirs -v <<<<< # Display in verbose format
0 /tmp
1 /bin
2 ~
> popd <<<<<< # Change dir back to /bin and remove /tmp from dir stack
/bin ~
> pwd
/bin
>>> Note: dir=directory
```

Options

-\l
   Output is expanded explicitly to home or the path name of the home directory for the user.

-\n
   Entries are wrapped before they reach the edge of the screen.

-\v
   Entries are printed one per line, preceded by their stack positions.
   If more than one of -\n or -\v is given, -\v takes precedence.

-\S
   Saves the directory stack to file name as a series of cd and pushd commands.
The tcsh shell sources file name, which is presumably a directory stack file saved by the -S option or the savedirs mechanism. In either case, dirsfile is used if file name is not given and ~/.cshdirs is used if dirsfile is unset.

Login shells do the equivalent of dirs -L on startup and, if savedirs is set, you should issue dirs -S before exiting. Because only ~/.tcshrc is normally sourced before ~/.cshdirs, dirsfile should be set in ~/.tcshrc instead of ~/.login.

--c
Clear the directory stack.

Related information
tcsh

echotc built-in command for tcsh: Exercise the terminal capabilities in args

Format

```
echotc [-sv]arg ...
```

Description
echotc takes advantage of the terminal capabilities in args. For example, echotc cm 3 10 sends it to column 3 and row 10.

If arg is baud, cols, lines, meta or tabs, echotc prints the value of that capability (either yes or no, which indicates that the terminal does or does not have that capability). You might use this to make the output from a shell script less verbose on slow terminals, or limit command output to the number of lines on the screen:

```
> set history=`echotc lines`
> @ history--
```

Termcap strings might contain wild cards which will not echo correctly. Use double quotation marks when setting a shell variable to a terminal capability string, as in the following example which places the date in the status line:

```
> set standout=`echotc sò`
> set end_standout=`echotc sè`
> echo -n "$standout"; date; echo -n "$end_standout"
Mon Oct 25 10:06:48 EDT 1999
```

Note: The date, as indicated, is printed out in standard output.
The infocmp command can be used to print the current terminal description in termcap format (instead of terminfo format).

Options

- -s
Nonexistent capabilities return the empty string instead of causing an error.

- v
Messages are verbose.

Related information
tcsh
filetest built-in command for tcsh: Apply the op file inquiry operator to a file

**Format**

```bash
filetest -op file -
```

**Description**

filetest applies op (which is a file inquiry operator) to each file and returns the results as a space-separated list. For more information about file inquiry operators, see [File inquiry operators](#).

**Examples**

1. To use the filetest command to retrieve the security label:

   ```bash
   > filetest -m myfile
   SYSLOW
   ```

2. To test for a specific security label using an if statement:

   ```bash
   if ( -m myfile == "SYSLOW" ) then
     echo "myfile has seclabel of SYSLOW"
   endif
   ```

**Related information**

tcsh

glob built-in command for tcsh: Write each word to standard output

**Format**

```bash
glob wordlist
```

**Description**

glob is like echo, but \ (backslash) escapes are not recognized and words are delimited by null characters in the output. glob is useful for programs that want to use the shell to file name expand a list of words.

**Related information**

echo, tcsh

hashstat built-in command for tcsh: Print a statistic line on hash table effectiveness

**Format**

```bash
hashstat
```

**Description**

hashstat prints a statistics line indicating how effective the internal hash table has been at locating commands (and avoiding exec's). An exec is attempted for each component of the path where the hash function indicates a possible hit, and in each component which does not begin with a / (forward slash).

z/OS systems have a vfork() command, however, tcsh is not compiled to use it. Typically on machines without vfork, hashstat prints only the number and size of hash buckets, but on z/OS systems, a hashstat print out would be similar to the following display:
> hashstat
> hashstat 512 hash buckets of 8 bits each
>

Related information
tcsh

**hup built-in command for tcsh: Run command so it exits on a hang-up signal**

**Format**

hup [command]

**Description**

With *command*, hup runs the command such that it will exit on a hangup signal and arranges for the shell to send it a hang-up signal when the shell exits. Commands can set their own response to hangups, overriding hup. Without an argument (allowed only in a shell script), hup causes the shell to exit on a hangup for the remainder of the script.

Related information

nohup, tcsh

**limit built-in command for tcsh: Limit consumption of processes**

**Format**

limit [-h][resource [maximum-use]]

**Description**

Limit limits the consumption by the current process and each process it creates in order to not individually exceed maximum-use on the specified resource. If no maximum-use is given, then the current limit is printed; if no resource is given, then all limitations are given. If the -h flag is specified, the hard limits are used instead of the current limits. The hard limits impose a ceiling on the values of the current limits. All hard limits can be raised only by a process which has superuser authority but a user can lower or raise the current limits within the legal range. If a user attempts to make a soft limit "unlimited", and their effective UID is not 0, then limit (or unlimit) sets the soft limit to the current hard limit value.

Resources include:

*addressspace*

The maximum address space size for the process, measured in kilobytes. If the limit is exceeded, malloc() and mmap() functions will fail. Also, automatic stack growth will fail. An attempt to set the address space size limit lower than the current usage or higher than the existing hard limit will fail.

*coredumpsize*

The size of the largest core dump file that will be created. A value of 0 (zero) prevents file creation. Dump file creation will stop at this limit.

*cputime*

The maximum amount of CPU time, in seconds, to be used by each process. If the limit is exceeded, a SIGXCPU signal is sent to the process and the process is granted a small CPU time extension to allow for signal generation and delivery. If the extension is used up, the process is terminated with a SIGKILL signal. An attempt to set the CPU limit lower than that already used will fail.

*datasize*

The data size limit is the maximum size of the break value for the process, in units of 1024 bytes. This resource always has unlimited hard and soft limits.
descriptors
The maximum number of open file descriptors allowed for the process. This number is one greater than the maximum value that can be assigned to a newly created descriptor. Any function that attempts to create a new file descriptor beyond the limit will fail. An attempt to set the open file descriptors limit lower than that already used will fail.

filesize
The largest single file which can be created by a process. A value of 0 (zero) prevents file creation. If the size is exceeded, a SIGXFSZ signal is sent to the process. If the process is blocking, catching, or ignoring SIGXFSZ, continued attempts to increase the size of a file beyond the limit will fail.

memlimit
The amount of storage, in megabytes, above the 2 gigabyte bar that a process is allowed to have allocated and unhidden at any given time. An attempt to set the storage size limit lower than the current usage or higher than the existing hard limit will fail.

stacksize
The maximum size of the automatically-extended stack region for a process. The stack is a per-thread resource that has unlimited hard and soft limits.

maximum-use can be given as a (floating point or integer) number followed by a scale factor. For cputime the default scaling is seconds, while m for minutes or h for hours, or a time of the form mm:ss giving minutes and seconds can be used. For memlimit, the default scaling is in megabytes. For all limits for which the scale is not specified, the default scale is k or kilobytes (1024 bytes); a scale factor of m or megabytes can also be used.

For both resource names and scale factors, unambiguous prefixes of the names suffice.

Usage notes
If the command fails because of an attempt to set a resource limit lower than the current amount in use or higher than the existing hard limit, the resulting error message might indicate an invalid argument.

Related information
tcsh, ulimit, unlimit
Also see setrlimit() in z/OS XL C/C++ Runtime Library Reference.

log built-in command for tcsh: Print the watch tcsh shell variable

Format
log

Description
log prints the watch shell variable and reports on each user indicated in watch who is logged in, regardless of when a user last logged in.

Restriction: The z/OS tcsh shell is compiled to use watchlog. If you attempt to use log on a z/OS system, you will get an error that says "Command not found".

Related information
tcsh, watchlog

login built-in command for tcsh: Terminate a login shell

Format
login
Description
login terminates a login shell, replacing it with an instance of /bin/login. This is one way to log off and is included for compatibility with sh.

Related information
logout, tcsh

logout built-in command for tcsh: Terminate a login shell

Format
logout

Description
logout terminates a login shell. It is especially useful if ignoreeof is set.

Related information
login, tcsh

ls-F built-in command for tcsh: List files

Format
ls-F [-switch ...] [file ...]

Description
In the tcsh shell, ls-F lists files like ls -F, but works much faster. It identifies each type of special file in the listing with a special character:

/  Directory
*  Executable
#  Block device
%  Character device
|  Named pipe
=  Socket
@  Symbolic link

If the listlinks shell variable is set, symbolic links are identified in more detail on systems that have them.

@  Symbolic link to a non-directory
>
  Symbolic link to a directory
&  Symbolic link to nowhere
**listlinks** also slows down *ls-F*.

If you use files that are set up as follows:

```
# creating a file
  touch file1
# creating a symbolic link to the file
  ln -s file1 link1
# creating a directory
  mkdir dir1
# creating a symbolic link to the directory
  ln -s dir1 linkdir1
# creating a symbolic link to a file that doesn't exist
  ln -s nonexistent linktonowhere
```

when you issue an *ls-F* with **listlinks** unset, you will get the following output:

```
> ls-F
dir1/ file1 link1@ linkdir1@ linktonowhere@
>
```

with **listlinks** set:

```
> set listlinks>
ls-F
dir1/ file1 link1@ linkdir1> linktonowhere&
>
```

If the **listflags** shell variable is set to x, a or A, or any combination thereof (for example, xA), they are used as flags to *ls-F*, making it act like *ls -xF*, *ls -Fa*, *ls -FA* or a combination *ls -FxA*. On z/OS systems, *ls -C* is the default. However, on machines where *ls -C* is not the default, *ls-F* acts like *ls -CF*, unless **listflags** contains an x, in which case it acts like *ls -xF*.

See “tcsh: Invoke a C shell” on page 671.

**Usage notes**

To view an online description for the *ls-F* command, you must type *ls-F* without the dash. To see the man page, for example, issue:

```
man lsF
```

**Related information**

*ls, tcsh*

**notify built-in command for tcsh: Notify user of job status changes**

**Format**

```
notify [%job ...]
```

**Description**

*notify* causes the shell to notify the user asynchronously when the status of any of the specified jobs (or, without %job, the current job) changes, instead of waiting until the next prompt. *job* can be a number, a string, ", %, + or '-' as described in “Jobs” on page 693. See also the **notify** shell variable.

**Related information**

*tcsh*
onintr built-in command for tcsh: Control the action of the tcsh shell on interrupts

Format
onintr [-|label]

Description
onintr controls the action of the shell on interrupts. Without arguments, onintr restores the default action of the shell on interrupts, which is to terminate shell scripts or to return to the terminal command input level. With `-`, causes all interrupts to be ignored. With `label`, causes the shell to execute a goto `label` when an interrupt is received or a child process terminates because it was interrupted.

onintr is ignored if the shell is running detached and in system startup files, where interrupts are disabled anyway.

Related information
goto, tcsh

popd built-in command for tcsh: Pop the directory stack

Format
popd [-p] [-l] [-n] [-v] [+n]

Description
popd without options, pops the directory stack and returns to the new top directory. With a number `+n`, discards the `n`th entry in the stack. All forms of popd print the final directory stack, just like `dirs`. The `pushdsilent` shell variable can be set to prevent this.

Options
-`l`
- Output is expanded explicitly to home or the path name of the home directory for the user.
-`n`
- Entries are wrapped before they reach the edge of the screen.
-`p`
- Overrides `pushdsilent`.
-`v`
- Entries are printed one per line, preceded by their stack positions.
  If more than one of `-n` or `-v` is given, `-v` takes precedence.

Related information
tcsh

pushd built-in command for tcsh: Make exchanges within directory stack

Format
pushd [-p] [-l] [-n] [-v] [name/ +n]

Description
pushd with options, exchanges the top two elements of the directory stack. If `pushdtohome` is set, pushd without arguments does `pushd ~`, like `cd`. With `name`, pushd pushes the current working directory
onto the directory stack and changes to name. If name is '-', it is interpreted as the previous working directory (see File name substitution). If dunique is set, pushd removes any instances of name from the stack before pushing it onto the stack. With a number +n, pushd rotates the n'th element of the directory stack around to be the top element and changes to it. If dextract is set, however, pushd +n extracts the n'th directory, pushes it onto the top of the stack and changes to it. So, instead of just rotating the entire stack around, dextract lets the user have the n'th directory extracted from its current position, and pushes it onto the top. For example:

```
> pushd /tmp
 /tmp ~
> pushd /bin
 /bin /tmp ~
> pushd /u
 /u /bin /tmp ~
> pushd /usr
 /usr /u /bin /tmp ~
> pushd +2
 /bin /tmp ~ /usr /u
> set dextract
dirs
 /bin /tmp ~ /usr /u
> pushd +2
 ~ /bin /tmp /usr /u
```

Finally, all forms of pushd print the final directory stack, just like dirs. The pushdsilent tcsh shell variable can be set to prevent this.

**Options**

- **-I**
  Output is expanded explicitly to home or the path name of the home directory for the user.

- **-n**
  Entries are wrapped before they reach the edge of the screen.

- **-p**
  Overrides pushdsilent.

- **-v**
  Entries are printed one per line, preceded by their stack positions.

  If more than one of -n or -v is given, -v takes precedence.

**Related information**
cd, tcsh

### rehash built-in command for tcsh: Recompute internal hash table

**Format**

```
rehash
```

**Description**

rehash causes the internal hash table of the contents of the directories in the path variable to be recomputed. This is needed if new commands are added to directories in path while you are logged in. This should only be necessary if you add commands to one of your own directories, or if a systems programmer changes the contents of one of the system directories. Also flushes the cache of home directories built by tilde (~) expansion.

**Related information**

hashstat, tcsh
repeat built-in command for tcsh: Execute command count times

**Format**

```
repeat count command
```

**Description**

The specified `command` is executed `count` times. `repeat` is subject to the same restrictions as the command in the one line `if` statement. I/O redirections occur exactly once, even if `count` is 0.

**Related information**

tcsh

sched built-in command for tcsh: Print scheduled event list

**Format**

```
sched
sched hh:mm command
sched n
```

**Description**

`sched` used alone prints the scheduled-event list. The sched shell variable can be set to define the format in which the scheduled-event list is printed. `sched hh:mm command` adds command to the scheduled-event list. For example:

```
> sched 11:00 echo It's eleven o'clock.
```

causes the shell to echo 'It's eleven o'clock.' at 11 a.m. The time can be in 12-hour a.m. or p.m. format

```
> sched 5pm set prompt='[%h] It's after 5; go home: >'
```

or it can be relative to the current time:

```
> sched +2:15 /usr/lib/uucp/uucico -r1 -sother
```

A relative time specification cannot use a.m. or p.m. format together. The third form removes item `n` from the event list:

```
> sched
1 Wed Apr 4 15:42 /usr/lib/uucp/uucico -r1 -sother
2 Wed Apr 4 17:00 set prompt='[%h] It's after 5; go home: >'
```

```
> sched -2
> sched
1 Wed Apr 4 15:42 /usr/lib/uucp/uucico -r1 -sother
```

A command in the scheduled-event list is executed just before the first prompt is printed after the time when the command is scheduled. It is possible to miss the exact time when the command is to be run, but an overdue command will execute at the next prompt. A command which comes due while the shell is waiting for user input is executed immediately. However, normal operation of an already-running command will not be interrupted so that a scheduled-event list element can be run.

This mechanism is similar to, but not the same as, the `at` command on some UNIX systems. Its major disadvantage is that it might not run a command at exactly the specified time. Its major advantage is that because sched runs directly from the shell, it has access to shell variables and other structures. This provides a mechanism for changing one's working environment based on the time of day.
**setenv** built-in command for tcsh: Set environment variable name to value

**Format**

`setenv [name [value]]`

**Description**

`setenv` without arguments, prints the names and values of all environment variables. Given `name`, sets the environment variable name to `value` or, without `value`, to the null string.

**Related information**

tcsh

**settc** built-in command for tcsh: Tell tcsh shell the terminal capability cap value

**Format**

`settc cap value`

**Description**

`settc` tells the tcsh shell to believe that the terminal capability `cap` (as defined in `termcap`) has the value `value`. No sanity checking is done. Concept terminal users might have to `settc xn no` to get proper wrapping at the rightmost column.

**Related information**

tcsh

**setty** built-in command for tcsh: Control tty mode changes

**Format**

`setty [-d|-q|-x] [-a] [+|+]mode`

**Description**

`setty` controls which tty modes (see the `stty` command description which contains lists of mode operands, such as `echo` and `echok`) the shell does not allow to change. Without arguments, `setty` lists the modes in the chosen set which are fixed (`+mode`) or off (`-mode`). The available modes, and thus the display, vary from system to system. With `+mode`, `-mode or mode` fixes mode on or off or removes control from mode in the chosen set. For example, `setty +echok echoe` fixes echok mode on and allows commands to turn echo mode on or off, both when the shell is executing commands.

**Options**

- `-a`
  List all tty modes in the chosen set whether or not they are fixed.
- `[-d|-q|-x]`
  Tells `setty` to act on the edit, quote or execute set of tty modes respectively; without `-d`, `-q` or `-x`, execute is used.
Related information
tcsh

source built-in command for tcsh: Read and execute commands from name

Format
source [-h] name [args ...]

Description
Using source, the shell reads and executes commands from name. The commands are not placed on the history list. If any arguments are given, they are placed in argv. source commands can be nested; if they are nested too deeply the shell might run out of file descriptors. An error in a source at any level terminates all nested source commands.

Options
-h
Commands are placed on the history list instead of being executed, much like history -L.

Related information
history, tcsh
telltc built-in command for tcsh: List terminal capability values

Format
telltc

Description
telltc lists the values of all terminal capabilities.

Related information
tcsh
uncomplete built-in command for tcsh: Remove completions whose names match pattern

Format
uncomplete pattern

Description
uncomplete removes all completions whose names match the specified pattern. For example, uncomplete * removes all completions. It is not an error for nothing to be uncompleted.

Related information
complete, tcsh

unhash built-in command for tcsh: Disable use of internal hash table

Format
unhash
**unlimit built-in command for tcsh: Remove resource limitations**

**Format**

unlimit [-h] [resource]

**Description**

unlimit removes the limitation on resource or, if no resource is specified, all resource limitations.

The hard limit can be lowered to any value that is greater than or equal to the soft limit. All hard limits can be raised only by a process which has superuser authority. This behavior is identical to ulimit in the z/OS shell. If a user attempts to remove the soft limit on a resource, and their effective UID is not 0, then unlimit sets the soft limit to the current hard limit value.

**Options**

- **-h**
  
  Corresponding hard limits are removed. Only the superuser can use this option.

**Related information**

limit, tcsh, ulimit

Also see setrlimit() in z/OS XL C/C++ Runtime Library Reference.

**unsetenv built-in command for tcsh: Remove environmental variables that match pattern**

**Format**

unsetenv pattern

**Description**

unsetenv removes all environment variables whose names match pattern. For example, unsetenv * removes all environment variables; we strongly recommend against this. It is not an error if nothing is removed.

**Related information**

setenv, tcsh

**watchlog built-in command for tcsh: Print the watch shell variable**

**Format**

watchlog

**Description**

watch is an alternate name for the log built-in command. It prints the watch shell variable and reports on each user indicated in watch who is logged in, regardless of when a user last logged in.
See the `version` shell variable.

**Related information**

log, tcsh

**where built-in command for tcsh: Report all instances of command**

**Format**

`where command`

**Description**

`where` reports all known instances of `command`, including aliases, built-ins and executables in path.

**Related information**

tcsh, which

**which built-in command for tcsh: Display next executed command**

**Format**

`which command`

**Description**

`which` displays the command that is executed by the shell after substitutions and path searching. This command correctly reports tcsh aliases and built-ins. The displayed command has passed access checks by the security product based on the effective user IDs. See also the `which-command` editor command.

**Related information**

tcsh, where

**tee - Duplicate the output stream**

**Format**

`tee [-ai] [file ... file ...]`

**Description**

`tee` clones an output stream. It copies the standard input to each output file as well as to the standard output.

**Options**

- `-a`
  
  Appends to (rather than overwrites) each output file.

- `-i`
  
  Ignores interrupt signals, making it suitable for use as a background process.
Examples
The following command runs the program `prog` and pipes the program's standard output into `tee`:

```
prog | tee file
```

As a result, `tee` writes the output to both the standard output and the specified file.

Localization
`tee` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values

0
   Successful completion.

1
   Failure due to any of the following reasons:
   - Out of memory when I/O buffers are allocated.
   - I/O error reading or writing to a file.
   - Error creating an output file.
   - Error opening an output file for appending.

2
   Failure due to incorrect command-line option.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Related information
cat, script

test - Test for a condition

Format

test expression [ expression ]

Description

test checks for various properties of files, strings, and integers. It does not produce any output other than error messages, but returns the result of the test as the exit status.

The second form of the test command [ expression ] is synonymous with the first. For more information about [[ ... ]], see `rewoco` in the `ssh` command description.

The following is a list of recognized operands:
The command line is a Boolean expression. The simplest expression is a string that is true if the string is nonempty (that is, has nonzero length). More complex expressions are composed of operators and operands, each of which is a separate argument (that is, surrounded by white space). The operators imply the number and type of their operands. The operators taking a file operand evaluate as false (without error) if the file does not exist.

The following is a list of recognized operands:

- **-Aa file**
  True if file has an extended access ACL entry.

- **-Ad file**
  True if file is a directory with a directory default ACL.

- **-Af file**
  True if file is a directory with a file default ACL.

- **-b file**
  True if file is a block special file (block special files are not supported)

- **-B file**
  True if the file is tagged as binary (not text)

- **-c file**
  True if file is a character special file

- **-d file**
  True if file is a directory

- **-e file**
  True if file exists

- **-Ea file**
  True if the file has the APF extended attribute

- **-Ep file**
  True if the file has the program control extended attribute

- **-Es file**
  True if the file has the shared address space extended attribute

- **-El file**
  True if the file has the shared library extended attribute

- **-f file**
  True if file is an ordinary file

- **-g file**
  True if the set-group-ID attribute of file is on

- **-h file**
  True if file is a symbolic link

- **-k file**
  True if the sticky bit is on file is on

- **-L file**
  True if file is a symbolic link

- **-Ma file**
  True if the file has any multilevel security label.

- **-n string**
  True if the length of string is greater than zero

- **-p file**
  True if file is a FIFO (named pipe)

- **-r file**
  True if file is readable (based on the security product's check against the effective user/group)

- **-s file**
  True if size of the file is nonzero
-f file
  True if the numeric file descriptor `fd` is open and associated with a terminal

-T file
  True if the file is tagged as text

-u file
  True if the set-user-ID attribute of `file` is on

-w file
  True if `file` is writable (based on the security product's check against the effective user/group)

-x file
  True if `file` is executable (based on the security product's check against the effective user/group)

-z string
  True if the length of the `string` is zero

string
  True if `string` is not a null string

string1 = string2
  True if `string1` and `string2` are identical

string1 != string
  True if `string1` and `string2` are not identical

number1 -eq number2
  True if `number1` and `number2` are equal

Within the shell, either number can be an arbitrary shell arithmetic expression; the same applies for the other five numerical comparisons that follow. Both `number1` and `number2` must be integers.

number1 -ge number2
  True if `number1` is greater than or equal to `number2`

number1 -gt number2
  True if `number1` is greater than `number2`

number1 -le number2
  True if `number1` is less than or equal to `number2`

number1 -lt number2
  True if `number1` is less than `number2`

number1 -ne number2
  True if `number1` is not equal to `number2`

file1 -nt file2
  True if `file1` is newer than `file2`

file1 -ot file2
  True if `file1` is older than `file2`

file1 -ef file2
  True if `file1` has the same device and inode number as `file2`

file -CS codeset
  True if the file is tagged with the codeset

file -ML seclabel
  True if the file has the multilevel security label `seclabel`. False if the file does not have a security label that matches `seclabel`

expr1 -a expr2
  Logical AND; true if both `expr1` and `expr2` are true

expr1 -o expr2
  Logical OR; true if either `expr1` and `expr2` is true

! expr
  Logical negation; true if `expr` is false
( expr )
  Binding: true if expr is true

The precedence of the operators in descending order is: unary operators, comparison operators, logical AND, logical OR.

The second form of the test command:

[ expression ]

is synonymous with the first.

Usage notes
1. test is a built-in shell command.
2. test can compare variables; however, if the variable is null, the expression might be incorrect for test. For example:

```bash
NULL=
  test $NULL = "so"
```

does not work, because the z/OS shell expands this to:

```bash
test = "so"
```

which is not a valid expression for test. A way to get around this is to prepend some value to both strings, as in:

```bash
test x$NULL = x"so"
```

Failure to quote variable expansions is a common mistake. For example:

```bash
test $NULL != string
```

If NULL is undefined or empty, this results in:

```bash
test != string
```

which is not a valid test expression. This problem can be fixed by enclosing $NULL in quotes.

These two examples perform basically the same function; that is, they protect the command against a variable having a possible null value.

Examples
The following command reports on whether the first positional parameter contains a directory or a file:

```bash
if [ -f $1 ]
  then echo $1 is a file
elif [ -d $1 ]
  then echo $1 is a directory
else echo $1 neither file nor directory
fi
```

This example illustrates the use of test, and is not intended to be an efficient method.

Localization
test uses the following localization environment variables:

- LANG
- LC_ALL
Exit values

0  The expression was true
1  The expression was false
2  The expression was badly formed

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

The -k, -L, -nt, -ot, -ef, -a, and -o operators plus the use of parentheses to group operators together are all extensions of the POSIX standard.

Related information

expr, find, let, ls, sh

tic - Put terminal entries in the terminfo database

Format

tic [-v]number [-c] file

Description

tic creates the terminfo database. It puts the compiled terminal entries in the directory /usr/share/lib/terminfo. If the TERMIINFO environment variable is set, the results are placed in the directory specified by the TERMIINFO environment variable rather than in the directory /usr/share/lib/terminfo.

The Curses application uses the terminfo database, which contains a list of terminal descriptions. This enables you to manipulate a terminal's display regardless of the terminal type. For information about defining the terminfo database, see the section on customizing the terminfo database in z/OS UNIX System Services Planning.

For more information about curses, see z/OS C Curses.

Options

-vNumber  Writes trace entries on the progress of tic. Number is an integer that indicates the level of verbosity. Levels 1, 2, 5, 7, 8, and 9 or greater are supported.

-c  Specifies that the input terminal specifications are to be checked for correctness, but the terminfo database is not to be updated. If an incorrect terminal specification is encountered, a message identifying the error is written to stdout. The checking continues until all of the input terminal specifications have been processed.
**file_name**

Specifies the name of a file containing the terminal specifications. Only a single file name can be specified. The files supported by z/OS Curses are identical to the specifications with the exception that the source code must be EBCDIC rather than ASCII.

If the files are copied from an MVS data set into the z/OS UNIX file system, the MVS data set must be in record format VB. If a file name is not specified, terminal specifications are read from the terminfo.src file. (The terminfo.src file is in the directory /samples.)

The .ti files are located in the /samples directory.

**Examples**

A sample command is:

```
tic /samples/ibm.ti
```

There is no output to the shell.

**Environment variables**

tic uses the following environment variable:

**TERMINFO**

Contains the path name of the terminfo database.

**Related information**

infocmp

---

**time - Display processor and elapsed times for a command**

**Format**

```
time [-p] command-line
```

tcsh shell: `time [command]`

**Description**

`time` runs the command that is given as its argument and produces a breakdown of total time to run (`real`), total time spent in the user program (`user`), and total time spent in system processor overhead (`sys`).

Times that are given are statistical, based on where execution is at a clock tick. Output is written to standard error.

`time` is a built-in shell command.

In the tcsh shell, `time` executes `command` (which must be a simple command, not an alias, a pipeline, a command list, or a parenthesized command list) and prints a time summary as described under the tcsh `time` variable (see “tcsh: Invoke a C shell” on page 671). If necessary, an extra shell is created to print the time statistic when the `command` completes. Without `command`, `time` prints a time summary for the current shell and its children.

**Option**

- `-p`

Guarantees that the historical format of the `time` command is output.
Localization

time uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_NUMERIC
• NLSPATH


Exit values

If time successfully invokes command-line, it returns the exit status of command-line. Otherwise, possible exit status values are:

0  Successful completion.
1  An error occurred in the time utility.
2  Failure due to an invalid command-line option.
2  Invalid command-line argument.
126  time found command but could not invoke it.
127  time could not find command.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.

Related information

sh, tcsh

times - Get process and child process times

Format

times [-p]

Description

times displays user and system times accumulated by the shell and commands run as children of the shell. Times are displayed in minutes and seconds. User time is CPU time spent in user programs. System time is CPU time spent in the operating system on behalf of the user process.
Options

-p
Formats the output in seconds without units. For example, 1 minute and 3.7 seconds is displayed as:

| 63.47 |

Times are displayed in minutes and seconds. User time is processor time spent in user programs. System time is processor time spent in the operating system on behalf of the user process. The output layout is:

| shell user time | shell system time |
| child user time | child system time |

Usage notes

times is a built-in shell command.

Localization

times uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


Exit values

0
Successful completion.

2
Failure that resulted in a usage message, usually due to an incorrect command-line option.

Portability

X/Open Portability Guide
The -p option is an extension to the XPG standard.

Related information

sh, time

**touch - Change the file access and modification times**

Format

```
touch [-acm] [-f agefile] [-r agefile] [-t time] file ...
touch [-acm] time file ...
```

Description

The `touch` command changes certain dates for each `file` argument. By default, `touch` sets both the date of last file modification and the date of last file access to the current time. It maintains the correct release times for software and is useful with the `make` command.
Options

-a
Sets only the access time.

-c
Does not create any file that does not exist. Normally, touch creates such files.

-m
Sets only the modification time.

If you do not specify -a or -m, touch behaves as though you specified both.

To tell touch to use a time other than the current, use one of the following options:

-f agefile
Is an obsolete version of the -r option.

-r agefile
Sets the access and modification times (as indicated by the other options) to those times kept for agefile.

-t time
Specifies a particular time by using this format:


where:
• cc is the first two digits of the year (optional).
• yy is the last two digits of the year (optional).
• mm is the number of the month (01-12) (optional).
• dd is the day of the month (optional).
• hh is the hour in 24-hour format (required).
• mm is the minutes (required).
• ss is the seconds (optional).

An obsolete (but still supported) version of this command lets you omit the -t, but the format is:

[[mm]dd]hhmm[ .ss]

or:

mmddhhmmyy[ .ss]

Examples

1. To set the modification time of newfile to the present, enter:

   touch newfile

2. To set the modification time of oldfile to 13:05 on July 3, 1994, enter:

   touch -t 9407031305 oldfile

3. To set the modification time of newfile to that of oldfile, enter:

   touch -r oldfile newfile
Environment variables

`touch` uses the following environment variable:

**TZ**

Contains the time zone that `touch` is to use when interpreting times.

Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.

Localization

`touch` uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Exit values

0  
Successful completion.

1  
Failure due to any of the following:
  • Inability to access the requested file.
  • Too early a date was specified.
  • Inability to create a file.
  • Inability to change a file’s times.

2  
Failure that resulted in a usage message, including the following causes:
  • Unknown command-line option.
  • Only one of `-t`, `-f`, or `-r` is allowed.
  • `-r` was missing the `agefile`.
  • `-t` was missing its argument.
  • Incorrect date string.

Messages

Possible error messages include:

**Age file inaccessible**

 Indicates that time could not be found for the file given with the `-f` or `-r` option either because that file does not exist or because the requesting user is not granted the appropriate permission for the file.

**Missing age file argument**

You specified `-f` or `-r`, but did not give a file name after it.

**Years earlier than year incorrect**

Your system recognizes dates only back to the given year. `touch` does not accept dates before that time.

**Bad date conversion**
tput

Only one -r, -f, or -t flag allowed

Missing the date or time argument

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Related information
cp, date
Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.

### tput - Change characteristics of terminals

#### Format

tput [-T type] capname[parm1...parm9]
tput [-T type] -s

#### Description
tput lets you change your terminal’s characteristics. The capname arguments indicate how you want to change the characteristics. Possible capnames are:

- **clear**
  - Clears the screen
- **init**
  - Initializes your terminal
- **reset**
  - Resets your terminal

tput does its work by outputting appropriate character sequences to the standard output. These character sequences are terminal-specific.

Usually, tput looks for an environment variable named TERM. If TERM exists, tput uses its value as the terminal type. If it doesn’t exist, tput assumes a default terminal type.

#### Options

- **-T type**
  - Identifies the type of your terminal. This overrides the TERM environment variable.

The second format of this command provides extensions for XPG/System V. This format of tput accepts an additional option, -S.

- **-s**
  - Takes input from standard input, one capability/capname per line. A blank line terminates input.

An additional capname is supported for System V:

- **longname**
  - Returns the long descriptive name of the terminal.

An extension to provide System V capabilities allows capname to be a capability from the terminfo database. If the capability requires arguments, they appear after the capname option.
Localization

tput uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Environment variables

tput uses the following environment variables:

TERM
Contains the current terminal type.

TERMINFO
Can be used to override the default database.

Exit values

0

tr successfully wrote an appropriate character sequence to change the terminal's characteristics, or a Boolean terminfo variable is true.

1

A Boolean terminfo variable is false.

2

Failure that generated a usage message such as:
- capname was not a recognized keyword.
- You specified an incorrect command-line option.

3

tr has no information about the terminal type given by -T or TERM.

4

The requested capname cannot be performed on your type of terminal.

>4

An error occurred.

Portability

POSIX.2 User Portability Extension, UNIX systems

Related information

stty, tabs

**tr: Translate characters**

Format

```
tr [-c | C] [-s] string1 string2
tr -s [-c | C] string1
tr -d [-c | C] string1
tr -ds [-c | C] string1 string2
```
Description

`tr` copies data that is read from the standard input to stdout, substituting or deleting characters as specified by the options and `string1` and `string2`. `string1` and `string2` are considered to be sets of characters. In its simplest form, `tr` translates each character in `string1` into the character at the corresponding position in `string2`.

Note: `tr` works on a character basis, not on a collation element basis. Thus, for example, a range that includes the multicharacter collation element `ch` in regular expressions, does not include it here.

Options

- **-c**
  If the variable `_UNIX03` is unset or is not set to YES, the behavior of `-c` option complements the set of characters that are specified by `string1`. `tr` constructs a new set of characters, consisting of all the characters not found in `string1` and uses this new set in place of `string1`.
  
  If the variable `_UNIX03=YES` is set, the behavior of `-c` option complements the set of values that are specified by `string1`. `tr` constructs a new set and the complements of the values specified by `string1` (the set of all possible binary values, except for those that are specified in the `string1` operand) are placed in this new set in ascending order by binary value. The new set is used in place of `string1`.

- **-C**
  Complements the set of characters specified by `string1`. This means that `tr` constructs a new set and the complements of the characters that are specified by `string1` (the set of all characters in the current character set, as defined by the current setting of LC_CTYPE, except for those that are specified in the `string1` operand) are placed in this new set in ascending collation sequence, as defined by the current setting of LC_COLLATE. This behaves the same as `-c` when the variable `_UNIX03` is unset or is not set to YES.

- **-d**
  Deletes input characters that are found in `string1` from the output.

- **-s**
  `tr` checks for sequences of a `string1` character repeated several consecutive times. When this happens, `tr` replaces the sequence of repeated characters with one occurrence of the corresponding character from `string2`. If `string2` is not specified, the sequence is replaced with one occurrence of the repeated character itself. For example:

  ```
  tr -s abc xyz
  ```

  translates the input string `aaaabccccb` into the output string of `xyzy`.

  If you specify both the `-d` and `-s` options, you must specify both `string1` and `string2`. In this case, `string1` contains the characters to be deleted, whereas `string2` contains characters that are to have multiple consecutive appearances replaced with one appearance of the character itself. For example:

  ```
  tr -ds a b
  ```

  translates the input string `abbbaaacbb` into the output string `bcb`.

  The actions of the `-s` option take place after all other deletions and translations.

String options

You can use the following conventions to represent elements of `string1` and `string2`:

- **character**
  Any character not described by the conventions that follow represents itself.

- **\**
  An octal representation of a character with a specific coded value. It can consist of one, two, or three octal digits (01234567). Double-byte characters require multiple, concatenated escape sequences of this type, including the leading `\` for each byte.
The `\` (backslash) character is used as an escape to remove the special meaning of characters. It also introduces escape sequences for nonprinting characters, in the manner of C character constants: `\b`, `\f`, `\n`, `\r`, `\t`, and `\v`.

c1-c2
In the POSIX locale, as long as neither endpoint is an octal sequence of the form `\ooo`, this represents all characters between characters `c1` and `c2` (in the current locale's collating sequence) including the end values. For example, ‘`a–z`’ represents all the lowercase letters in the POSIX locale, whereas ‘`A–Z`’ represents all that locale's uppercase letters. One way to convert lowercase and uppercase is with the following filter:

```
tr 'a-z' 'A-Z'
```

This is not, however, the recommended method; use the `[[:class:]` construct instead.

If the second endpoint precedes the starting endpoint in the collation sequence, it causes an error.

If either or both of the range endpoints are octal sequences of the form `\ooo`, this represents the range of specific coded values between the two range endpoints, inclusive.

This construct `c1–c2` is only applied in POSIX locale.

**Note:** The current locale has a significant effect on results when this method is used to specify subranges. If the command is required to give consistent results irrespective of locale, the use of construct c1-c2 should be avoided.

[c*n]
This represents `n` repeated occurrences of character `c`. (If `n` has a leading zero, `tr` assumes it is octal; otherwise, it is assumed to be decimal.) You can omit the number for the last character in a subset. This representation is valid only in `string2`.

[:class:]
This represents all characters that belong to the character class `class` in the locale indicated by LC_CTYPE. When the class `[:upper:]` or `[:lower:]` appears in `string1` and the opposite class, `[:lower:]` or `[:upper:]` appears in `string2`, `tr` uses the LC_CTYPE `tolower` or `toupper` mappings in the same relative positions.

[=c=]
This represents all characters that belong to the same equivalence class as the character `c` in the locale indicated by LC_COLLATE. Only international versions of the code support this format.

**Usage notes**
When `string2` is shorter than `string1`, `tr` does not pad `string2`. The remaining characters in `string1` will not be translated. For example:

```
tr '0123456789' 'd'
```

only translates '0' to 'd', '123456789' remain unchanged.

Coding the example in the following way:

```
tr '0123456789' '[d*]'
```

translates all digits to the letter 'd'.

**Examples**
This example creates a list of all words (strings of letters) found in `file1` and puts it in `file2`:

```
tr -cs "[:alpha:]" "[\n*]" <file1 >file2
```
Environment variables

`tr` uses the following environment variable: `_UNIX03`.

For more information about the effect of _UNIX03 on this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Localization

`tr` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


Exit values

0
  Successful completion.

1
  Failure because of unknown command-line option, or too few arguments.

Portability

POSIX.2, X/Open Portability Guide.

`tr` is compatible with earlier versions of both the UNIX Version 7 and System V variants of this command, but with extensions (C escapes, handles ASCII NUL, globalization).

---

`trap` - Intercept abnormal conditions and interrupts

Format

`trap ['handler'] [event ...]`

Description

`trap` intercepts certain kinds of exception conditions. Any signal may be intercepted by specifying an event corresponding to the signal number.

With an event of ERR, `trap` invokes the handler after receiving any having a nonzero exit status. The exception to this is conditions in `if`, `while`, and `until` statements. This trap is not inherited within a function.

With a trap number of 0 or EXIT, `trap` invokes the handler during exit from the shell. Within a function, it is invoked during exit from the function.

Any other event corresponds to a signal number or signal name. (See `kill` for a table of valid signal numbers and their names.) If a signal is being ignored when you enter the shell, the shell continues to ignore it without regard to any traps.

Because system initialization sets the value of the SIGIOERR signal to `ignore`, this signal cannot be set by `trap`. 
The *handler* argument is a command list. It is usually more than one word, and so you must quote it to appear as a single argument. It is scanned when the trap function is initially invoked. When the trap condition is raised, the shell scans the command list again and runs the commands. A missing argument or an argument of - (dash) resets the default trap condition. A null argument ("") causes the trap condition to be ignored.

If there are no arguments at all, `trap` prints a list of all the traps and their commands.

**Usage notes**

`trap` is a special built-in shell command.

**Examples**

1. On error or exit, this example deletes a temporary file that was created during command execution.

   ```bash
   trap 'rm -f /tmp/xyz$$; exit' ERR EXIT
   ```

   When an interrupt signal is received, the example prompts whether to abort, and exits if the answer is y.

   ```bash
   trap 'read REPLY?'ABORT??'
   case $REPLY in
   y) exit 1;;
   esac' 2
   ```

2. This example saves your shell history file (specified by the value you give the HISTFILE environment variable) before timing you out, so you can restore it when you log on again.

   ```bash
   trap 'cp $HISTFILE $HOME/old_hist.bak; exit' ALRM
   ```

**Localization**

`trap` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Exit values**

0  
Successful completion.

1  
Failure due to any of the following reasons:
- Incorrect signal name.
- Incorrect signal number.

2  
Incorrect command-line argument.

**Messages**

Possible error messages include:

*name not a valid trap name*

You specified an unrecognized trap name. The usual cause of this error is a typing mistake on the command line.
true - Return a value of 0

Format
true [argument ...]

Description
true simply yields an exit status of zero (success). It ignores any arguments given on the command line. This can be surprisingly useful, for example, when you are evaluating shell expressions for their side effects.

Usage notes
ttrue is a built-in shell command.

Localization
ttrue uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH


Exit values
Since true always succeeds, the only possible exit status is:

0
  Successful completion.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.

Related information
sh

tso - Run a TSO/E command from the shell

Format
tso [-o] [-t] TSO_command
Description

tso runs a TSO/E command from the shell using the TSO/E service routine or the OMVS interface.

Restriction: The tso command cannot be used to issue authorized TSO commands when the TSO/E command service is used. See “tsocmd - Run a TSO/E command from the shell (including authorized commands)” on page 757 if that is required.

Options

-o
Specifies that the command be issued through the OMVS interface.

-t
Specifies that the command be issued through the TSO/E service routine. If a mini-TSO/E environment is to be established, use environment variables to specify the allocations that you need. Because the tso command uses an exec() of itself to clean up at completion, the tso command must be run using a valid PATH environment variable or be run using its full path name, /bin/tso.

If you do not specify an option, the following rules determine how to run the TSO/E command:

• If stdout is not a tty, the TSO/E service routine is used because it is possible that the command output will be redirected to a file or piped to another command.

• If the controlling tty supports 3270 passthrough mode, the OMVS interface is used.

• If neither rule is applicable, then the TSO/E service routine is used.

Examples

1. To use OPUT to copy an MVS data set to a file in your current directory, issue:

   tso -t "oput 'source.c(hello)' 'hello.c' "

   If you do not specify -t, the command is run in your TSO/E session through OMVS, if possible. This copies the file to a file relative to the working directory of your TSO/E session, which is typically your home directory.

   Quotation marks are used around the command to avoid shell parsing.

2. To use OPUTX to copy all members of a PDS to your current directory, issue:

   tso -t "oputx source.c . lc suffix(c)"

   If you do not specify the -t option, the command is run in your TSO/E session through OMVS, if possible. This copies the file to a file relative to the working directory of your TSO/E session which is typically your home directory.

   Quotes are used around the command to avoid shell parsing.

Because OPUTX uses ISPF, allocations for the ISPF DD names must be performed to run this command. The following is an example of the environment variables that are set to perform these allocations. This can be included in your .profile for convenience. Make sure the export statements start in column one. The data set names might differ on your system.

# Assign the DD names to allocate
#
export TSOALLOC=ispprof:ispplib:ispmlib:isptlib:ispplib:ispslib:\
isptabl:ispllog:sysexec
#
# Allocate an empty, temporary ISPF profile data set
#
export ispprof="alloc new unit(sysvio) space(1,1) cyl dir(5) \recfm(f,b) lrecl(80) blksize(3120)"
#
# Allocate an empty, temporary ISPF table data set
#
export isptabl="alloc new unit(sysvio) space(1,1) cyl dir(5) \
Environment variables

If the **tso** command is to be run through the TSO/E service routine, you might need to perform allocations or other customization for the TSO/E environment. These tasks can be specified using environment variables. You can use the following environment variables:

**SYSEEXEC**

Specifies the allocation specification for the SYSEXEC DD name. If the TSOALLOC variable is set, this variable is not automatically used.

**SYSPROC**

Specifies the allocation specification for the SYSPROC DD name. If the TSOALLOC variable is set, this variable is not automatically used.

**TSOALLOC**

Specifies the names of the environment variables that contain allocation specifications. The names are separated by colons. Case is respected; lowercase letters are treated as lowercase. The names of the environment variables also correspond to the name of the DD name to be allocated. The DD name is always treated as uppercase but the variable name can be specified in mixed case to avoid possible conflict with similar environment variable names.

The HOLD attribute is supported for SYSOUT allocation in the BPXWDYN text interface and TSOALLOC environment variable.

**TSOOUT or tsoout**

Specifies the allocation attributes for SYSTSPRT. The format of the variable is in bpwxwdyn format without a dd name. For example:

```
export tsoout="alloc path('/dev/tty') pathopts(owronly) filedatal(text)"
```

If both TSOOUT and tsoout are used, TSOOUT takes precedence.

**TSOPREFIX**

Specifies a prefix for temporary data sets that need to be cataloged. Lowercase letters are treated as uppercase letters. If you do not specify this variable, the user’s login name (user ID) is used.
**TSOPROFILE**

Resets the profile with the arguments that you specify when running the TSO/E command. (The specified arguments replace the default values.) For example, to set the TSO prefix and to turn off message IDs, issue:

```
export TSOPROFILE="prefix(wjs) nomsgid"
```

The value of this variable is passed to the TSO/E PROFILE command as is. If the PROFILE command fails, the requested command is not run. The output from the PROFILE command is sent to stdout along with the PROFILE command that was issued.

An allocation specification can be either a list of cataloged data set names separated by colons or a data set allocation request. If a list of data set names is used, lowercase letters are treated as uppercase and the data set names must be fully qualified.

Specify a request for data allocation by beginning the specification with the keyword `alloc` followed by keywords or keyword-value pairs in a format similar to the TSO/E ALLOCATE command. Keys are separated by blanks. A complete listing of keys can be found in *z/OS Using REXX and z/OS UNIX System Services*. Following is a list of keys:

- **DA (data set name [(member name)]) | DSN (data set name [(member name)])**
  - Data set name to allocate. The name must be fully qualified and can include a member name. Quotation marks can be used but are ignored.

- **MOD | NEW | OLD | SHR**
  - Specifies the status of the data set.

- **CATALOG | DELETE | KEEP | UNCATALOG**
  - Specifies the data set disposition.

- **TRACKS**
  - Specifies that space be allocated in units of tracks.

- **CYL**
  - Specifies that space be allocated in units of cylinders.

- **DIR(directory blocks)**
  - Specifies the number of directory blocks.

- **SPACE(primary[,secondary])**
  - Specifies that primary and (optionally) secondary space be allocated.

- **VOL(volume serial)**
  - Specifies the VOLSER.

- **UNIT(unit name)**
  - Specifies the unit name, device type, or unit address.

- **SYSOUT[(class)]**
  - Specifies that a sysout data set is to be allocated and optionally defines the output class.

- **HOLD**
  - Specifies that the output data is to be held until released by user or operator.

- **WRITER(external writer name)**
  - Specifies the external writer.

- **FORMS(forms name name)**
  - Specifies the print form.

- **DEST(destination)**
  - Specifies the output destination.

- **COPIES(number of copies)**
  - Specifies the number of copies to be printed.

- **DUMMY**
  - Specifies that a dummy data set be allocated.
BLKSIZE(block size)
   Specifies the block size.
LRECL(record length)
   Specifies the logical record length.
DSORG(PS|PO|DA)
   Specifies the data set organization.
RECFM(format[,format...])
   Specifies the record format. The values are A, B, D, F, M, S, T, U, and V. You can combine several of these values.
STORCLAS(storage class)
   Specifies the storage class.
MGMTCLAS(management class)
   Specifies the management class.
DATACLAS(data class)
   Specifies the data class.
RECFM(LS)
   Specifies that a VSAM linear data set be created.
DSNTYPE(LIBRARY|PDS|HFS)
   Specifies the data set type.
SPIN(UNALLOC)
   Specifies that a sysout data set be spun off at allocation.
NORECALL
   Specifies that the allocation request be failed if the data set is migrated.
PATH('pathname')
   Specifies that the allocation is for a file in the z/OS UNIX file system.
PATHOPTS(pathopt,[pathopt]...)
   Specifies a list of path options: ORDWR OEXCL OSYNC OTRUNC OCREATE OWRONLY ORDONLY OAPPEND ONOCTTY ONONBLOCK.
PATHMODE(pathmode,[pathmode]...)
   Specifies a list of path modes: SIRUSR SIWUSR SIXUSR SIWRXU SIWGRP SIXGRP SIWRXG SIROTH SIWOTH SIXOTH SIWRXO SISUID SISGID SISVTX
PATHDISP(KEEP|DELETE[,KEEP|DELETE])
   Specifies the normal and abnormal file disposition.
FILEDATA(TEXT|BINARY)
   Specifies whether the data is to be treated as text or binary.

Messages
0 - 254
   Successful completion.
255
   The return code is outside the range 0 - 254 or the tso command ended in error.

Related information
tsocmd
**tsocmd - Run a TSO/E command from the shell (including authorized commands)**

**Format**

`tsocmd TSO_command`

**Description**

`tsocmd` runs a TSO/E command from the shell by using the TSO/E terminal monitor program (IKJEFT01). Unlike the `tso` command, the `tsocmd` command can be used to issue authorized TSO/E commands. (For more information about the `tsocmd` command, see “tso - Run a TSO/E command from the shell” on page 752). Because the TSO Terminal Monitor Program (TMP) is run in a separate address space and process from the `tsocmd` command, TSO/E commands that are issued do not affect the environment that the `tsocmd` is issued from.

See z/OS TSO/E Programming Services for more information about the TSO/E TMP.

**Usage notes**

1. The BPXWRFD environment variable can be set to cause file descriptors to be inherited by the TSO command processor. See the description of environment variable BPXWRFD “Environment variables” on page 757 for information about which file descriptors are inherited.

2. Do not issue 'tsocmd output' from the OMVS shell. If you do, you might not receive any output.

3. The standard input stream can also be queued as input to the TSO command. For example, if you have a file redirected as input and you run a TSO command before processing that file, some or all of the file can be queued to the TSO command. If input is the terminal, the queued input can be queued to the TSO command. This characteristic can be used to interact with some TSO commands. To prevent an abend while reading from SYSTSIN, the regular file allocated to STDIN (fd0) must have logical records that are at most 80 characters long.

**Examples**

1. To issue the authorized RACF command RDEFINE, issue:

```bash
   tsocmd "RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)"
```

   Quotation marks are used around the command to avoid shell parsing.

2. To use OGET to copy from a file in your current directory to an MVS data set, issue:

```bash
   tsocmd "oget hello.c 'source.c(hello)"
```

   Quotation marks are used around the command to avoid shell parsing.

3. The examples in the `tso` command also apply to `tsocmd` where one replaces the string 'tso -t' with the string 'tsocmd'. See the examples in the `tso` command section (“Examples” on page 753) for more information.

**Environment variables**

Depending on the `tsocmd` command that is being issued, you might need to perform allocations or other customization for the TSO/E environment. These environment variables can be used:

**BPXWRFD**

Specifying YES in a REXX program before the TSO process is started causes the TSO process to inherit open file descriptors 10 through 99.
Specifying a numeric value in a REXX program before the TSO process is started causes the TSO process to inherit open file descriptors 10 through that numeric value. The maximum value that can be specified is limited by the setting of the system configuration (SYSCONF) value for OPEN_MAX. If a higher value is specified, then it is limited by OPEN_MAX. If a value lower than 10 or a nonnumeric value other than YES is specified, then no additional file descriptors beginning at 10 are inherited.

**SYSEXEC**
Specifies the allocation specification for the SYSEXEC DD name. If the TSOALLOC variable is set, this variable is not automatically used.

**SYSPROC**
Specifies the allocation specification for the SYSPROC DD name. If the TSOALLOC variable is set, this variable is not automatically used.

**TSOALLOC**
Specifies the names of the environment variables that contain allocation specifications. The names are separated by colons. Case is respected; lowercase letters are treated as lowercase. The names of the environment variables also correspond to the name of the DD name to be allocated. The DD name is always treated as uppercase but the variable name can be specified in mixed case to avoid possible conflict with similar environment variable names.

The HOLD attribute is supported for SYSOUT allocation in the BPXWDYN text interface and TSOALLOC environment variable.

**TSOPROFILE**
Resets the profile with the arguments that you specify with the TSO/E command. (The specified arguments replace the default values.) For example, to set the TSO prefix and to turn off message IDs, issue:

```
export TSOPROFILE="prefix(wjs) nomsgid"
```

The value of this variable is passed to the TSO/E PROFILE command as is. If the PROFILE command fails, the requested command is not run. The output from the PROFILE command is sent to stdout along with the PROFILE command that was issued.

An allocation specification can be either a list of cataloged data set names that are separated by colons or a data set allocation request. If a list of data set names is used, lowercase letters are treated as uppercase and the data set names must be fully qualified.

Specify a request for data set allocation by beginning the specification with the keyword `alloc` followed by keywords or keyword-value pairs in a format similar to the TSO/E ALLOCATE command. Keys are separated by blanks. A complete listing of keys can be found in *z/OS Using REXX and z/OS UNIX System Services*. You can also refer to the list of keys in the `tso` command description in `tso`.

**Exit values**

0
The TSO/E command was successful.

1-254
The TSO/E command ended in an error with the listed return code.

255
The TSO/E command ended with an unexpected error, the TSO/E command return code is outside the range 0-254, no command was entered, an error occurred when processing an environment variable, or the command was not found.

**Localization**
The `tsocmd` command is not sensitive to a user's locale. It is up to the user to provide input that is acceptable to TSO/E.
Related information
tso

tsort: Sort files topologically

Format

tsort [file]

Description

tsort reads input from files (or from the standard input if you do not specify a file) and produces an ordered list of items consistent with a partial ordering of items provided by the input.

Input to tsort takes the form of pairs of items (nonempty strings) separated by blanks. A pair of two different items indicates ordering. A pair of identical items indicates presence, but not ordering.

Examples

The command:

```
tsort <<EOF
a b c d e
f g e f
h h
EOF
```

produces the output:

```
a
b
c
d
e
f
g
h
```

Localization

tsort uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


Exit values

0
   Successful completion.

>0
   An error occurred.

Portability

POSIX.2, X/Open Portability Guide

X/Open Portability Guide.
tty - Return the user's terminal name

Format
tty [-s]

Description
tty displays the file name of the terminal device associated with the standard input.

Options
-s
  Does not display the name; the exit status of tty indicates whether the standard input is a terminal.

Localization
tty uses the following localization environment variables:
  • LANG
  • LC_ALL
  • LC_CTYPE
  • LC_MESSAGES
  • NLSPATH

Exit values
0
  Standard input is a terminal.
1
  Standard input is not a terminal.
2
  Failure because of an unknown command-line option, or too many arguments.

Messages
Possible error messages include:

Not a tty
  The standard input is not associated with a terminal.

Portability
POSIX.2, X/Open Portability Guide, UNIX systems.
The POSIX standard considers the -s option to be obsolete.

type - Tell how the shell interprets a name

Format
type name ...
Description

type identifies the nature of one or more names. Names can be shell reserved words, aliases, shell functions, built-in commands, or executable files. For executable files, the full path name is given.

Usage notes

type is a built-in shell command.

Localization


type uses the following localization environment variables:

• LANG
• LC_ALL
• LC_MESSAGES
• NLSPATH

Exit values

0
   Successful completion.
2
   Failure because of an incorrect command-line argument.

Messages

name is not found
   type could not locate the specified name. Check that the name was specified properly and that you have the appropriate permissions.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

Related information

alias, command, sh, whence

typeset - Assign attributes and values to variables

Format

  typeset + -f [tux] name ...
  typeset [ + -lprtuxH] [ + -iLR[number]] [variable [=value] ...]

Description

Invoking typeset with no options displays a list of all variables and their attributes. This list is sorted by variable name and includes quoting so that it can be reinput to the shell with the built-in command eval. When only arguments of the form +option are specified, typeset displays a list of the variables that have all specified attributes set. When only arguments of the form -option are present, typeset displays a list of all the variables having all the specified attributes set, and also displays their values.

When the $ option is used, typeset applies to functions; otherwise, it applies to variables. For functions, the only other applicable options are -t, -u and -x.
If the command line contains at least one variable, the attributes of each variable are changed. In this case, parameters of the form `-option` turn on the associated attributes. Parameters of the form `+option` turn off the associated attributes. (Notice that, contrary to what you might expect, `-` means on, and `+` means off.) Parameters of the form `variable=value` turn on the associated attributes and also assign `value` to `variable`.

When `typeset` is invoked inside a function, a new instance of each variable is created. After the function ends, each variable is restored to the value and attributes it had before the function was called.

**Options**

- `-f`
  - Specifies attributes of functions.

- `-H`
  - Performs file mapping from POSIX to the host name.

- `-l[number]`
  - Marks each variable as having an integer value, thus making arithmetic faster. If `number` is given and is nonzero, the output base of each `variable` is `number`. The default is decimal.

- `-l`
  - Converts uppercase characters to lowercase in any value that is assigned to a variable. If the `-u` option is turned on, this option turns it off.

- `-P`
  - Writes output to the coprocess. This option is not currently implemented.

- `-r`
  - Makes each variable read-only. See `readonly`.

- `-T`
  - Tags each variable. Tags are user-defined, and have no meaning to the shell. For functions with the `-f` option, this turns on the `xtrace` option. See `set` for a discussion of the `xtrace` option.

- `-u`
  - Converts lowercase characters to uppercase in any value that is assigned to a variable. If the `-l` option is turned on, this option turns it off.

  When used with `-f`, the `-u` option indicates that the functions that are named in the command line are not yet defined. The attributes that are specified by the `typeset` command are applied to the functions once they are defined.

- `-x`
  - Sets each variable for automatic export. See `export`.

The last three options that follow justify, within a field, the values assigned to each variable. The width of the field is `number` if it is defined and is nonzero; otherwise, the width is that of the first assignment that is made to `variable`.

- `-L[number]`
  - Left-justifies the values that are assigned to each variable by first removing any leading blanks. Leading zeros are also removed if the `-Z` option has been turned on. Then, blanks are added on the end or the end of the value is truncated as necessary. If the `-L` flag is turned on, this option turns it off.

- `-R[number]`
  - Right-justifies the values that are assigned to each variable by adding leading blanks or by truncating the start of the value as necessary. If the `-L` flag is turned on, this option turns it off.

- `-Z[number]`
  - Right-justifies values that are assigned to each variable. If the first nonblank character of value is a digit, leading zeros are used. See also the `-L` option.
Usage notes

typeset is a built-in shell command as well as a separate utility.

An autoloaded function is defined (loaded) by the /bin/sh shell when invoked as a command name, it's not already defined to the shell, and the function definition file is found in a directory specified in the FPATH shell variable. (For more information see Command execution under the sh command.) To replace an autoloaded function, use the unset -f name command. The next time the function name is invoked, the FPATH search will find the new version.

Localization

typeset uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


Exit values

0  Successful completion.

2  Failure due to an incorrect command-line argument.

If the command is used to display the values of variables, the exit status value is the number of names that are incorrect.

Messages

Possible error messages include:

- **Base number not in \[2,36\]**
  You used the -i option to specify a base for an integer, but the base was not in the range 2 through 36.
  All bases must be in this range.

- **name not a function**
  You tried to declare the given name as a function, but the name already referred to something that was not a function (for example, a variable).

Portability

POSIX.2. It is an extension to the POSIX.2 and XPG standards.

Related information

export, readonly, sh

---

uconvdef - Create binary conversion tables

Format

```
uconvdef [-f SrcFile] [-v] uconvTable
```
Description

uconvdef reads SrcFile and creates uconvTable, a binary conversion table. SrcFile is the input source file that defines a mapping between UCS-2 and multibyte code sets. UCS-2 is the Universal Character Set, coded in 2 octets, as defined by ISO/IEC 10646-1:1993(EE), while multibyte code sets consists of one or more bytes per character.

uconvTable is in a format that can be opened and read by iconv conversion functions.

Options

- `f SrcFile`
  SrcFile is the input source file that defines a mapping between UCS-2 and another single or multibyte code set. If this option is not used, standard input is read. For information about the format of the input source table, refer to the `ucmap` description in z/OS XL C/C++ Programming Guide.

- `v`
  Specifies that the SrcFile file statements be displayed.

uconvTable
  Specifies the path name of the compiled table created by the uconvdef command. This file defines conversions into and out of UCS-2.

Examples

To create the compiled uconvTable that defines the conversion table between IBM-1047 and UCS-2, issue:

```
uconvdef -f IBM-1047.ucmap /usr/lib/nls/locale/uconvTable/IBM-1047
```

The \ (backslash) is a line continuation character that is needed if the command is broken into multiple lines.

Exit values

0
  Successful completion.

>0
  An error occurred.

Related information

iconv

The iconv subroutine, iconv_close subroutine, iconv_open subroutine (refer to z/OS XL C/C++ Programming Guide).

ulimit - Set process limits

Format

```
ulimit [-SHaAcdfMnst] [num]
```

Description

ulimit sets or displays the resource limits on processes created by the user.
Options

-S
Set or display the soft limits. The soft limit may be modified to any value that is less than or equal to the hard limit. For certain resource values, the soft limit cannot be set lower than the existing usage.

-H
Set or display the hard limits. The hard limit may be lowered to any value that is greater than or equal to the soft limit. The hard limit can be raised only by a process which has superuser authority.

-a
Display all resource limits that are available.

-A
Set or display the maximum address space size for the process, in units of 1024 bytes. If the limit is exceeded, storage allocation requests and automatic stack growth will fail. An attempt to set the address space size limit lower than the current usage or to set the soft limit higher than the existing hard limit will fail.

-c
Set or display the core file limit. The core file limit is the maximum size of a dump of memory (in 512-byte blocks) allowed for the process. A value of 0 (zero) prevents file creation. Dump file creation will stop at this limit.

-d
Set or display the data size limit. The data size limit is the maximum size of the break value for the process, in units of 1024 bytes. This resource always has unlimited hard and soft limits.

-f
Set or display the file size limit. The file size limit is the maximum file size (in 512-byte blocks) allowed for the process. A value of 0 (zero) prevents file creation. If the size is exceeded, a SIGXFSZ signal is sent to the process. If the process is blocking, catching, or ignoring SIGXFSZ, continued attempts to increase the size of a file beyond the limit will fail.

-M
Set or display the amount of storage above the 2 gigabyte bar that a process is allowed to have allocated and unhidden, in megabyte increments. An attempt to set the storage size limit lower than the current usage or to set the soft limit higher than the existing hard limit will fail.

The amount of storage that ulimit -M displays does not necessarily reflect the MEMLIMIT setting found in the user’s RACF OMVS segment. The value displayed will depend on how the user entered the OMVS shell and whether a change of identity was performed.

-n
Set or display the file descriptors limit. The file descriptors limit is the maximum number of open file descriptors allowed for the process. This number is one greater than the maximum value that may be assigned to a newly created descriptor. Any function that attempts to create a new file descriptor beyond the limit will fail. An attempt to set the open file descriptors limit lower than that already used will fail.

-s
Set or display the stack size limit. The stack size limit is the maximum size of the stack for a process, in units of 1024 bytes. The stack is a per-thread resource that has unlimited hard and soft limits.

-t
Set or display the CPU time limit. The CPU time limit is the maximum amount of CPU time (in seconds) allowed for the process. If the limit is exceeded, a SIGXCPU signal is sent to the process and the process is granted a small CPU time extension to allow for signal generation and delivery. If the extension is used up, the process is terminated with a SIGKILL signal. An attempt to set the CPU limit lower than that already used will fail.

\textit{num}

The new limit. \textit{num} can be specified as “unlimited”. 
Usage notes

1. **ulimit** is a built-in shell command. It cannot be used with the tcsh shell.
2. If the command fails because of an attempt to set a resource limit lower than the current amount in use or higher than the existing hard limit, the resulting error message may indicate an invalid argument.

Localization

**ulimit** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


Related information

setrlimit in z/OS XL C/C++ Runtime Library Reference.

------------------

**umask - Set or return the file mode creation mask**

**Format**

`umask [-S] [mode]`

tcsh shell: `umask [value]`

**Description**

**umask** sets the file-creation permission-code mask of the invoking process to the given `mode`. You can specify the `mode` in any of the formats that are recognized by `chmod`. For more information, see `chmod` for more information.

The `mode` can be specified in symbolic (rwx) or octal format. The symbolic form specifies what permissions are allowed. The octal form specifies what permissions are disallowed.

The file-creation permission-code mask (often called the umask) modifies the default (initial) permissions for any file that is created by the process. The umask specifies the permissions which are not to be allowed.

If the bit is turned off in the umask, a process can set it on when it creates a file. If you specify:

```
umask a=r
```

you have allowed files to be created with read and execute access for all users. If you were to look at the mask, it would be 0222. The write bit is set, because write is not allowed. If you want to permit created files to have read, write, and execute access, then set umask to 0000. If you call `umask` without a `mode` argument, `umask` displays the current umask.

To calculate the permissions that will result from specific umask values, subtract the umask from 666 for files, and from 777 for directories. For example, a umask of 022 results in permissions of 644.

In the tcsh shell, `umask` sets the file creation mask to `value`, which is given in octal. Common values for the mask are 002, giving all access to the group and read and execute access to others, and 022, giving read and execute access to the group and others. Without `value`, `umask` prints the current file creation mask. For more information, see “tcsh: Invoke a C shell” on page 671.
Options

-S
  Displays the umask in a symbolic form:

  u=perms, g=perms, o=perms

giving owner, group and other permissions. Permissions are specified as combinations of the letters r (read), w (write), and x (execute).

Localization

umask uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values

0
  Successful completion.
1
  Failure due to an incorrect command-line argument, or incorrect mode.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

Related information

chmod, tcsh

unalias - Remove alias definitions

Format

unalias name ...
unalias -a

tcsh shell: unalias pattern

Description

unalias removes each alias name from the current shell execution environment.

In the tcsh shell, unalias removes all aliases whose names match pattern. For example,

unalias *

removes all aliases. It is not an error for nothing to be unaliased. For more information, see “tcsh: Invoke a C shell” on page 671.
uname

Options
-a
   Removes all aliases in the current shell execution environment.

Localization
unalias uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH

Usage notes
unalias is a built-in shell command.

Exit values
0
   Successful completion.
1
   There was an alias that could not be removed.
2
   Failure due to an incorrect command-line option or there were two aliases that could not be removed.
>2
   Tells the number of aliases that could not be removed.

Portability

Related information
alias, sh, tcsh

uname - Display the name of the current operating system

Format
uname [-aImnsv]

Note: Option -I is an uppercase i, not a lowercase L.

Description
The uname command lets shell scripts and other programs determine configuration information about the machine upon which the shell is running.

Options
The following options select the information to be displayed:
-a
  All fields (equivalent to -mnrsv).

-I
  The product name information. This option affects the value of the system name, release, and version fields. This option might affect the output of the -a, -r, -s and -v options. When -I is not specified (the default), the OS/390® product name information is returned.

-m
  The processor or machine type.

-n
  The node name of this particular machine. The node name is set by the SYSNAME sysparm (specified at IPL), and it usually differentiates machines running at a single location.

-r
  The release (minor version) number of the operating system.

-s
  The name of the operating system. This is the default output, when no options are given.

-v
  The version (major version) number of the operating system.

uname displays the selected information in the following order:

<system name> <nodename> <release> <version> <machine>

Examples
1. The following shell command changes the prompt to identify the node name of the system:

   export PS1="'uname –n ' $ "

2. The following indicates what is returned when you specify the –I option and when you do not (not specifying –I is the default):

   If running on z/OS 1.2:
   issuing >uname -rsv gives you
   OS/390 12.00 03
   issuing >uname -rsvI gives you
   z/OS 02.00 01

Localization

uname uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


Exit values

0
  Successful completion

1
  Failure due to inability to find the desired information

2
  Failure due to an incorrect command-line option
uncompress - Undo Lempel-Ziv compression of a file

Format
uncompress [-cDfv [file]]

Description
uncompress expands compressed data written by the Lempel-Ziv compression program compress. Data is read from file or the standard input. On UNIX systems, the name of the file to be uncompressed must end with .Z. If it doesn't, uncompress adds one before looking for the file. It places the uncompressed output in a file with the same name but without the .Z extension. If this file already exists, uncompress asks if you want to overwrite it, unless you specify the -f option.

Since the number of bits of compression is encoded in the compressed data, uncompress automatically uses the correct number of bits. This includes the 9–14 bit compression range specified by POSIX.

Options
-c
   Writes uncompressed output to the standard output (like zcat).

-D
   Must be used to uncompress a sorted dictionary file compressed using the -D option of compress.

-f
   Forces file to be uncompressed, regardless of whether a file with the same base name already exists.

-V
   Prints version number information for uncompress.

-v
   Displays name of each file when it is uncompressed.

Localization
uncompress uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Exit values
0
   Successful completion

1
   Failure due to any of the following reasons:
• Unknown command-line option.
• Inability to obtain information about an argument file.
• File has more than one link.
• File is not a regular file.
• File is not in compressed format.
• File was compressed using more than 16 bits.
• There is no space for decompress tables.
• A compressed file is corrupt.

Portability

uncompress is found on many UNIX systems.
The -D option is an extension to traditional implementations of uncompress; the -D and -V options are extensions to the POSIX standard.

Related information
compress, cpio, pack, unpack, zcat

unexpand - Compress spaces into tabs

Format

unexpand [-Ba] [-t tablist] [-W option[option]] ... [file ...]

Description

unexpand replaces blank characters in the data from each file argument with the most efficient use of tabs and spaces. If you do not specify any files, unexpand reads the standard input. The result is sent to standard output.

Backspace characters are preserved. By default, unexpand compresses only leading spaces into tabs; tab stops are set every eight spaces.

Options

unexpand supports the following options:

-B
Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-a
Compresses spaces into tabs wherever the resulting output is shorter, regardless of where the spaces occur in the line.

-t tablist
Specifies tab stops. The numbers in tablist are delimited by blanks or commas. If tablist is a single number, then unexpand places tab stops every tablist positions. If tablist contains multiple numbers, unexpand places tab stops at those specific positions. Multiple numbers in tablist must be in ascending order. This option, like the -a option, compresses spaces to tabs at any appropriate point in the line. If you specify -t, unexpand ignores the presence or absence of -a.

-W option[,...]
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:
**filecodeset=**<i>codeset</i>
Performs text conversion from one code set to another when reading from the file. The coded character set of the file is <i>codeset</i>. <i>codeset</i> can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command <code>iconv -l</code> lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

**pgmcodeset=**<i>codeset</i>
Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is <i>codeset</i>. <i>codeset</i> can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command <code>iconv -l</code> lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

**Examples**

1. To compress spaces in a text file into tabs every 3 positions:
   
   ```
   unexpand -t 3 myTextFile
   ```

2. To compress spaces in a text file containing ASCII characters into tabs, assuming that
   
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file)

   then issue:
   
   ```
   unexpand -a -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 myAsciiFile
   ```

3. To compress spaces in a text file containing EBCDIC characters into tabs, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as UTF-8:

   ```
   unexpand -a -B myMisTaggedFile
   ```
Localization

unexpand uses the following localization variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Environment variables

unexpand uses the following environment variable:

_TEXT_CONV

Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-w option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

Exit values

0
Successful completion

1
Failure due to any of the following reasons:

• An incorrect command-line argument.
• An inability to open the input files.
• The code set is not valid.
• Could not turn off automatic conversion.
• Could not perform requested text conversion.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide, 4.2BSD.

The -B and -W options are extensions of the POSIX standard.

Related information

expand, pr

uniq - Report or filter out repeated lines in a file

Format

uniq [-c|-d|-u] [-B] [-f number1] [-s number2] [-W option[option] ... [input_file [output_file]]
uniq [-c|-d|-u] [-number] [+-number] [-B] [-W option[option] ... [input_file [output_file]]

Description

unique manipulates lines that occur more than once in a file. The file must be sorted, since unique only compares adjacent lines. When you invoke this command with no options, it writes only one copy of each line in input_file to output_file. If you do not specify input_file or you specify -, unique reads the standard input.

Shell command descriptions 773
If you do not specify output_file, uniq uses the standard output. The specified output_file cannot be a FIFO.

Options

-B
Disables the automatic conversion of tagged files. This option is ignored if the filecodeset or pgmcodeset options (-W option) are specified.

-c
Precedes each output line with the number of times that line occurred in the input.

-d
Displays only lines that are repeated (one copy of each line).

-f number1
Ignores the first number1 fields when comparing lines. Blanks separate fields in the input.

-s number2
Ignores the first number2 characters when comparing lines. If you specify both -s and -f, uniq ignores the first number2 characters after the first number1 fields.

-u
Displays only those lines that are not repeated.

-W option[,option]...
Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

filecodeset=codeset
Performs text conversion from one code set to another when reading from the file. The coded character set of the file is codeset. codeset can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If pgmcodeset is specified but filecodeset is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

When specifying values for filecodeset, use the values that Unicode Service supports. For more information about supported code sets, see z/OS Unicode Services User’s Guide and Reference.

pgmcodeset=codeSet
Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is codeSet. codeSet can be a code set name known to the system or a numeric coded character set identifier (CCSID). Note that the command iconv -l lists existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.
If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

You can choose only one of the -c, -d, or -u options.

**-number**
Equivalent to -f number (obsolescent).

**+number**
Equivalent to -s number (obsolescent).

**Examples**
1. The command:
   ```
   uniq
   ```
   is a filter which prints one copy of each different line in its sorted input.

2. The command:
   ```
   uniq -f 2 -s 1
   ```
   compares lines starting with the second character of the third field.

3. The command:
   ```
   uniq -d
   ```
   prints one instance of each repeated line in the input (and omits all unique lines).

4. To print one copy of each different line in a text file containing UTF-8 characters, assuming that
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file)
   then issue:
   ```
   uniq -W filecodeset=UTF-8,pgmcodeset=IBM-1047 myUtf8File
   ```

5. To print only those lines that are not repeated in a text file containing EBCDIC characters, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as ASCII:
   ```
   uniq -u -B myMisTaggedFile
   ```

**Localization**

**unique** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Environment variables**

**unique** uses the following environment variable:
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-w option) is specified. For more information about text conversion, see Appendix L, "Controlling text conversion for z/OS UNIX shell commands," on page 993.

**Exit values**

0

Successful completion.

1

Failure due to any of the following reasons:

- Incorrect command-line option.
- Missing `number` after `-f`.
- Missing or incorrect `number` after `-s`.
- Inability to open the input or output file.
- The code set is not valid.
- Could not turn off automatic conversion.
- Could not perform requested text conversion.

3

Failure while reading the input file:

- Error during system call.
- Input line too long.
- Incorrect character in input.

**Messages**

Possible error messages include:

**Missing character skip count**

You specified `-s` but did not supply a number after the `-s`.

**Missing number of fields to skip**

You specified `-f` but did not supply a number after the `-f`.

**Field skip not a number in string**

In a `-number` or `+number` construct, `number` was not a valid number. This could happen because of a typographical error in entering a - option.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

The -B and -W options are extensions of the POSIX standard. The -number and +number options are considered obsolete.

**Related information**

comm, sort

**unlink - Removes a directory entry**

**Format**

`unlink file`
**Description**

unlink removes a directory entry.

Following the format, *file* specifies the entry to be removed, which can refer to a path name, a hard link, or a symbolic link. If *file* refers to a symbolic link, unlink removes the symbolic link but not any file or directory named by the contents of the symbolic link. If the entry that is unlinked is the last one associated with a file, then the file itself is deleted.

unlink is implemented as a shell built-in.

**Localization**

unlink uses the following localization environment variables:

- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


**Exit values**

0  
Successful completion.

1  
Failure due to any of the following reasons:
- No write permission for the directory that contains the link to be removed.
- Attempting to unlink a file that does not exist.
- Path name specified is a directory.

2  
Failure due to incorrect number of arguments specified.

**Related information**

mv, rm, rmdir

**unmount - Remove a file system from the file hierarchy**

**Format**

```
mount [-R|-f] [-v] [-o normal|drain|immediate|force|reset] [-m] name ...
```

**Description**

The `mount` shell command, which is located in `/usr/sbin`, unmounts file systems.

You must have mount authority before you can issue the `mount` command. See the section on `mounting` file systems in `z/OS UNIX System Services Planning`. 
Options

-f
The list of names to unmount are file system names instead of path names.
This option is mutually exclusive with -R.

-m
Specifies that the name ... parameter can be any file or directory within the file system to be unmounted.

-R
Unmounts the specified file system and all the file systems under it in the file system hierarchy.
This option is mutually exclusive with -f.

-o normal|drain|immediate|force|reset

  normal
  Specifies that if no user is accessing any of the files in the specified file system, the system processes the mount request. Otherwise, the system rejects the mount request. This is the default

  drain
  Specifies that an unmount drain request is to be made. The system will wait for all use of the file system to be ended normally before the unmount request is processed or until another UNMOUNT command is issued.

    unmount -o drain is not supported in a sysplex. If an unmount -o drain is issued in a sysplex, the following behavior is exhibited:
    • If there is no activity in the file system, unmount -o drain performs the unmount operation, but it behaves like an unmount normal.
    • If there is activity in the file system, unmount -o drain returns a Return_value of -1 with Return_code EINVAL and Reason_code JrNotSupInSysplex.

  immediate
  The system immediately unmounts the file system. Any users who are accessing files in the specified file system will receive failing return codes. All data changes to files in the specified file system are saved. If the data changes cannot be saved, the mount request fails.

  force
  Also specifies that the system will unmount the file system immediately. Any users who are accessing files in the specified file system will receive failing return codes. If possible, all data changes to files in the specified file system are saved. If the data changes to the files cannot be saved, the unmount request continues and the data is lost.

  The -f filesystem_name option must also be used.

  An unmount -o immediate request must be issued before you can request an unmount -o force of a file system. Otherwise, unmount -o force will fail.

reset
  A reset request stops a previous unmount -o drain request.

  Restriction: unmount -o reset is not supported in a sysplex.

-v
Lists all file systems that are unmounted.

name ... specifies the path name of the mount point directory to use when locating the file system to be unmounted or the name of the file system to be unmounted. If the -m option is used, the name can be for any file or directory within that file system.
Examples

1. To unmount a file system that is mounted on /u/wjs, issue:

   unmount /u/wjs

2. The output of mount -q can be used for the input of `mount`. For example:

   mount -q /ict/hfsdir
   can be used as input:
   unmount $(mount -q /ict/hfsdir)

3. To unmount a file system that contains the file or directory /u/wjs, using the `-m` option to specify the directory:

   unmount -m /u/wjs

4. To unmount a file system that contains the file or directory /u along with all other file systems mounted over or below that file system, using the `-m` option to specify the directory:

   unmount -R -m /u

Usage notes for `unmount`

1. When the `mount` command completes, the unmount operation is complete on the owner system, and on the system where the unmount command was issued from, but not necessarily on the other systems in the sysplex. On those systems, the unmount operation completes asynchronously.

2. If a path name is used, that path name must be accessible or you will have to use the `unmount -f` option using the file system name. If the path name is not accessible, use the console command DOMVS,F to obtain the file system name if it is not known.

3. Because the path name for `mount` is a node, symbolic links cannot be followed unless a trailing slash is added to the symbolic link name. For example, if /etc has been converted into a symbolic link, issuing unmount -R /etc without the trailing slash will result in trying to unmount -R /etc -> $SYSNAME/etc. Depending on the security access for the symbolic link, RACF errors might occur. However, if you specify unmount -R /etc/ with the trailing slash, the symbolic link will be followed and RACF will determine the access from the file being linked to.

4. When the `-m` option is specified, the `mount` shell command operates on the path name and its associated file system. If the path name does not have a file system mounted on it, the associated file system is the one that contains the path. For example:

   mkdir /mega
   mount -f 'posix.hfs.mega' /mega
   mkdir /mega/wellie0
   mount -f 'posix.hfs.wellie0' /mega/wellie0
   then:
   unmount -R /mega

   unmounts the file system that is mounted at /mega/wellie0 and /mega. If you enter the same command again:

   unmount -R /mega

   The unmount fails because there is no file system that is mounted at /mega.
If you then issue the command with the \texttt{-m} options

\begin{verbatim}
unmount -R -m /mega
\end{verbatim}

the unmount operation attempts to unmount the file system that contains the \texttt{/mega} directory (in this case, the root) and any other file systems that are mounted on the root.

**Exit values**

0  
Successful completion.

**Related information**

\texttt{chmount, mount}

\section*{unpack - Decode Huffman-packed files}

**Format**

\texttt{unpack file...}

**Note:** The unpack utility is fully supported for compatibility with older UNIX systems. However, it is recommended that the \texttt{uncompress} utility be used instead because it might provide greater functionality and is considered the standard for portable UNIX applications as defined by POSIX.2 IEEE standard 1003.2-1992.

**Description**

\texttt{unpack} uncompresses files compressed by \texttt{pack}, using a Huffman minimal redundancy code. By default, \texttt{unpack} looks for \texttt{file} with a \texttt{.z} extension. It places the decompressed output in a file with the same name, but without the extension. The owner, permissions, and times of last access and last modification are also preserved. Packed files can be identified by \texttt{file}. You can use \texttt{pcat} to view packed text files without unpacking them in place.

\texttt{unpack} does not unpack a file if:

\begin{itemize}
  \item The file name is too long after the \texttt{.z} is removed.
  \item The input file cannot be opened.
  \item An existing file has the same name as the output file.
  \item The output file can't be created.
  \item The input file doesn't appear to have been created by pack.
\end{itemize}

**Localization**

\texttt{unpack} uses the following localization environment variables:

\begin{itemize}
  \item \texttt{LANG}
  \item \texttt{LC_ALL}
  \item \texttt{LC_COLLATE}
  \item \texttt{LC_CTYPE}
  \item \texttt{LC_MESSAGES}
  \item \texttt{NLSPATH}
\end{itemize}

**Files**

unpack uses the following file:

`pk$*`

Temporary copy of input file. (You might see this in the current directory if unpack is interrupted.) The file is in the same directory as the file being unpacked.

**Exit values**

0

Successful completion

n

Indicates that files might not be unpacked properly. For example, if three out of six files might not be unpacked properly, the exit status is 3.

Possible reasons for failure include:

- Unknown command-line option.
- Error creating a name for a temporary file.
- Error opening an input file or a temporary file.
- Error writing to a temporary file.
- Inability to rename a temporary file.
- Inability to restore the modification time on a packed file.
- Input file was not packed.
- A packed file is corrupted.

**Messages**

Possible error messages include:

file: Not a packed file

pack did not process the file. In this case, the file is not changed.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

**Related information**

file, pack, pcat

---

**unset - Unset values and attributes of variables and functions**

**Format**

unset name ...

unset -fv name ...

tcsh shell: unset pattern

**Description**

Calling unset with no options removes the value and attributes of each variable or function name.
unset

In the tcsh shell, unset removes all variables whose names match pattern, unless they are read-only. For example:

```
unset *
```

which we strongly recommend you do not do, will remove all variables unless they are read-only. It is not an error for nothing to be unset. For more information, see “tcsh: Invoke a C shell” on page 671.

**Options**

- `-f`
  - Removes the value and attributes of each function *name*.

- `-v`
  - Removes the attribute and value of the variable *name*. This option is the default if no options are specified.

`unset` cannot remove names that have been set read-only.

**Usage notes**

`unset` is a special built-in shell command.

**Localization**

`unset` uses the following localization environment variables:

- LANG
- LC_ALL
- LC_MESSAGES
- NLSPATH


**Exit values**

- **0**
  - Successful completion.

- **1**
  - Failure due to an incorrect command-line option.

- **2**
  - Failure due to an incorrect command-line argument.

Otherwise, `unset` returns the number of specified *names* that are incorrect, not currently set, or read-only.

**Messages**

Possible error messages include:

- **name readonly variable**
  - The given *name* cannot be deleted because it has been marked read-only.

**Portability**

POSIX.2, X/Open Portability Guide.

**Related information**

`readonly, sh, tcsh`
uptime - Report how long the system has been running

Format

uptime

Description

uptime gives a one-line display of the following information:
• The current time.
• How long the system has been running.
• Number of users who are currently logged into z/OS UNIX and the system load averages for the past 1, 5, and 15 minutes. Load averages are not supported on z/OS UNIX, and are displayed as 0.00.

Files

uptime uses the following file:

/etc/utmpx
  The current login status file.

Localization

uptime uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH


Exit values

0
  Successful completion.
1
  Invalid /etc/utmpx file
  Command syntax error.

uucc - Compile UUCP configuration files

Format

uucc

Description

uucc reads the contents of the uucp configuration files and compiles them into a single configuration file called /usr/lib/uucp/config. The configuration files are:
• Systems
• Devices
• Dialers
uucico daemon

- Dialcodes
- Permissions

Because uucc expects these text files to be in the current working directory, you need to change the directory (with the cd command) to /usr/lib/uucp before issuing uucc. See the section on creating or editing UUCP configuration files in *z/OS UNIX System Services Planning* for more information.

**Files**

uucc uses the following files:

- **/usr/lib/uucp/Systems**
  - Contains a list of remote systems and the methods for connecting with them.

- **/usr/lib/uucp/Devices**
  - Describes the physical and logical connections listed in the Systems file.

- **/usr/lib/uucp/Dialers**
  - Contains dialing information for the modems and dialers listed in the Devices file.

- **/usr/lib/uucp/Dialcodes**
  - Contains abbreviations that can be used in the phone numbers specified in Systems.

- **/usr/lib/uucp/Permissions**
  - Defines the commands and areas of the file system that remote sites can access on your system.

- **/usr/lib/uucp/config**
  - Contains the previous information compiled into one file for use by the uucp utilities.

**Exit values**

- **0**
  - Successful completion.

- **1**
  - Failure due to any of the following reasons:
    - Inability to open file.
    - Insufficient memory.
    - Ctrl-C interrupt.

**Related information**

uucp

**uucico daemon: Process UUCP file transfer requests**

**Format**

```
uucico [-f] [-g grade] [-r0|-r1] [-s system] [-x type]
```

**Description**

The uucico daemon processes file transfer requests that were queued by uucp and uux. It establishes the connection with remote sites and manages the transfer of data between the local and remote sites as specified by the queued uucp or uux job.

uucico is automatically invoked after the uucp or uux command completes (unless the -r option was specified on the uucp or uux command). To process requests that cannot be successfully completed at the time the uucp or uux command was executed and to initiate transfers from remote sites, the traditional approach is to use cron to start uucico at regular intervals. (See *z/OS UNIX System Services Planning* for more information.)
Planning for more information about using cron to start uucico. It contains information about creating crontab entries.)

**uucico** has two modes: slave mode and master mode.

- **In slave mode,** **uucico** receives requests from the remote site. The `-r0` option (the default option) starts **uucico** in slave mode. **uucico** is typically started in slave mode by either the uucpd daemon (for remote connections via TCP/IP) or as the login shell for special UUCP user IDs that can be logged onto via serial connections. See **uucpd** and **uucp** for more information.

- **In master mode,** **uucico** processes requests from the local site; the `-r1` and `-s` options start **uucico** in master mode. **uucico** is typically started in master mode via **cron**. **uucp** and **uux** also invoke **uucico** in master mode by default.

If **uucico** cannot contact a remote system, it does not allow itself to run again until a specified amount of time has passed. You can specify how long the daemon should wait before trying to call each system again by setting a parameter in the Permissions file. For information about the permissions file, see z/OS UNIX System Services Planning.

If **uucico** receives a SIGQUIT, SIGTERM or SIGPIPE signal, it ends any current conversation with a remote site and exits.

**Options**

- **-f**
  
  Ignores the required wait period for all remote systems and makes calls as requested.

- **-g grade**
  
  Processes outgoing work only if it is designated priority `grade` or better. `grade` is a number (0-9) or letter (A-Z, a-z), where 0 is the highest priority and z is the lowest.

- **-r0 | -r1**
  
  Specifies the mode for **uucico** to use. `-r0` (the default) specifies slave mode; `-r1` specifies master mode. If you want **uucico** to call a remote system (master mode), specify `-r1`.

- **-s system**
  
  Calls the remote system. By default, **uucico** calls all defined systems.

- **-x type**
  
  Turns on debugging. `type` is a number indicating the level of detail. 0 is the least detail and 9 is the most detail. The debugging output is written to stderr if **uucico** is run in the foreground, or to `/usr/spool/uucp/LOGFILE` if **uucico** is run in the background by **uucpd** or by a remote **uucico** logging into a UUCP user ID.

  The LOGFILE must be monitored so that it does not fill up your file system.

**Examples**

To call the remote site west, with debugging output sent to stdout:

```plaintext
uucico -r1 -x 9 -s west
```

**Files**

**uucico** uses the following files:

- **/usr/lib/uucp/config**
  
  UUCP configuration file. See **uucc**.

- **/usr/spool/uucp/LOGFILE**
  
  UUCP debug file.

- **/usr/spool/locks**
  
  The directory containing the lock files created by **uucico**.

- **/usr/spool/uucp/.Status**
  
  UUCP status file.
**Exit values**

0  Successful completion.

1  Failure due to any of the following reasons:
   - Unknown command-line option.
   - Not running setuid uucp.
   - Argument list too long.
   - Unable to open log file.
   - CTRL-C interrupt.

**Portability**

X/Open Portability Guide.

**Related information**

uucc, uucp, uulog, uux, uuxqt

---

**uucp - Copy files between remote UUCP systems**

**Format**

```
uucp [-Ccfdjm] [-g grade] [-n user] [-x debug_level] [site1!] file1 [site2!] file2
uucp [-Ccfdjm] [-g grade] [-n user] [-x debug_level] file... [site2!] directory
```

**Description**

*uucp* copies a source file or files from one site to a target on another site. The source can be a file or group of files specified by metacharacters. The source cannot be a directory. The target can be a corresponding file name or directory.

File names given to *uucp* have the form:

```
[site!] path
```

or

```
[site1![site2!]... path
```

where *site* names the remote site. If a site is not specified, *pathname* is a path name on your machine. *site* must be on the list of site names that *uucp* knows about. Use uuname to list sites that are known to *uucp*.

You can also specify multiple site names as a way of sending files to a site that your system does not have a direct connection to. File names that contain multiple site names are called *multinode or multihop* names. When processing a *uucp* request involving multihop names, only the destination name can be a multihop name. The source file name cannot be a multihop name.

The path names can have one of these forms:

- A full path name.
- A path name preceded by `~name/`, where `~name` is replaced on the specified site by the login directory of user `name`.
- A path name preceded by `~/`, where `~/` is replaced on the specified site with the name of the public UUCP directory.
- A file name or prefix name containing the current directory on your machine as a prefix.
Destination path names cannot begin with exactly two slashes, which indicate an MVS file name. If the target is a directory, you must append / to the end of the path name to ensure that it is not treated as a file. If the / is not appended to a directory name, then the name is treated as a file name and multiple copies to that command will behave like the cp command. That is, each subsequent copy will overlay the previous one.

Path names can contain the shell metacharacters ?, *, and [ ]. The character ~ also has a special meaning, as previously described. The appropriate site expands these characters after encountering them. If the destination file is a multihop name, then the source file cannot contain shell metacharacters because uucp uses uux to handle multihop requests, and uux does not allow shell metacharacters in names. Be careful when using metacharacters, because expansions on other sites may occur in unforeseen ways. For more information about metacharacters and their expansion by the shell, see sh.

Options

-C
Copies named files to the spool directory for transfer. If both this option and the -c option are given, this option takes precedence. This option is useful if you will be changing the file after running the uucp command and want to send the version of the file before you changed it.

-c
Does not copy files to the spool directory for transfer. This is the default.

-d
Makes all necessary intermediate directories to complete file transfer. This is the default.

-f
Does not make intermediate directories. If -f is specified with the -d option, -f takes precedence.

-g grade
Sets the priority of this job to grade. It is a number (0-9) or letter (A-Z, a-z), where 0 is the highest priority and z is the lowest.

-j
Passes the UUCP job ID number to standard output; this job ID can be used with uustat to determine the job's status or to terminate it. If uucp generates several job requests and several job IDs, only the last one appears.

-m
Sends mail notifying you when the copy finishes. The default is to send mail only if an error occurs that prevents the copy from being made.

-n user
Notifies the user at the destination site when a file you sent to the destination site arrives. This option has no effect when you use uucp to get files from the remote system.

-r
Queues the job to be processed later. Do not start uucico to begin transferring the file.

-x debug_level
Sets the verbosity of the debugging information to the specified debug level, which is a number, 0 or greater. Level 0 provides tersed messages while level 9 provides verbose messages. Values greater than 9 give no additional information. The default level is 0.

Options are not passed on to remote sites when the destination of your uucp command is a multihop name. For this uucp command:

```
uucp -mf file1 site1!site2!/file1
```

the -m and -f options are ignored. For multihop, uucp creates a uux request to run a uucp command at the next site (site1 in our example). But because site1 can be any system that supports uucp, it is possible that this particular system might not support the same options that are supported by uucp. For that reason, options are not passed to the uucp command to be run at site1.

To summarize the restrictions when using multihop destination names:
Options are not passed.
Shell metacharacters cannot be used in source file names.

Examples
1. To copy the file /notes/memo from your site to a file named minutes in the public UUCP directory on a site named south:

```bash
uucp /notes/memo south! ~/minutes
```

2. You can also copy files locally. To copy the file resume.txt on your site to the file /business/resumes/november on your site:

```bash
uucp resume.txt /business/resumes/november
```

You must have read permission on the current directory. If the directories business/resumes do not exist, they are created if you have write permission in /

3. To copy the file index from the public UUCP directory on north to the current directory on the local site:

```bash
uucp north! ~/index
```

You must have write permission on the current directory.

4. To copy the file index from the public UUCP directory on south to the subdirectory south/records in the public UUCP directory on the current site:

```bash
uucp -f -m south! ~/index ' ~/south/records/
```

You must protect the tilde so the shell does not expand them to the user’s home directory. If the subdirectory south/records does not exist, the file copy fails. Mail is sent to you when the transfer is completed successfully.

5. You want to copy a file from your system to the site named east. Your system does not have a connection to east, but you do have a connection to north, and north has a connection to east:

```bash
uucp memo north!east! ~/memo
```

6. You want to use shell metacharacters to specify the files to be transferred to a remote site.

In this command, the source path name is expanded by the shell. The `uucp` command succeeds if there is at least one file that matches the name specification:

```bash
uucp /mystuff/file?.[ab&]* remote!/tmp/
```

In this command, the source path name is not expanded by the shell, because it cannot find any matching file. The ‘!’ is not allowed, because `uucp` interprets all ‘!’ characters as delimiting system names.

```bash
uucp remote!/tmp/file?.[!b]* /mystuff/
```

Environment variables
`uucp` uses the following environment variable:

TZ
Sets the time zone used with date and time messages

Localization
`uucp` uses the following localization environment variables:

• LANG
Files
uucp uses the following files:

/usr/lib/uucp/config
  UUCP configuration file generated by uucc.

/usr/spool/uucp/LOGFILE
  Log file for uucp and other UUCP utilities.

/usr/spool/uucppublic
  Public UUCP directory.

/usr/spool/uucp/.Sequence/sitename
  Sequence files, one for each remote site.

Usage notes
uucp does not convert files to or from EBCDIC. If a text file is sent from an ASCII system to an MVS system, it must be converted to EBCDIC after its arrival. Similarly, if an EBCDIC text file is sent to an ASCII system, the file is not automatically converted to ASCII. The receiving user must convert the file to ASCII.

Exit values

0
  Successful completion.

1
  Failure due to any of the following reasons:
  • Inability to open log file.
  • Insufficient memory.
  • Ctrl-C interrupt.

2
  Unknown command-line option.

uucp can also have partial failures, where a file is inaccessible or a host could not be determined. uucp returns the 1 exit value and logs the partial failure in the log file /usr/spool/uucp/LOGFILE. Files that were accessible or had a known host are still queued for transfer.

Portability
X/Open Portability Guide, UNIX systems.
The -g option is an extension to the POSIX standard.

Related information
uucc, uucico, uulog, uux
uucpd daemon - Invoke uucico for TCP/IP connections from remote UUCP systems

Format

uucpd [-l seconds] [-x debug_level]

Description

Warning: The uucpd daemon is by nature unsecure because it requires user authentication that uses cleartext sockets. A remote attacker can exploit this vulnerability to sniff logins and passwords. Do not use this daemon if possible.

The uucpd program allows remote uucico programs to communicate with local uucico in order to perform file transfers via TCP/IP connections. inetd starts uucpd when the remote uucico connects to the UUCP port. uucpd manages the login sequence with the remote uucico. After successful login, it then starts uucico to complete the transfer.

In order for inetd to start uucpd, the inetd configuration file (for example, inetd.conf) must contain a UUCP entry such as the following:

```
uucp stream tcp nowait OMVSKERN /usr/sbin/uucpd uucpd -l0\debug_level
```

Options

- `-l seconds`
  Sets the login timeout value in seconds. When `seconds` are specified as zero, the login will wait without timing out.

- `-x`
  Invoke uucpd and uucico with the -x option. `debug_level` indicates the level of detail (0 has the least detail and 9 has the most detail). The uucpd login sequence debug output is written to a file in TMPDIR with a file name beginning with uucpd and followed by randomly generated characters. The uucico debug output is written to the UUCP log file.

  **Note:** If you are using the -x option, monitor the UUCP logfile so that it does not become too large and fill up the file system.

  The Permissions file provides an alternative method for setting debug for connections on a system-by-system basis. See the section on permissions file in z/OS UNIX System Services Planning for more information.

Usage notes

uucpd is not affected by the locale information that is specified in locale-related environment variables.

Exit values

- `0`
  Successful completion.

- `1`
  Failure to establish a connection with the remote system.

- `>1`
  uucico failure.

Portability

X/Open Portability GuideX/Open Portability Guide.
uudecode - Decode a transmitted binary file

Format

\texttt{uudecode \[\textit{-o outfile} \] \textit{infile}...}

Description

\texttt{uudecode} decodes data that was encoded by \texttt{uuencode}. If an \textit{infile} is specified on the command line, \texttt{uudecode} decodes that file; if no \textit{infile} is specified on the command line, input is read from the standard input. Output is written to the file name that was specified when the file was encoded. When the \texttt{-o} option is specified, the file name that was specified when the file was encoded is overridden by the \textit{outfile} operand. See \texttt{uuencode} for more information.

\texttt{uudecode} automatically strips off any leading and trailing lines added by mailers.

For a summary of the UNIX03 changes to this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Options

\texttt{-o outfile}  
A path name of a file is used instead of any path name that is contained in the input data. Specifying an \textit{outfile} option-argument of \texttt{/dev/stdout} indicates standard output is used.

Localization

\texttt{uudecode} uses the following localization environment variables:

\begin{itemize}
  \item LANG
  \item LC_ALL
  \item LC_CTYPE
  \item LC_MESSAGES
  \item NLSPATH
\end{itemize}


Usage notes

If the path name of the file to be produced exists and is writable, the file is overwritten. If it exists, but is not writable by the user, \texttt{uudecode} will end with an error.

Exit values

\begin{itemize}
  \item \texttt{0}  \hspace{1cm} \text{Successful completion.}
  \item \texttt{1}  \hspace{1cm} \text{Failure due to any of the following reasons:}
    \begin{itemize}
      \item Inability to open the input file.
      \item Missing begin line in the input file.
      \item Inability to create the output file.
      \item Missing end line in the input file
      \item A file that is too short.
    \end{itemize}
\end{itemize}
uuencode

2
Failure because of an incorrect command-line option.

Portability
POSIX.2 User Portability Extension, X/Open Portability Guide. Generally found on most UNIX systems.

Related information
uuencode

uuencode - Encode a file for safe transmission

Format
uuencode [-m] [infile] remotefile

Description
When files are transmitted over a network or over phone lines, nonprintable characters (for example, control characters) might be interpreted as commands, telling the network to do something. In general, therefore, it is not safe to transmit a file if it contains nonprintable characters.

uuencode Converts a binary file into a special code that consists entirely of printable characters from the POSIX portable character set. A file encoded in this way is generally safe for transmission over networks and phone lines. uuencode is often used to send binary files through electronic mail.

If an infile is specified on the uuencode command line, uuencode reads that file as input. Otherwise, it reads the standard input. uuencode always writes the encoded result to the standard output. The encoded version of the data is about 35% larger than the original. If the size is a problem, you can shrink the file with compress before encoding it. The recipient must decode it and then uncompress it.

The remotefile command-line argument is the name that the file should be given after it has been transmitted to its destination. Specifying a remotefile operand of /dev/stdout indicates that uudecode is to use standard output. When the file reaches its destination, uudecode can be used to convert the encoded data into its original form. The first line of the encoded file records the file's access permission bits and the remotefile argument.

Because the encoded file consists entirely of printable characters, you can use a text editor to edit the file. Of course, the only things you are likely to edit are the name of the original file or the name of the remote file.

For a summary of the UNIX03 changes to this command, see Appendix N, “Shell commands changed for UNIX03,” on page 1005.

Options
-m
Encode the output by using the MIME Base64 algorithm. If -m> is not specified, the historical algorithm is used.

Examples
This command encodes the file long_name.tar.Z so it decodes with the name arc.trz and redirects the output to arc.ue:

```
uuencode long_name.tar.Z arc.trz > arc.ue
```
Localization

uuencode uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLS PATH


Exit values

0
   Successful completion.

1
   Failure because of an incorrect command-line option, or a missing command-line argument.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide. Generally found on most UNIX systems.

Related information

uudecode

**uulog - Display log information about UUCP events**

Format

```
uulog [-s site]
```

Note: The uulog utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, do not use this utility for applications that are intended to be portable to other UNIX-branded systems.

Description

uulog displays information about UUCP events, such as file transfers and remote command execution. It also displays the most recent debug output to the log. In order to use uulog, you must have permission to read the file /usr/spool/uucp/LOGFILE.

The format of the display is:

```
user ID   local_site   date/time   messagetext
```

where:

**user ID**

Login ID of the user who requested the file transfer or requested that the command be run. Entries that are created by uuxqt or by programs that are spawned by uuxqt have the ID uucp.

**local site**

Name of the local site.

**date/time**

Date and time of the event in the form (mm/dd-hh:mm).
**uulog**

**message text**
Text of the log entry. The message text depends on the event being recorded; most entries are self-explanatory.

**Options**
If you do not specify an option, `uulog` displays the debug information for the last conversation that failed.

`-s site`
Displays information about UUCP events for this site.

**Environment variables**
`uulog` uses the following environment variable:

`TZ`
Sets the time zone that is used with date and time messages.

**Localization**
`uulog` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_LCTIME`
- `NLSPATH`


**Files**
`uulog` uses the following files:

- `/usr/lib/uucp/config` UUCP configuration file. (See `uucc`.)
- `/usr/spool/uucp` UUCP spool directory.
- `/usr/spool/uucp/LOGFILE` Log file for `uulog` and other UUCP utilities.

**Exit values**
0  
Successful completion.

1  
Failure due to any of the following reasons:

- LOGFILE could not be opened.
- Could not lock LOGFILE.

2  
Unknown command-line option.

**Portability**
X/Open Portability Guide, UNIX systems.
Related information
uucc, uucp, uux

uname - Display list of remote UUCP systems

Format
uname [-l]

Note: The uname utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, do not use this utility for applications that are intended to be portable to other UNIX-branded systems.

Description
uname displays a list of all remote systems that are known to UUCP. Systems are listed in the order they are entered in /usr/lib/uucp/Systems. To display only the local system name, use the -l option.

Options
- l
  Displays the local system name.

Localization
uname uses the following localization environment variables:
• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_LCTIME
• NLSPATH

Files
uname uses the following file:
/usr/lib/uucp/config
  UUCP configuration file. See uucc.

Exit values
0
  Successful completion.
1
  Failure due to any of the following reasons:
  • Argument list too long.
  • Inability to open log file.
  • Insufficient memory.
  • Ctrl-C interrupt.
2
  Unknown command-line option.
**unpick - Manage files sent by uuto and uucp**

**Format**

```
unpick [-s system]
```

The `unpick` utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, this utility should be avoided for applications that are intended to be portable to other UNIX-branded systems.

**Description**

`unpick` is an interactive shell script that is used to manage files in the UUCP public receive directory that were sent to you using the `uuto` command. Only those files in the receive directory are managed.

For each file or directory entry found, `unpick` prompts you with one of the following messages, depending on the type of the entry:

```
from system:  file name ?
from system:  dir name ?
```

where `system` is the name of the system that sent the file or directory, and `name` is the name of the file or directory.

To tell `unpick` how to handle an entry, issue one of the following commands:

- **ENTER**
  - Skips this entry and go to the next one.
- *****
  - Display the `unpick` command summary.
- **d**
  - Deletes the specified entry.
- **m [target]**
  - Moves the entry to the named target directory or file. If the target does not specify an absolute path name or no directory, the path name is assumed to be relative to the current directory. If no directory is given, `unpick` assumes the current directory.
- **a [dir]**
  - Moves all files from `system` to the target directory `dir`.
- **p**
  - Prints the contents of the entry to standard output. If the entry is a directory, `p` lists the files in the directory.
- **q**
  - Quits `unpick`.
- **CTRL-D**
  - Quits `unpick`.
- **!command**
  - Escapes to the shell in order to perform `command`.

The tilde (~) does not stand for the public UUCP directory in path names that are specified inside `unpick`. It is interpreted by the command shell that is being used.
Options
-s system
   Displays only files from the system system.

Localization
uupick uses the following localization environment variables:
   • LANG
   • LC_ALL
   • LC_CTYPE
   • LC_MESSAGES
   • LC_LCTIME
   • NLSPATH

Files
uupick uses the following files:
/usr/lib/uucp/config
   The UUCP configuration file, which contains the list of known systems as well as the location of the public UUCP library. See uucc.
/usr/spool/uucppublic
   /usr/spool/uucpublic file. The default value for the public UUCP directory. The public UUCP directory is always the home directory of the user uucp as defined in the user database.
/usr/spool/uucppublic/receive/user/system
   When files are sent to your system using uuto, they are placed into /usr/spool/uucppublic/receive/user/system, where user is your login name and system is the name of the system that sent the files.

Usage notes
1. uupick does not convert files from EBCDIC. If you receive a file from an ASCII system, you will need to convert it to EBCDIC.
2. When moving files, uupick does not check for files of the same name in the destination directory. These files are overwritten.
3. uupick is a shell script.

Portability
X/Open Portability Guide.

Related information
uuto

uustat - Display status of pending UUCP transfers

Format
uustat [-j jobid | -k jobid | -r jobid]
uustat [-m]
**uustat**

**uustat [-q]**
**uustat [-s site] [-u user]**
**uustat [-a [-o number] [-y number]]**

**Description**

`uustat` displays reports on the progress of pending UUCP transfers. You can display the status of transfers for a particular job ID or user ID. `uustat` can also stop or restart jobs in the queue.

If you do not specify any options, it displays the status of all UUCP requests for all sites that are made by the current user.

**Options**

- **-a**
  Displays the jobs queued for all users instead of only the jobs for the user issuing the command.

- **-j jobid**
  Displays the status of the specified job.

- **-k jobid**
  Stops the UUCP job identified by `jobid`. `uustat` can display the job ID of a job in the queue, when used with one of the other options. You cannot use this option with the `-q` or `-r` options.

- **-o number**
  Displays the jobs that are older than `number` hours.

- **-q**
  Displays the latest conversation status and times tried for all sites that recently had errors, as well as a count of the jobs queued. You cannot use this option with the `-k` or `-r` options.

- **-m**
  Displays the latest conversation status and times tried for all sites, as well as a count of the jobs queued. You cannot use this option with the `-k` or `-r` options.

- **-r jobid**
  Restarts the UUCP request specified by `jobid`. This option updates the timestamp on the file, making the request appear recent. It cannot restart jobs that have been stopped with the `-k` option. You cannot use this option with the `-k` or `-q` options.

- **-s site**
  Displays the status of all UUCP transfers requested for `site`.

- **-u user**
  Displays the status of all UUCP transfers requested by `user`.

- **-y number**
  Displays the jobs that are younger than `number` hours.

**Output**

`uustat` uses a variety of output formats, depending on the options that were specified.

If you do not specify an option, or if you specify the `-s` and `-u` options, the output is in this format, one line to every request within a work file:

```
job ID mo/dy-hh:mm rtype site user information
```

The following list explains the fields:

- **job ID**
  Identifies the job. If a job contains more than one request, subsequent requests are displayed below the first, without a job ID.

- **mo/dy-hh:mm**
  Time of the request.
**rtype**
The request type, either S (for send) or R (for receive).

**site**
The name of the remote site.

**user**
The name of the user who requested the job.

**information**
Describes the request. The format depends on the type of request.

For a send request, **information** has the format:

```
size  filename
```

where `size` is the size in bytes of the file to be sent and `filename` is either the absolute path name on your site, or the UNIX-style filename relative to your spool directory for the remote site.

For a receive request, **information** has the format:

```
filename
```

For a remote execution request (such as a request produced by `mailx`, the command to be run is displayed after any data files associated with it.

For the `-q` and `-m` options, the output is in this format:

```
site  transfersC (age)   commandsX(age)   status retry
```

where:

**site**
Remote site name.

**transfersC(age)**
Number of file transfer jobs pending; if any are over one day old, the age in days of the oldest job is given in parentheses.

**commandsX(age)**
Number of pending command requests that have been received; if any are over one day old, the age in days of the oldest job is given in parentheses.

**status**
Time and result of the last attempt to call this site. The **status** field shows the status of attempts made by this system to connect to other systems. When other systems call this system, this field is not updated.

**retry**
Time to the next connection attempt in *hours:minutes* and the current retry count. The **retry** field is displayed only between retry attempts.

For the `-k` and `-r` options, `uustat` displays a message that tells you if the attempt to stop or restart a job was successful.

**Examples**

1. To display all waiting UUCP requests:

   ```
   uustat
   ```

2. To display all jobs that are waiting for remote site `east`:

   ```
   uustat -s east
   ```

3. To stop the UUCP job that is associated with job ID `westn0003`:

   ```
   uustat -k westn0003
   ```
Environment variables
uustat uses the following environment variable:

TZ
Sets the time zone used with date and time information.
Appendix I, “Format of the TZ environment variable,” on page 987 explains how to set the local time zone with the TZ environment variable.

Localization
uustat uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• NLSPATH

Files
uustat uses the following files:

/usr/lib/uucp/config
UUCP configuration file.

/usr/spool/uucp
UUCP spool directory, containing site-specific subdirectories and information files.

/usr/spool/uucp/site
Subdirectory containing queued job requests, work files, data files, and execution files for the UUCP host site.

/usr/spool/uucp/.Status/site
Status file for the remote UUCP host site. uustat queries the status file with the -q option.

Exit values

0
Successful completion.

1
Failure due to any of the following reasons:

• Argument list too long.
• Unable to open log file.
• Log files.
• Insufficient memory.
• CTRL-C interrupt.

2
Unknown command-line option.

Portability
X/Open Portability Guide, UNIX systems.

Related information
uucp, uulog, uxqt
uuto - Copy files to users on remote UUCP systems

Format

`uuto [-mp] file ... destination`

The `uuto` utility is fully supported for compatibility with older UNIX systems. However, because it is no longer supported by POSIX.2 IEEE standard 1003.2-1992, do not use this utility for applications that are intended to be portable to other UNIX systems.

Description

`uuto` is a simplified method of using uucp to copy a file, or files, to a user on another system. `file` is a file, or files, on your system. The destination has the following form:

```
system!user
```

where `system` is a system known to uucp and `user` is the login name of a user on the remote system. You can use `uuname` to list the names of the remote system known to uucp. Make sure to enter the user name in the correct case. Otherwise, the recipient will not be able to use `uuto` to receive the files that you sent.

`uuto` sends files to the UUCP public directory on the remote system. In particular, the files are sent to the directory:

```
pubdir/receive/user/sendsystem
```

where `pubdir` is the UUCP public directory, `user` is the user's name that is specified in the destination, and `sendsystem` is the name of the sending system.

The recipient is notified by mail when the files arrive. If several files are sent, the recipient is notified when the last file arrives. Depending on the nature of the remote system, the recipient can move files from this directory by using the `uupick` utility or by using the system copy commands.

Options

- `-m`
  Sends the user a note when the copy is completed.

- `-p`
  Places files in spool directory before transfer to remote system.

Localization

`uuto` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`
- `LC_LCTIME`
- `NLSPATH`


Files

`uuto` uses the following files:
/usr/lib/uucp/config
The configuration file for UUCP contains the list of systems that uucp knows about. This configuration file is compiled from a number of text configurations by using the uucc utility. (See uucc for more information.)

/usr/spool/uucppublic
The public UUCP directory.

Usage notes
1. uuto is a shell script.
2. uuto does not convert files to ASCII. If you use uuto to send a file to an ASCII system, it must be converted to ASCII after it has been sent.

Portability
X/Open Portability Guide, UNIX systems.

Related information
uucc, uulog, uupick, uustat, uux

uux - Request command execution on remote UUCP systems

Format
uux [-bCjnprz] [-g grade] [-x debug_level] [site!] commandstring

Description
uux specifies that commandstring be run on another site. If files required to run the command are on different sites, uux generates the UUCP requests to gather the files together on one site, runs the command, and sends the standard output of the command to a file on a specified site.

commandstring is any valid command for the remote site, with arguments, except that the command and any file names can specify a site in the UUCP manner:

```
site1!command site2!file1
```

where site1 is the name of the site where the command is to be run, and site2 is the name of the site where file1 is.

- If you do not specify any site names, then the command and any files are assumed to reside at your site.
- If you specify a site for the command, but not for the files, then the files are assumed to reside on the same site that is named for the command.
- If you specify a site for some of the files, then those files without a site name are assumed to reside on the site named.

site must be a valid site name, as listed by the uuname command. Specifying multiple site names, such as site1!site2!command or site1!site2!file is not allowed for uux

Pipes of commands are valid, but only the first command in a pipeline can have a site name. All other commands in the pipeline take place on the site that is specified for the first command.

File names can have one of these forms:
- A full path name.
- A path name that is preceded by ~name/, where ~name is replaced on the specified site by the login directory of user name.
• A path name preceded by ~/ where ~/ is replaced on the specified site with the name of the public UUCP directory.

• A file name or prefix name containing the current directory on your machine as a prefix.

Unlike arguments to uucp, path names cannot contain the shell metacharacters ?, *, and [ ].

Nonlocal file names must be unique within the command, or the command fails. This is because nonlocal files are copied to a working directory on the remote site; if the filenames are not unique, one overwrites another.

If the command fails, you are notified by electronic mail.

**Options**

- **-b**
  Mails input back to the user. The contents of stdin are sent back to the user if the command fails.

- **-C**
  Copies named files to the spool directory for transfer. If both this option and the –c option are given, this option takes precedence. This option is useful if you will be making changes to the file after running the uux command and want to send the version of the file before you changed it.

- **-c**
  Does not copy files to the spool directory for transfer. This is the default.

- **-g grade**
  Sets the priority of the job to grade. It is a number (0–9) or a letter (A–Z, a–z), where 0 is the highest priority and z is the lowest.

- **-j**
  Passes the UUCP job ID number to standard output. This job ID can be used with uustat to determine the job’s status or to terminate it. If the uux request generates several job IDs, only the last is shown.

- **-n**
  Does not send mail if the command fails.

- **-p**
  Uses standard input of uux as the standard input for the specified command. The input is stored in a temporary file that is passed to the command when it runs.

- **-r**
  Queues the job to be processed later. Do not start uucico to begin transferring the file.

- **-x debug_level**
  Sets the verbosity of the debugging information to debug_level, which is a number that is 0 or greater. Level 0 provides terse messages while level 9 provides verbose messages. Values greater than 9 give no additional information. The default level is 0.

- **-z**
  Returns notification of success to the user who issued the uux command.

Commands on remote sites are actually run by uuxqt in its own directory, /usr/spool/uucp/.Xqtdir.

**Special characters**

The command string passed to uux can use the shell metacharacters <, >, ;, and |. If any of these characters are not valid for the command interpreter on the destination system, the command fails.

More complex redirection, such as 2>, is not handled by uux because the 2 is interpreted as a parameter to the preceding command). Only the simple metacharacters that are listed are allowed.

To escape a file name or quoted string, use parentheses. Parentheses pass the file name to the command on the remote site without special interpretation by uux. For example, the following command will not do what you expect because "hello" is treated as a file unless enclosed in parentheses.

```
uux "Remote!echo hello >test.out"
```
The correct way to enter that command is:

```
uux "Remote!echo (hello) >test.out"
```

**Examples**

1. Suppose that a neighboring site, south, has a program called `laser` for printing and formatting documents. You have execute permission for `laser`. To print the file `inventor.y` in south’s public UUCP directory using south’s `laser` program:

```
uux south!laser ' ~/inventor.y'
```

   The tilde needs protection from shell expansion.

   To print the file `report.001` in your public UUCP directory:

```
uux south!laser ! ~/report.001
```

2. Suppose that you have execute permission for `uucp` on south. To request that south use `uucp` to copy the file `index` from its public UUCP directory to west, a neighbor of south:

```
uux south!uucp (~/index) (west! ~/)
```

   The arguments `~/index` and `west! ~/` are not interpreted by `uux` because of the parentheses. The backslashes are necessary to escape the parentheses on the z/OS shell.

**Security**

`uux` is potentially a security risk to your system. UUCP minimizes the risk by allowing you to specify the commands that can be run by each remote site. See the section on permissions file in *z/OS UNIX System Services Planning* for more information.

For electronic mail, each remote site must be able to execute a mail routing agent on your site. Further permissions can be granted at your discretion.

**Localization**

`uux` uses the following localization environment variables:

- `LANG`
- `LC_ALL`
- `LC_CTYPE`
- `LC_MESSAGES`


**Files**

`uux` uses the following files:

- `/usr/lib/uucp/config`
  UUCP configuration file
- `/usr/spool/uucp/site`
  Subdirectory containing queued job requests, work files, data files, and execution files for the UUCP host site.
- `/usr/spool/uucp/LOGFILE`
  Log file for `uux` and other UUCP utilities.
- `/usr/spool/uucp.Sequence/sitename`
  Sequence file containing the 4-digit sequence number of the last job queued. If `uux` requires a sequence number, it is based on the value in this file. If this file does not exist, `uux` creates it with the
sequence number 0000. sitename is the name of a remote site; each remote site has its own sequence number.

**Exit values**

0  
Successful completion.

1  
Failure due to any of the following reasons:
  - Argument list too long.
  - Inability to open log file.
  - Insufficient memory.

2  
Unknown command-line option.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

The -g, -p, -r, and -x options are extensions to the POSIX standard. They are retained for compatibility with other UNIX UUCP implementations.

**Related information**

uucc, uucico

---

**uuxqt daemon - Carry out command requests from remote UUCP systems**

**Format**

```bash
uuxqt [-c command_name] [-s site] [-x debug_level]
```

**Description**

The `uuxqt` daemon carries out the command requests made on other sites by mail programs, news programs, or by the `uux` command.

`uuxqt` is automatically started after `uucico` completes. Additionally, `cron` can be used to start `uuxqt` at predetermined times.

**Options**

- `-c command_name`
  - Processes only requests to run `command_name`.

- `-s site`
  - Runs only commands that are requested by `site`.

- `-x debug_level`
  - Sets the verbosity of the debugging information to `debug_level`, which is a number, 0 or greater. Level 0 provides terse messages while level 9 provides verbose messages. Values greater than 9 give no additional information. The default level is 0.

**Examples**

To run all of the commands that are requested by the remote site `north`:

```bash
uuxqt -s north
```
Usage notes

1. The `uuxqt` command is a security risk on all sites, because it allows outside access to your computer. UUCP limits the danger by setting execute permissions for every site in the configuration file.

2. `uuxqt` checks the command requests from each site against the list of allowed commands and either runs them or sends a mail message that says:

   Permission denied

Localization

`uuxqt` does not use localization environment variables.

Files

`uuxqt` uses the following files:

`/usr/lib/uucp/config`

UUCP configuration file

`/usr/spool/uucp/.Xqtdir`

This file contains permissions for UUCP sites.

`/usr/spool/uucp/.Sequence/sitename`

Sequence file containing the four-digit sequence number of the last job queued. If `uuxqt` requires a sequence number (for example, to mail a message), it is based on the value in this file. If this file does not exist, `uuxqt` creates it with the sequence number 0000. `sitename` is the name of a remote site; each remote site has its own sequence file.

`/usr/spool/uucp/site`

Subdirectory containing commands from the UUCP host site (as well as all work files and data files that are associated with `site`). The format of the execute files is described in `uucp`.

`/usr/spool/uucp/.Xqtdir`

Working directory for `uuxqt`. All required files are copied here before `uuxqt` runs a command.

Exit values

0

Successful completion.

1

Failure because of any of the following reasons:

- Argument list too long.
- Unable to open log file.
- Insufficient memory.
- Ctrl-C interrupt.

2

Unknown command-line option.

Portability

X/Open Portability Guide

Some UUCP systems produce execute files with command lines that are not supported by `uuxqt`.

Related information

`uucc, uucp, uux`
**vi - Use the display-oriented interactive text editor**

**Format**

```
vi [-BelRrsv] [+command] [-c command] [-t tag] [-w size] [-W option[option] ... [file ...]
```

Table 35 on page 807 lists the symbols that are used throughout this command description:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-L followed by a single letter</td>
<td>Control character that is transmitted by holding down the Ctrl key and the letter key at the same time.</td>
</tr>
<tr>
<td>BACKSPACE</td>
<td>The real backspace key. This key might differ from the Ctrl-H key.</td>
</tr>
<tr>
<td>ENTER</td>
<td>The ENTER key, which is labeled RETURN on some keyboards.</td>
</tr>
<tr>
<td>ESCAPE</td>
<td>The Escape key.</td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>The break key; often Ctrl-C.</td>
</tr>
<tr>
<td>→</td>
<td>The right arrow key.</td>
</tr>
<tr>
<td>←</td>
<td>The left arrow key.</td>
</tr>
<tr>
<td>↓</td>
<td>The down arrow key.</td>
</tr>
<tr>
<td>↑</td>
<td>The up arrow key.</td>
</tr>
</tbody>
</table>

**Description**

**vi** has two components: a screen editor (**vi**) and a line editor (**ex**). Each has a different set of commands. You can invoke the line editor from within the screen editor. Conversely, you can invoke the screen editor from within the line editor.

In the screen editor, you are in either **command mode** or **insert mode**. In command mode, every character you type is immediately interpreted as a command. In insert mode, every character you type is added to the text that you are editing.

There are two ways to start your session in **ex** mode:

- Invoke the command under the name **ex**.
- Invoke it under the name **vi** but specify the **-e** option.

Similarly, there are two ways to start your session in **vi** mode:

- Invoke it under the name **ex** but specify the **-v** option.
- Invoke the command under the name **vi** (without specifying **-e**).

**vi** and **ex** work on files containing text data. If a file contains the null character (value .0 or \0), it is turned into the value 0x7F. The newline character is interpreted as a line delimiter. Each line is limited to a maximum length of \{LINE_MAX\} bytes, including the newline. Any lines exceeding that length are truncated at that length. If the last line in the file does not end in a newline, a newline is added. In all those cases, **vi** marks the file as modified and displays a message.

**vi** is available if you log in to the shell with the **rlogin** command or via telnet. It is not available if you log in with the OMVS command.
The current position marker

The *current position marker* indicates a position in the text that is being edited (or was edited). In **ex** mode, the current position pointer is just the line number of the line being edited. In **vi** mode, the pointer gives this line number plus the position of the cursor within the line. The line indicated by the current position pointer is always on the screen.

**vi display conventions**

**vi** displays the input for search commands (/ and ?), **ex** commands (:), and system commands (!) on the bottom line of the screen. Error and informational messages also appear on this line. If the last line in the file is ahead of the last line of the screen, then screen lines beyond the end of the file are displayed with a single ~ character in column one. In certain infrequent circumstances (typically involving lines longer than the width of the screen), **vi** is unable to fill the display with complete lines. In this case, one or more screen lines are shown with a single @ character in column one. These lines are not part of the file content and should be ignored.

**Options**

**+command**

Begins the editing session by running the specified editor **command**. To specify multiple commands, separate them with a vertical bar (|).

**-B**

Disables the automatic conversion of tagged files. This option is ignored if the `filecodeset` or `pgmcodeset` options (-W option) are specified.

**-c command**

Runs **command** before displaying any text on the screen. **command** is any **ex** command. You can specify multiple **ex** commands by separating them with an or-bar (|) and enclosing them in quotation marks. The quotation marks ensure that the shell does not interpret the | as a pipe character. For example:

```
-c 'set all | ver'
```

**-e**

Invokes **ex**.

**-i**

Sets LISP mode. The ( and ) commands use blocks of LISP code as their context rather than sentences.

**-R**

Sets the `readonly` variable, preventing the accidental overwriting of files. Any command that writes to a file requires the ! suffix.

**-r**

Tries to recover all files that are specified on the command line after a system or editor crash. If you do not specify any files, **vi** displays a list of all recoverable files.

When you are using **vi -r** to recover a file that was being edited with automatic conversion, the file must also be recovered with automatic conversion enabled when writing the data back to the original tagged text file. Likewise, if explicit conversion was being used when editing the file (by using the `-W` `filecodeset` or `-W pgmcodeset` options), the same options must be specified when writing the recovered data back to the original file. Failure to do either of these might result in incorrectly coded character data being written to the file when you save the recovered version.

**-s**

Turns on quiet mode. The editor does not print file information messages, thus allowing **ex** to be used as a filter. Because the file is not displayed, the editor does not read the value of the TERM environment variable. This option also keeps **ex** from reading any startup files (.exrc or the file specified by EXINIT).
-t tag
   Searches for a tag in the same way that you use with the ex tag command.
   Pu-vts the editor into vi mode.

-w size
   Sets the option variable window to size. See Setting the vi options for more information.

-W option[,...]
   Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:
   filecodeset=codeset
      Performs text conversion from one code set to another when reading from the file. The coded
      character set of the file is codeset. codeset can be a code set name that is known to the system or
      a numeric coded character set identifier (CCSID). The command iconv -l lists existing CCSIDs
      along with their corresponding code set names. The filecodeset and pgmcodeset options can
      be used on files with any file tag.
      If pgmcodeset is specified but filecodeset is omitted, then the default file code set is
      ISO8859-1 even if the file is tagged with a different code set. If neither filecodeset nor
      pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or
      the _TEXT_CONV environment variable indicates text conversion. For more information about text
      conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page
      993.
      If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this
      command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System
      Services Planning for more information about automatic conversion.
      When specifying values for filecodeset, use the values that Unicode Service supports. For
      more information about supported code sets, see z/OS Unicode Services User’s Guide and
      Reference.
   pgmcodeset=codeset
      Performs text conversion from one code set to another when reading from the file. The coded
      character set of the program (command) is codeset. codeset can be a code set name known to the
      system or a numeric coded character set identifier (CCSID). The command iconv -l lists existing
      CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset
      options can be used on files with any file tag.
      If filecodeset is specified but pgmcodeset is omitted, then the default program code set is
      IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur
      unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text
      conversion. For more information about text conversion, see Appendix L, “Controlling text
      conversion for z/OS UNIX shell commands,” on page 993.
      If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this
      command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System
      Services Planning for more information about automatic conversion.
      Restriction: The only supported values for pgmcodeset are IBM-1047 and 1047.

Examples

1. To edit or browse a file that contains ASCII characters by using the vi editor, assuming that:
   • The file is untagged and you do not want to tag it or enable automatic conversion, and
   • You cannot alter the tag (for example, you are browsing an untagged public file or a read-only file)
   issue:

   vi -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 myAsciiFile
2. To edit or browse a file that contains EBCDIC characters by using the `vi` editor, assuming that automatic conversion has been enabled but the file is incorrectly tagged as UTF-8, issue:

```
vi -B myMisTaggedFile
```

**vi command summary**

The `vi` command is divided into categories: scrolling commands, movement commands, object manipulation commands, text insertion commands, and miscellaneous commands.

**Scrolling commands**

Scrolling commands adjust the position of text on the screen. The current position pointer only changes if the current line is scrolled off the screen. For example, Ctrl-E scrolls the text on the screen up one line. The cursor remains pointing to the same text that it was pointing to, unless that text is moved off the screen. See Scrolling commands.

`vi` scrolling and movement commands can be preceded by a decimal integer that serves as a `count`, as in:

```
[count] command
```

`count` means different things with different commands. If you type `count`, it is not displayed anywhere on the screen.

**Ctrl-B**

Scrolls text back by a page, (that is, a screen), less two lines. The cursor is placed on the last line of the screen. `count` specifies a number of pages to scroll. The default value for `count` is 1.

**Ctrl-D**

Scrolls text onto the bottom of the screen. The current position pointer moves forward the same amount in the text, which means that the cursor stays in the same relative position on the screen. If `count` is given, the screen scrolls forward by the given number of lines; this number is used for all future Ctrl-D and Ctrl-U commands (until a new `count` is given). The default for the scrolling amount is half the screen.

**Ctrl-E**

Scrolls a new line onto the bottom of the screen. The current position pointer is not changed unless the current line scrolls off the top of the screen; then the pointer is set to the top line. If `count` is given, the screen scrolls forward the given number of lines. The default value for `count` is 1.

**Ctrl-F**

Scrolls text forward a page (that is, a screen), less two lines. The cursor is placed on the top line of the screen. `count` specifies the number of pages to scroll. The default value for `count` is 1.

**Ctrl-U**

Scrolls text onto the top of the screen. The current position pointer moves backward the same amount in the text, which means that the cursor stays in the same relative position on the screen. `count` operates as for Ctrl-D. The default scrolling amount is half the screen.

**Ctrl-Y**

Scrolls a new line onto the top of the screen. The current position pointer is not changed unless the current line scrolls off the bottom of the screen; then the pointer is set to the bottom line. If `count` is given, the screen scrolls backward the given number of lines. The default value for `count` is 1.

```
[n] z [ni] type
```

Redraws the screen in a window of `m` lines. `type` determines the position of the current line. If `type` is the newline character, the current line is placed at the top of the window. If `type` is a period (.), the current line is placed in the middle of the window. If `type` is a minus sign (-), the current line is placed at the bottom of the window. If `n` is given, the current position pointer is first set to that absolute line number; then the screen is positioned according to `type`. If you omit `n`, it defaults to the current line. If you omit `m`, it defaults to `window`. (See Setting the `vi` options.)
Movement commands

Movement commands move the cursor in the file. For example, the character j moves the cursor down one line and the screen is scrolled only if necessary. There are two types of movement commands: absolute movement commands and context-dependent movement commands.

Absolute movement commands

Absolute movement commands move the cursor, regardless of the nature of the surrounding text. For example, j always moves the cursor down one line.

All the following movement commands except m, 0, ^, ` and ’ can be preceded by count to repeat the movement that many times.

G
Moves to the absolute line number specified as count. As a special case, if count is zero or is not specified, the cursor is moved to the last line of the file.

h
Moves the cursor one position to the left.

BACKSPACE
Moves the cursor one position to the left.

← (left arrow)
Moves the cursor one position to the left.

Ctrl-H
Moves the cursor one position to the left.

↓ (down arrow)
Moves the cursor to the next line at the same column on the screen. Scroll the screen one line if needed.

j
Moves the cursor to the next line at the same column on the screen. Scroll the screen one line if needed.

Ctrl-J
Moves the cursor to the next line at the same column on the screen. Scroll the screen one line if needed.

Ctrl-N
Moves the cursor to the next line at the same column on the screen. Scroll the screen one line if needed.

k
Moves the cursor to the previous line at the same column on the screen. Scrolls the screen up one line if needed.

↑ (up arrow)
Moves the cursor to the previous line at the same column on the screen. Scrolls the screen up one line if needed.

Ctrl-P
Moves the cursor to the previous line at the same column on the screen. Scrolls the screen up one line if needed.

l
Moves the cursor one position to the right.

→ (right arrow)
Moves the cursor one position to the right.

SPACE
Moves the cursor one position to the right.
Records the current position pointer under a mark name. A mark name is a single lowercase letter, given immediately after the m. For example, the command ma records the current location of the current position pointer under the name a.

0
(Zero) Moves the cursor to the first character of the current line.

+ Moves the cursor to the first nonblank character on the next line. Scroll the screen one line if needed.

Ctrl-M
Moves the cursor to the first nonblank character on the next line. Scroll the screen one line if needed.

- Moves the cursor to the first nonblank character on the previous line. Scrolls the screen up one line if needed.

| Moves the cursor to the column number specified as count. This is a screen column number, not a character offset. If a double-byte character occupies column positions 5 and 6, the command 6| moves the cursor to the character that includes column 6.

If count is greater than the length of the current line, vi moves the cursor to the last character on the line. If the column indicated is spanned by a tab, vi moves the cursor to the first character after the tab.

^ Moves the cursor to the first nonblank character of the current line.

$ Moves cursor forward to the end of a line. count specifies the number of lines, including the current line, to move forward.

. When followed by a mark name, moves the cursor to the position that was associated with that name. The position is set by the m command. A grave character followed by another grave character moves the cursor to the previous context. The previous context is typically the last place where you made a change. More precisely, the previous context is set whenever you move the cursor in a nonrelative manner.

Similar to the grave (´) character, except that the cursor is set to the first nonblank character on the marked line.

Context-dependent movement commands
Context-dependent movement commands move the cursor based on the nature of the text. For example, w moves the cursor to the beginning of the next word, so it must look at the text to determine where the next word begins.

vi defines a word as:

- A sequence of letters, digits, and underscores delimited at both ends by characters that are not letters, digits, or underscores; the beginning or end of a line; or the end of the editing buffer.

- A sequence of characters other than letters, digits, underscores, or white space delimited at both ends by a letter, digit, underscore, white space, the beginning or end of a line, or the end of the editing buffer.

vi defines a fullword as a sequence of nonblank characters delimited at both ends by blank characters (space, tab, newline) or by the beginning or end of a line or file.

B Moves the cursor back to the first character of the current fullword. If the cursor is already at the beginning of a fullword, vi moves it to the first character of the preceding fullword.
b
Moves the cursor back to the first character of the current word. If the cursor is already at the beginning of a word, \texttt{vi} moves it to the first character of the preceding word.

E
Moves the cursor forward to the end of a fullword. If the cursor is already at the end of a word, \texttt{vi} moves it to the last character of the next fullword.

e
Moves the cursor forward to the end of a word. If the cursor is already at the end of a word, \texttt{vi} moves it to the last character of the next word.

Fc
Searches backward in the line for the single character \texttt{c} and positions the cursor on top of it. When \texttt{count} is given, the editor searches back for the \texttt{count} the such character.

fc
Searches forward in the line for the single character \texttt{c} and positions the cursor on top of it. When \texttt{count} is given, the editor searches for the \texttt{count} the such character.

H
Places the cursor on the first nonblank character of the top line of the screen. \texttt{count} specifies the number of lines from the top of the screen.

L
Places the cursor on the first nonblank character of the bottom line of the screen. \texttt{count} specifies the number of lines from the bottom of the screen.

M
Places the cursor on the first nonblank character of the middle line of the screen.

N
Repeats previous / or ?, but in the opposite direction.

n
Repeats previous / or ?.

Tc
Searches backward in the line for the character \texttt{c} and position the cursor after the character being sought. \texttt{count} searches backward for the \texttt{count} the matching character and then positions the cursor after the character being sought.

tc
Searches forward in the line for the character \texttt{c} and position the cursor on the preceding character. \texttt{count} searches forward for the \texttt{count} the matching character and then positions the cursor on the preceding character.

W
Moves forward to the start of the next fullword.

w
Moves forward to the start of the next word.

(  
Moves back to the beginning of the previous sentence. A sentence is bounded by a period (.), exclamation mark (!), or question mark (?); followed by any number of closing double quotation marks, ("), closing single quotation marks (’), closing parentheses ()), or closing square brackets (]); followed by two spaces or the end of the line. Paragraph and section boundaries are also sentence boundaries; see \texttt{[ [} and \texttt{] ].

If you specified the \texttt{lisp} option, a \texttt{lisp} s-expression is considered a sentence for this command.

)  
Moves forward to the beginning of the next sentence. See ( for the definition of a sentence.

If you specified the \texttt{lisp} option, a \texttt{lisp} s-expression is considered a sentence for this command.
{ Moves back to the beginning of a paragraph. A paragraph begins on a blank line, a section boundary, or a text formatter macro in the paragraphs variable. }

} Moves forward to the beginning of the next paragraph. See \{ for the definition of a paragraph.

[[ Moves back to the beginning of a section. A section begins on lines starting with a form feed (Ctrl-L), starting with an open brace \{, a text formatter macro in the sections variable, or begin or end of file. If you specified the lisp option, a section boundary is also identified by a line with a leading (.

]] Moves forward to the beginning of the next section. See [[ for the definition of a section. If you specified the lisp option, a section boundary is also identified by a line with a leading (.

% Finds the balancing character to that under the cursor. The character should be one of the following characters: [[[< >]]].

; Repeats the previous F, f, T, or t command.

, Repeats the previous F, f, T, or t command in the opposite direction.

/regexp , Search forward in the file for a line that matches the regular expression regexp and position the cursor at the first character of the matching string. When used with an operator to define a text range, the range begins with the character at the current cursor position and ends with the first character of the matching string. You can specify whole lines by following regexp with /+n or /-n, where n is the offset from the matched line.

?regexp Is similar to /, but searches backwards in the file.

Ctrl-] Uses the word after the cursor as a tag. (For information about tag, see ex.)

Object manipulation commands

Object manipulation commands change the text that is already in the file. An object manipulator command works on a block of text. The command character is followed immediately by any movement command. The object that is manipulated by the object manipulator command is the text from the current position pointer to wherever the movement command would leave the cursor.

For example, in dL, d is the object manipulator command to delete an object. It is followed by the movement command L which means move to the bottom line of the screen. The object that is manipulated by the command thus extends from the current line to the bottom line on the screen; these lines are deleted.

Typically, an object extends up to, but not including, the position of the cursor after the move command. However, some movements work in a line mode. For example, L puts the cursor on the first nonblank character of the last line on the screen. If it is used in an object manipulation command, it includes the entire starting line and the entire ending line. Some other objects include the cursor position. For example, d$ deletes up to and including the last character on a line; by itself the $ would have placed the cursor on the final character. Repeating the command letter implies working on a line basis; thus 5dd deletes five lines.
Use of buffers

Objects that are deleted or otherwise manipulated have their original values that are placed in a buffer, an area of computer memory that can hold text. There are several ways this can be done:

1. You can use a named buffer. Buffers are named with single lowercase letters. To place an object in a buffer, type a double quotation mark " followed by the buffer name, followed by the object manipulator command, as in:

   "adL

   This deletes text from the current line to the bottom line on the screen and puts the deleted text in buffer a. Typically, this sort of operation overwrites the current contents of the buffer. However, if you use the same form but specify the buffer name in uppercase, the object is appended to the current contents of the buffer. For example:

   "AdL

   deletes from the current line to the bottom line on the screen, and adds the deleted text to buffer a.

2. If you are deleting material and delete at least one full line, vi uses buffers numbered 1 through 9. The first time a full line or more is deleted, the text is placed in buffer 1. The next time, the previous contents of 1 are copied to 2, and the newly deleted text is put into 1. In the same way, deleted text continues to be rippled through the nine numbered buffers. When text is rippled out of buffer 9, it is gone for good.

3. In all other cases, the object that is manipulated goes to the unnamed buffer. For example, the unnamed buffer is used if you delete less than a line of text. The unnamed buffer is like the other buffers, but does not have a name.

Examples of buffers

Following are some examples of the use of buffers:

1. To delete text from the current cursor position through to the bottom of the screen and place it into buffer 1 (this will also ripple numbered buffers), enter:

   dL

2. To delete from the current cursor position through to the next position that contains (but not including) the string fred, and place the deleted text into buffer a, enter:

   "ad/fred/+0

3. To delete the current word and place it into an unnamed buffer, enter:

   dw

List of object manipulator commands

The following section lists the object manipulator commands.

- **c**
  Deletes the object and enters insert mode for text insertion after the current cursor position. If less than one line is changed, a dollar sign ($) is placed on the final character of the object and typing goes directly over top of the current object until the dollar sign ($) is reached. Additional text is inserted, with the existing text shifting to make room for the new text.

- **d**
  Deletes the object.

- **y**
  Moves the object to the appropriate buffer; the source is not changed. This can be used to duplicate or copy objects.
<  Shifts the object left by the value of the variable shiftwidth. This operator always works on a line basis. This command replaces all leading blanks and tabs required for the new indent amount. count shifts count lines.

>  Shifts the object right by the value of the variable shiftwidth. This operator always works on a line basis. This command replaces all leading blanks and tabs that are required for the new indent amount. count shifts count lines.

!  Filters the object through an external command. After typing the object, the command line opens up for a system command which is parsed in the same manner as the ex system command (!:). This operator then invokes the given command and sends the entire object on a line basis to that command. The object is then deleted and the output from the command replaces it. For example, 1G! Gsort moves to the first line of the file; then takes all the text from the first line to the last line and runs it through the sort command. The output of sort then replaces the original text.

The following shorthand commands are equivalent to the shown object manipulations. Each command can be preceded by count or by a buffer name to save the manipulated text.

C  Changes to the end of the current line. This is equivalent to the c$ command.

D  Deletes to the end of the current line. This is equivalent to the d$ command.

s  Substitute the character. This is equivalent to the cl command.

S  Substitute the line. This is equivalent to the cc command.

x  Deletes the current character. This is equivalent to the dl command.

X  Deletes the previous character. This is equivalent to the dh command.

Y  Yanks the current line. This is equivalent to the yy command.

Text insertion commands

Text insertion commands add new text to the existing text.

A  Enters insert mode at end of line. This is equivalent to the $a command.

a  Enters insert mode after the current cursor position.

I  Enters insert mode before first nonblank character on line. This is equivalent to the ^I command.

i  Enters insert mode before the current cursor position.

O  Opens up a new line before the current line and enters insert mode on it.

o  Opens up a new line after the current line and enters insert mode on it.

R  Replaces characters on the screen with characters typed up to the next ESC. Each character typed overlays a character on the screen. The newline character is an exception; it is simply inserted and no
other character is replaced. While you are doing this, the screen may not correspond exactly to the contents of the file, because of such things as tabs. The screen is updated when you leave insert mode.

\texttt{r}

Replaces the character under the cursor with the next character typed. When \textit{count} is given, \textit{count} characters following the cursor to the new character are changed. If \textit{count} is given and the newline character is the replacement character, \textit{count} characters are deleted (as usual) and replaced with a single newline character, not \textit{count} newlines.

\textbf{Miscellaneous commands}

\texttt{J}

Joins \textit{count} lines together. If you do not specify \textit{count}, or \textit{count} is less than 2, \texttt{vi} uses a \textit{count} of 2, joining the current line and the next line. This command supplies appropriate spacing: one space between words, two spaces after a period, and no spaces at all when the first character of the line is a \texttt{)}. When a line ends with white space, \texttt{vi} retains the white space, does not add any further spaces, and then appends the next line.

\texttt{p}

Same as \texttt{p} except that text is pasted before the cursor instead of after it.

\texttt{P}

Put buffer contents before the cursor. Also called a paste operation. If preceded by quote \texttt{buffername} (for example, \texttt{“b}), the contents of that buffer are used; otherwise the contents of the unnamed buffer are used. If the buffer was created in \texttt{ex} mode, the contents of the buffer are inserted before the current line. If the buffer was created in \texttt{vi} mode, the contents are inserted before the cursor. As a special case, if a paste operation is repeated with the period (.) command and it used a numbered buffer, the number of the buffer is incremented. Thus, \texttt{“1p . . .}, pastes in the contents of buffer 1 through buffer 6; in other words the last six things that were deleted are put back.

\texttt{Q}

Switches to \texttt{ex} mode. You leave \texttt{vi} mode and the \texttt{ex} prompt is shown on the bottom line of the screen.

\texttt{U}

Undoes all changes to current line. As soon as you move off a line or invoke an \texttt{ex} command on the line, the original contents of the line are forgotten and \texttt{U} is not successful.

\texttt{u}

Undoes last change. If repeated, you undo the undo (that is, go back to what the text was before the undo). Some operations are treated as single changes; for example, everything done by a global \texttt{G} is undone with undo.

\texttt{ZZ}

 Writes the file out, if changed, and then exits.

\texttt{.}

Repeats the last command. Any command that changes the contents of the file can be repeated by this command. If you do not specify \textit{count} with the \texttt{.} command, \texttt{vi} uses the \textit{count} that was specified for the command being repeated.

\texttt{~}

Toggles the case of the character under the cursor and moves the cursor right by one. This command can be preceded by \textit{count} to change the case of \textit{count} characters.

\texttt{&}

Repeats the previous \texttt{ex} substitute command, using the current line as the target. Flags set by the previous command are ignored. Equivalent to the \texttt{ex} command \texttt{&}.

\texttt{:}

Invokes a single \texttt{ex} command. The editor places the cursor on the bottom line of the screen and displays a colon (:) to prompt for input. You can then type one or more \texttt{ex} commands; when you press ESC or a RETURN, the line you have entered is passed to \texttt{ex} and executed there.
@ Invokes a macro. When the next character is a letter from a through z, vi treats it as the name of a buffer. The contents of that buffer are treated as input typed to vi. The text of a macro may contain an @ calling another macro. A macro may call itself, provided it is invoked at the end of the macro (tail recursion). Such a macro executes forever or until an error occurs or the INTERRUPT key is pressed. A vi error terminates all currently executing macros. All changes made during a macro call are treated as a unit and can be undone with a single u command.

= Reindents the specified line as though they were set via lisp and autoindent-set, if the lisp option was specified.

Ctrl-G Displays the current path name, current line number, total number of lines in the file, and the percentage of the way through the file. This is equivalent to the ex command file.

Ctrl-L Redraws the screen assuming another process has written on it. This should never happen unless a filter ! command writes to the screen rather than the standard output.

Ctrl-R Redraws the screen, removing any deleted lines that are flagged with the @ convention.

Ctrl-Z Stops the editor and returns you to system level. You can return to the editor with the fg command; however, when you resume a vi session in this way, all of the session's buffers are empty. The jobs command lists all the stopped vi jobs. The amount of available memory limits the number of vi sessions that can be stopped at one time (see fg and jobs).

Ctrl-^ Switches to editing the alternate file (see ex for an explanation of write). If you attempt this and you have not written out the file since you made the most recent change, vi does not switch to the alternate file.

Insert mode commands

The object manipulation command c, and the text insertion commands [AaIiOoRr] put vi into INSERT mode. In this mode, most characters typed are inserted in the file. The following characters have special meaning.

Ctrl-D Decrements the autoindent for the current line by one level. This is only relevant if the variable autoindent is on.

Ctrl-H Deletes the last typed character. The character is not removed from the screen; however it is no longer in your file. When you backspace over characters, new text overwrites the old ones. You are permitted to backspace to the start of the current line regardless of where you started to insert text. (This is not true of some other versions of vi.)

BACKSPACE Deletes the last typed character. The character is not removed from the screen; however it is no longer in your file. When you backspace over characters, new text overwrites the old ones. You are permitted to backspace to the start of the current line regardless of where you started to insert text. (This is not true of some other versions of vi.)

Ctrl-J Ends the current line and starts a new one.

Ctrl-M Ends the current line and starts a new one.

RETURN Ends the current line and starts a new one.
Ctrl-Q
Inserts the following character literally, instead of using its special meaning. You could use this to escape, say, the ESC character itself. It is impossible to insert a Ctrl-J or the null character in your line.

Ctrl-V
Inserts the following character literally, instead of using its special meaning. You could use this to escape, say, the ESC character itself. It is impossible to insert a Ctrl-J or the null character in your line.

Ctrl-T
Increments the autoindent for the current line by one level. This is only relevant if the variable autoindent is on.

Ctrl-W
Deletes the word preceding the cursor and blanks. Although the characters are not removed from the screen, they are no longer in your file.

Ctrl-@
If this is the first character typed after entering insert mode, the previously typed insert mode contents are repeated. After this, you exit insert mode. Only up to 256 characters from the previous insertion are inserted.

ESC
Leaves insert mode.

INTERRUPT
Leaves insert mode.

ex command mode

_vi_ enters _ex_ command mode if the program is invoked with the -e option or if the Q command is issued from _vi_. You can issue a single _ex_ command from _vi_ using the : command.

An _ex_ command takes the general form:

```
[address-list] [[command] [!] [parameters]]
```

Each part is optional and might be invalid for some commands. You can specify multiple commands on a line by separating them with an or-bar `|`.

**address-list**
Commands can take zero, one, or two addresses. The address `%` is a short form to indicate the entire file. You can omit any or all of the addresses. In the command descriptions to follow, the addresses shown are the addresses that the commands use by default. Possible default addresses are:

- `[..]` Indicates a two-address line range defaulting to the current line.
- `[1,]$` Indicates a two-address line range defaulting to the entire file.
- `[.+1]` Indicates a single address defaulting to the next line.

**address**
An address refers to a line in the text being edited. An address can be an expression involving the following forms:

- `. ` The value of dot; that is, the current line.
- `n` A line number that indicates an absolute line in the file; the first line has absolute line number 1.
- `$` The last line in the file.
+[n]  
  n lines forward in the file. If you omit n, it defaults to 1.

- [n]  
  n lines backward in the file. If you omit n, it defaults to 1.

'x  
  The value of the mark x.

/pat/  
  Search for regular expression pat forward from the current line.

?pat?  
  Search for regular expression pat backwards from the current line.

Thus:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pattern/+3</td>
<td>are three addresses: the first searches for a pattern and then goes three lines further; the second indicates two lines after dot; and the third indicates the 100th line in the file.</td>
</tr>
</tbody>
</table>

command  
  The command is a word, which can be abbreviated. Characters that are shown in square brackets are optional. For example:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[ppend]</td>
<td>indicates that the append command can be abbreviated to a.</td>
</tr>
</tbody>
</table>

!  
  Some commands have a variant; this is typically toggled with an exclamation mark (!) immediately after the command.

parameters  
  Many ex commands use parameters to allow you to specify more information about commands. Common parameters include:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>Specifies one of the named areas for saving text.</td>
</tr>
<tr>
<td>count</td>
<td>Is a positive integer, specifying the number of lines to be affected by the command. If you do not specify count, it defaults to 1.</td>
</tr>
<tr>
<td>file</td>
<td>Is the path name for a file. If file includes the % character, vi replaces that character with the path name of the current file. If file includes the # character, vi replaces that character with the path name of the alternate file. If you do not specify a file, the default is the current file.</td>
</tr>
<tr>
<td>flags</td>
<td>Indicate actions to be taken after the command is run. It can consist of leading plus (+) and minus (-) signs to adjust the value of the current line indicator, followed by p, l, or # to print, list, or number a line. Thus:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.+5 delete 6 ++#</td>
<td>deletes starting five lines down from dot; six lines are deleted; the current line indicator is set to the following line, then incremented by two; and that line is printed with its line number.</td>
</tr>
</tbody>
</table>

ex commands  
You can enter these commands as shown in ex mode. In vi mode, they must be preceded by the colon (:) character.
**ab{breviate} lhs rhs**
Indicates that the word lhs should be interpreted as abbreviation for rhs. (See “Context-dependent movement commands” on page 812 for the definition of word.) If you enter lhs surrounded by white space in vi INSERT mode, it is automatically changed into rhs. If you do not specify any arguments for the ab command, it displays the abbreviations that are already defined. Abbreviated names cannot contain # or any other form of punctuation.

**[..] a{ppend}[]!**
Enter ex INSERT mode. Text is read and placed after the specified line. An input line consisting of one period (.) leaves INSERT mode. If you specify an address of zero, text is inserted before the first line of the file. The current line indicator points to the last line typed.

If an exclamation mark (!) is specified, the autoindent option is toggled during input. This command cannot be invoked from vi mode.

**args**
Displays the current list of files being edited. The current file is shown enclosed by square brackets.

**cd[] path**
Changes the current directory to path. If you omit path, cd sets the current working directory to the directory identified by the HOME environment variable. If path is a relative path name, cd searches for it using the directories that are specified in the CDPATH environment variable. If path is -, then cd changes to the previous working directory. If you modified the buffer since the last write, vi displays a warning message. You can override this behavior by including the exclamation mark (!).

**[..] c{hange}[]! [count]**
Deletes the line range that is given and then enters INSERT mode. If an exclamation mark (!) is specified, autoindent is toggled during input. You cannot invoke this command from vi mode.

**chd[ir][]! [path]**
Same as cd.

**[..] c{o}py [addr] [flags]**
Copies the line range given after addr. If addr is zero, the lines are inserted before the first line of the file. The current line indicator points to the last line of the inserted copied text.

**[..] d{elete} [buffer] [count] [flags]**
Deletes the specified line range. After the line range is deleted, the current line indicator points to the line after the deleted range. A buffer can be specified as a letter a-z. If so, deleted lines are saved in the buffer with that name. If an uppercase letter is specified for buffer, the lines are appended to the buffer of the corresponding lowercase name. If no buffer name is given, deleted lines go to the unnamed buffer.

**e{dit}[]! [+line] [file]**
Beginning a new editing session on a new file; the new file replaces the old file on the screen. The text conversion that is specified on the vi or ex command (for example, the -B or -W option) is used. This command is usually invalid if you have modified the contents of the current file without writing it back to the file. Specifying an exclamation mark (!) goes on to start a new session even you have not saved the changes of the current session.

You can specify line as either a line number or as a string of the form /regexp or ?regexp where regexp is a regular expression. When line is a line number, the current line indicator is set to the specified position. When it has the form /regexp, vi searches forward through the file for the first occurrence of regexp and sets the current line indicator to that line. ?regexp is similar to /regexp except that vi searches through the file backwards. If you omit line and do not specify a file, the value of the current line indicator does not change. Otherwise, if a file is specified, the current line indicator is set to either the first or last line of the buffer, depending on whether the command was issued in vi or ex mode.

**ex[]! [+line] [file]**
Beginning a new editing session on a new file; the new file replaces the old file on the screen. The text conversion that is specified on the vi or ex command (for example, the -B or -W option) is used. This command is typically invalid if you have modified the contents of the current file without writing it back to the file. Specifying an exclamation mark (!) goes on to start a new session even you have not saved the changes of the current session.
You can specify line as either a line number or as a string of the form /regexp or ?regexp where regexp is a regular expression. When line is a line number, the current line indicator is set to the specified position. When it has the form /regexp, vi searches forward through the file for the first occurrence of regexp and sets the current line indicator to that line. ?regexp is similar to /regexp except that vi searches through the file backwards. If you omit line and do not specify a file, the value of the current line indicator does not change. Otherwise, if a file is specified, the current line indicator is set to either the first or last line of the buffer, depending on whether the command was issued in vi or ex mode.

f[file] [file]
Changes the current file name to file and marks it [Not edited]. If this file exists, it cannot be overwritten without using the exclamation mark (!) variant of the write command.

[1,$] g[lobal](!) /pat/ [commands]
Matches pat against every line in the given range. On lines that match, the commands are run. If the exclamation mark (!) variant is set, the commands are run on lines that do not match. This is the same as using the vi command.

The gglobal command and the undo command cannot occur in the list of commands. A subsequent undo command undoes the effect of the global command. In ex mode, multiple command lines can be entered by ending all but the last with a backslash (\). Commands that will take input are permitted; the input is included in the command list, and the trailing period (.) can be omitted at the end of the list. For example, g/rhino/a hippo appends the single line hippo to each line containing rhino. delim is an arbitrary, nonalphabetic character. The total length of a global command list is limited (see “Limits” on page 836).

[] [insert](!)
Enter ex INSERT mode, reads text and places it before the specified line. Otherwise, this is identical to the append command. This command cannot be entered from vi mode.

[,+1] [join](!) [count] [flags]
Joins together the lines of text within the range. Unless an exclamation mark (!) is specified, all white space between adjacent joined lines is deleted. Two spaces are provided if the previous line ended in a period, no spaces if the joined line begins with a opening parenthesis, and one space otherwise.

[.] k x
Synonymous with the mark command.

[.,] l[ist] [count] [flags]
Displays the line range in a visually unambiguous manner. This command displays tabs as ^I, and the end of lines as $. The only useful flag is #, for line numbering. The current line indicator points to the last line displayed.

map[l] lhs rhs
This defines macros for use in vi. The lhs is a string of characters; whenever that string is typed exactly, vi behaves as if the string rhs had been typed. If lhs is more than one character long, none of the characters are echoed or acted on until either a character is typed that isn’t in the lhs (in which case all the characters up to that point in the lhs are run) or the last character of lhs is typed. If the variable remap is set, rhs itself can contain macros. If the flag ! is specified, the map applies within vi INSERT mode; otherwise it applies to command mode. A map command with no arguments lists all macros currently defined.

[.] ma[ark] x
Records the specified line as being marked with the single lowercase letter x. The line can then be addressed at any point as ^x.

[.,] m[ove] [addr] [flags]
Moves the specified line range after the addr given. If addr is zero, the text is moved to the start of the file. The current line indicator is set to the last line moved.

n[ext]([+command] [file ...]
Beginning editing the next file in the file list (where the file list was either specified on the command line or in a previous next command). The text conversion that is specified on the vi or ex command (for example, the -B or -W option) is used. If the current file has been modified since the last write, ex typically prevents you from leaving the current file. You can get around this by specifying an
exclamation mark (!). If the autowrite is set, the current file is written automatically and you go to
the next file. If a list of files is specified, they become the new file list. If necessary, expressions in this
list are expanded. Thus:

```
next *.c
```

sets the file list to all the files in the current directory with names ending in .c (typically C source
files).

```
[.,] num[ber] [count] [flags] [.,] # [count] [flags]
```

Displays the specified line range with leading line numbers. The current line indicator points to the
last line displayed.

```
[.] o[pen] [pat] [flags]
```

Enters open mode, which is simply vi mode with a oneline window. If a match is found for the regular
expression pat in the specified line, then the cursor is placed at the start of the matching pattern.

```
p[I][set] [parameter-list]
```

Assigns or displays the values of option variables, If you do not specify a parameter list, set displays
all the variables with values that have changed since the editing session started. If the parameter all
is specified, ex displays all variables and their values. You can use the parameter list to set or display
each of many variable values. Each argument in the list is a variable name; if it is a Boolean variable,
the value is set on or off depending on whether the name is prefixed by no. Non-Boolean variables
alone in an argument are a request to display their values. A Boolean variable’s value can be displayed
by following the name by a question mark (?). You can set numeric or string variables with:
name=value In a string variable, spaces must be preceded by a backslash. For example:

```
set readonly? noautowrite shell=/bin/sh
```

shows the value of the readonly flag, sets noautowrite, and sets the shell to /bin/sh.

```
set report report=5
```

shows the value of the report variable, and then set the value to 5. For more details. See Setting the vi options for more details.

sh[ell]
Invokes a child shell. The environment variable SHELL is used to find the name of the shell to run.

so[urce] file
Runs editor commands from file. A file being executed with source can contain source commands of its own.

st[op]
Suspends the editor session and returns to system level. For more information, see the description of the vi command Ctrl-Z.

[..] s[ubstitute] [(pat/repl/) [options] [count] [flags]]
Searches each line in the line range for the regular expression pat and replaces matching strings with repl.

Normally, ex only replaces the first matching string in each line. If options contains g [global], all matching strings are changed.

If options contains c [confirm], ex first prints the line with caret (^) characters marking the pat matching location; you can then type y if you want ex to go ahead with the substitution. pat cannot match over a line boundary; however in ex mode, repl can contain a newline, escaped by a preceding backslash (\). See Appendix C, “Regular expressions (regexp),” on page 945 for full information about both pat and repl. If there is no pat or repl, ex uses the most recently specified regular expression or replacement string. You can use any nonalphabetic character in place of the slash (/) to delimit pat and repl.

su[spend]
This is synonymous with the stop command.

[..] t addr [flags]
This is synonymous with the copy command.

ta[g][!] tagname
Looks up tagname in the files listed in the variable tags. If the tag name is found in a tags file, that file also contains the name of the file that contains the tag and a regular expression required within that file to locate that tag. If the given file is different from the one you are currently editing, ex normally begins editing the new file. The text conversion that is specified on the vi or ex command (for example, the -B or -W option) is used. However, if you have modified the current file since the last time it was written out, ex does not start editing a new file unless the tag command contains an exclamation mark (!). If autowrite is on, the current file is automatically written out and the new file read in. When the new file is read in, the regular expression from the tags file is invoked with the magic variable off.

Tag names are typically used to locate C function definitions in C source files. The first step is to create a tags file using the ctags command. After you do this, you can use the ex tag command to look up a particular function definition and go directly to that definition in the file that contains it.

All characters in tag names are significant unless the variable taglength is nonzero; in this case, only the given number of characters are used in the comparison.

una[b abbreviate] lhs
The abbreviation lhs previously created by abbreviate is deleted.
undone
Undoes the last change or set of changes that modified the buffer. Globals and vi macros are both considered as single changes that can be undone. A second undo undoes the undo restoring the previous state. The edit command cannot be undone, because it cleans up the temporary file which is used to maintain undo information. You cannot undo operating system commands and commands that write output to the file system.

unmap lhs
Deletes the lhs map. If the flag ! is used, this applies to the insert mode maps; otherwise it applies to the command mode maps.

[1,$] v /pat/ commands
This is a synonym for the global command with the ! flag; that is, a global for all nonmatching lines. You can use any nonalphabetic character to delimit pat instead of the slash (/).

version
Displays the current version information for vi or ex.

. vi [type] [count] [flags]
Enters vi mode. If no type is specified, the current line is at the top of the screen. If type is caret (^), the bottom line of the screen is one window before the current line. If type is a minus sign, (-), the current line is at the top of the screen. If type is a period (.), the current line is in the middle of the screen.

You can use the undo command to undo all the changes that occurred during the vi command.

[1,$] w[rite][!] [>>] [file]
Writes the given range of lines to file. The text conversion that is specified on the vi or ex command (for example, the -B or -w option) is used. If two right angle brackets (>>) are included, the lines are appended to the current contents of the file. If the current file name is not set, a file must be given. This becomes the current file name. Otherwise, file becomes the alternate file name if it is specified. If the file begins with an exclamation mark (!), then it is taken as a system command. vi writes the given range to the command through a pipe.

If a file is given, it must not already exist. The variable readonly must not be set. If a file is not given, the file must be edited; that is, it must be the same file as that read in. All these conditions can be overridden by using the flag !.

[1,$] wn[!] [>>] [file]
Similar to write, except that it begins editing the next file in the file list immediately afterward (if the write is successful).

[1,$] wq[!] [>>] [file]
Similar to write, except that it exits the editor immediately afterward (if the write is successful).

x[it]
If you have modified the current file since the last write, performs a write command using the specified range and file name and then terminates.

[] y[ank] [buffer] [count]
Copies the given line range to the specified buffer (a letter from a through z). If a buffer is not specified, the unnamed buffer is used. Buffers are not destroyed by an edit command, so yank and put can be used to move text between files.

Because the edit command does not destroy buffers, you can use that command in conjunction with put and yank to move text between files.

[.+1]z [type] [count] [flags]
Displays count lines. If no count is specified, ex uses the current value of the scroll variable. The lines are displayed with the given line located according to the type. If type is a plus sign (+), the editor displays the given line and a screen after that. If type is a period (.), the editor displays a screen with the given line in the middle. If type is a minus sign (-), the editor displays a screen with the given line at the end. If type is a caret (^), the editor displays the screen before that. If type is an equal sign (=), the current line is centered on the screen with a line of hyphens printed immediately before and after it. The current line indicator points to the last line displayed.
[<...>] <[<...] [count] [flags]
Shifts the line range by the value of the shiftwidth variable. If there are multiple left angle brackets (<), each one causes another shift. The current line indicator points to the last line displayed. If a count is specified, that many lines are shifted.

[<...>] >[<...] [count] [flags]
Shifts the line range right by the value of the shiftwidth variable. If there are multiple right angle brackets (>), each one causes another shift. The current line indicator points to the last line displayed. If a count is specified, that many lines are shifted.

[range] ! command
Submits command to be run by the command interpreter that is named by the SHELL environment variable. If range is given, the command is invoked with the contents of that line range as input. The output from the command then replaces that line range. Thus: 1,$!sort sorts the entire contents of the file.

Substitutions are made in command before it is run. Any occurrences of an exclamation mark (!) are replaced by the previous command line, while occurrences of percentage (%) and hash mark (#) characters are replaced with the path names of the current and alternate files, respectively. If any such substitutions actually take place, the new command line is displayed before it is executed. (See the read and write sections in “ex command mode” on page 819 for more information about the current and alternative files.)

If the file has been modified and the variable autowrite is on, the file is written before calling the command. If autowrite is off, a warning message is given.

[$] =
Displays the given line number. The default line number is the last line of the file. The current line indicator is not changed.

“ a line of text
This is a comment.

[... & [options] [count] [flags]
Repeats the last substitute command. If any options, count, or flags are specified, they replace the corresponding items in the previous substitute command.

[...] ~ [options] [count] [flags]
Repeats the last substitute command. However, the regular expression that is used is the last regular expression; that is, if there has been a search, the search’s regular expression is used. The simple substitute with no arguments, or the & command, uses the regular expression from the previous substitute. substitute with an empty regular expression uses the last regular expression, like ~. If any options, count, or flags are specified, they replace the corresponding items in the previous substitute command.

@ buffer
Executes each line in buffer as an ex command. If you do not specify buffer or if you specify a buffer named @, the last buffer executed is used.

Ctrl-D
Displays the number of lines of text given by the scroll variable. The current line indicator points to the last line displayed.

Special characters in ex commands
When an ex command contains the percentage character (%), the character is replaced by the name of the current file. For example, if you are about to try out a macro and you are worried that the macro might damage the file, you could issue:

```
!cp % /tmp
```
to copy the current file to a safe holding place. As another example, a macro could use the percentage character (%) to refer to the current file.
When an `ex` command contains the hash mark (#), the character is replaced by the name of the alternate file. The name of the alternate file can be set with the `read` command as described previously. Thus a command like:

```
e #
```

tells `ex` to edit the alternate file. Using an alternate file can be convenient when you have two files that you want to edit simultaneously. The command just given lets you flip back and forth between the two files.

**Setting the `vi` options**

Options are set with the `set` command. For example:

```
set autowrite
```

sets the autowrite option. For options that are flags (that is, are not numeric), the variables can be turned off by putting `no` in front of the name in the `set` command, as in:

```
set noautowrite
```

In the following list, variables that are off by default are preceded by `no`. The minimal abbreviation of each option is shown after the comma. Default values are shown after the equal sign (=).

**autoflush, af**

When this option is set, it holds the maximum number of seconds of data a user would lose if a system crash occurs. `vi` flushes memory out to its temporary files approximately this many seconds, unless no changes were made to the current edit buffer, or the user is sitting idle. It allows you to eventually recover a more current representation of your edit buffer (after the `exrecover` daemon and `vi -r` is run) because it intermittently updates `vi`'s temporary files that are used by the `exrecover` daemon.

Note the following comments:

- The default is set to 120 seconds (2 minutes).
- To turn off this option, set `autoflush` to 0.
- This option does not affect on read-only files.
- This option is different from the previous `preserve` option because it works with `vi`'s temporary files (whose location is specified by the environment variables: TMP_VI, TMPDIR, or TMP) as opposed to recovered files found in `/etc/recover/$LOGNAME`.

**autoindent, ai**

When `autoindent` is on and you are entering text, the indentation of the current line is used for the new line. In `vi` mode, you can change this default indentation by using the control keys Ctrl-D (to shift left) or Ctrl-T to shift right. In `ex` mode, a tab or spaces can be typed at the start of a line to increase the indent, or Ctrl-D can be typed at the start of the line to remove a level. `^`Ctrl-D temporarily removes the indentation for the current line. `^`Ctrl-D places the current line at a zero indent level, and the next line has this indent level as well.

The size of indent levels is defined by the variable `shiftwidth`. Based on this value and the value of `tabstop`, the editor generates the number of tabs and spaces needed to produce the required indent level.

The default is `noautoindent`.

**autoprint, ap**

When this option is set in `ex` mode, the current line is printed after the following commands: `copy`, `delete`, `join`, `move`, `substitute`, undo, &, ~, <, and >. Automatic displaying of lines does not take place inside global commands.

The default is `autoprint`. 

Shell command descriptions 827
**autowrite, aw**
When this option is on, the current file is automatically written out if it has been changed since it was last written and you have run any of the following commands: `next`, `rewind`, `tag`, Ctrl-^ (vi), and Ctrl-] (vi). Using an exclamation mark (!) with any of these commands stops the automatic write.

The default is noautowrite.

**beautify, bf**
When this option is on, the editor discards all nonprinting characters from text that is read in from files.

The default is nobeautify.

**cdpath**
Used by cd to find relative path names when changing the directory. You must delimit entries with a colon (:). If the current directory is to be included in the search, it must be indicated by a dot (.).

cdpath defaults to the contents of the CDPATH environment variable if it exists, or to dot (.) if it does not.

**directory, dir**
The editor uses temporary files with unique names under the given directory. Any error on the temporary files is fatal.

The default is directory=tmp.

**edcompatible**
When this option is on, the editor attempts to make substitution commands behave in a way that is compatible with the ed editor. The `g` and `c` options on the substitute commands are remembered and toggled by their occurrence. The `r` option uses the last regular expression rather than the last substitute regular expression. Percentage mark (%) as the entire pattern is equivalent to the previous pattern.

The default is nocompatible.

**errorbells, eb**
When this option is on, vi precedes error messages with the alert character. When it is off, the editor warns you of an error by displaying a message using a standout mode of your terminal (such as reverse video).

The default is noerrorbells.

**exrc**
When this option is on, ex and vi access any .exrc files in the current directory during initialization. If it is off, ex and vi ignore such files unless the current directory is the HOME directory.

**home**
Used as the destination directory by cd. If no path is specified, home defaults to the contents of the HOME environment variable if it exists, or to the vi startup directory if it does not.

**ignorecase, ic**
When this option is on, the case of letters is ignored when matching strings and regular expressions.

The default is noignorecase.

**linedelete**
vi sets the line delete character automatically to the current terminal line delete character, as specified by the user. Within vi, you can set the line delete character with the linedelete variable. The value that you specify is the numeric value of the line delete character. The default is 0x15, the ASCII value for Ctrl-U. Another value is 0x18 for Ctrl-X.

**list**
When this option is on, tabs are displayed as a caret mark (^) rather than expanded with blanks, and the ends of lines are indicated with a dollar sign ($).

The default is nolist.
magic
When this option is off (nomagic), regular expression characters ^ \ and $ become the only ones with special meanings. All other regular expression metacharacters must be preceded by a backslash (\) to have their special meaning.

The default is magic.

maxbuffers
The number of K units (1024 bytes) of memory to be used for the editor buffers. These are allocated in units of 16 K.

The default is maxbuffers=512, but if that is not available upon entry, this is set to the number actually obtained. At least 32 K is needed. This is in addition to the code and data space that is required by vi; this may be as much as 128 K. Changing maxbuffers has no effect.

mesg
When this option is on, ex allows others to use the write or talk commands to write to your terminal while you are in visual mode. The command

\bmesg \b n\b

overrides this variable (see mesg). This option does not affect systems that do not support mesg.

number, nu
When this option is on, line numbers are displayed to the left of the text that is being edited.

The default is nonumber.

paragraphs
This list of character pairs controls the movement between paragraphs in vi mode. Lines beginning with a period (.) followed by any pair of characters in the list are paragraph boundaries (for example, .IP). Such lines are typically commands to text formatters like nroff or troff.

The default is paragraphs="IPLPPPQPP LIpplpipbp"

prompt
When this option is on, ex command mode prompts with a colon (:). No prompts are given if input is not being read from a terminal.

The default is prompt.

pwd
This is a read-only variable. The value always refers to the current working directory, and can only be changed by the cd command.

quiet
When this option is on, vi does not display file information messages.

The default is set by the -s option.

readonly
When this option is on, vi does not let you write to the current file.

The default is based on the permissions of the current file. If you do not have write permission on this file, the default is readonly. Otherwise, the default is set by the -R option.

remap
If this option is on and a map macro is expanded, the expansion is reexamined to see if it also contains map macros.

The default is remap.

report
The editor displays a message whenever you issue a command that affects more than this number of lines.

The default is report=5.
restrict
All file names are restricted to the current directory. Subcommands cannot be called. This variable is automatically set if you invoke the editor with a command that starts with the letter r, as in rvi. When the option is turned on, it cannot be turned off.

The default is norestrict.

scroll
This sets the number of lines to scroll for the \texttt{z ex} and \texttt{Ctrl-D (ex)} commands.

The default is the value of the variable window, divided by two.

sections
This list of character pairs controls the movement between sections in \texttt{vi} mode. Lines beginning with a period (\texttt{.}) followed by any pair of characters in the list are section boundaries (for example, \texttt{. SH}). Such lines are typically commands to text formatters like nroff or troff.

The default is \texttt{sections=\"SHNH \ HU\"}

shell, sh
This is the name of the command interpreter to be used for \texttt{!} commands and the shell command. The default value is taken from the SHELL environment variable.

shiftwidth, sw
This sets the width of indent that is used by shift commands and autoindent.

The default is \texttt{shiftwidth=8}.

showmatch, sm
If this option is on and you type a closing parenthesis or closing brace in input mode, the cursor moves to the matching open parenthesis or brace. It stays there for about one second and then moves back to where you were. This lets you note the relationship between opening and closing parentheses/braces.

The default is no\texttt{showmatch}.

showmode
When this option is on, \texttt{vi} displays an indicator in the lower right corner of the screen if you are in Insert/Open/Change/Replace mode. If no indicator is displayed, you are in Command mode.

The default is no\texttt{showmode}.

tabstop
Tab stops for screen display in \texttt{vi} mode are set to multiples of this number.

The default is \texttt{tabstop=8}.

taglength, tl
If this variable is nonzero, tags are only compared for this number of characters.

The default is \texttt{taglength=0}.

tags
The value of this variable should be a list of file names that are separated by a backslash (\texttt{\}) followed by a space. If there is no backslash before the space, \texttt{vi} treats the second and subsequent tags as part of an \texttt{option=value} combination. For example:

\begin{verbatim}
  set tags=file1\ file2\ file3\ 
\end{verbatim}

These are used by the tag \texttt{ex} command and the \texttt{Ctrl-]} \texttt{vi} command. The files are typically created with the \texttt{ctags} program.

The default is \texttt{tags=tags}.

term
The value of this variable is the terminal type. The TERM environment variable specifies this variable's default value.
**terse**

If this option is on, messages are displayed in an abbreviated form. The default is *not* *terse*.

**warn**

When this option is on, *commands* with an exclamation mark (!) print a warning message if the current file has been modified. No message is printed if this option is off.

The default is *warn*.

**window**

This variable gives the number of text lines available in **vi** mode or the default number of lines to display for the command.

The default is given by the `-w` option. If it is not specified with the `-w` option, its value defaults to the environment variable LINES or the value found in the terminfo database for TERM.

**wrapmargin wm**

If this variable is nonzero in **vi** insert mode, when a line reaches this number of characters from the right of the screen, the current word moves down to the next line automatically; you do not have to press ENTER.

The default is *wrapmargin*=0.

**wrapscan, ws**

If this option is off, forward searches stop at the end of the file and backward searches stop at the beginning.

The default is *wrapscan*

**writeany, wa**

If this option is off, the editor does not let a file marked [Not edited] overwrite an existing file.

The default is *nowriteany*.

**Regular expressions**

Many **ex** commands use regular expressions when searching and replacing text. A *regular expression* (indicated by *pat* in the command descriptions) is used to match a set of characters.

A regular expression consists of a string of normal characters that exactly match characters in a line. These can be intermixed with special characters (known as *metacharacters*), which allow matching in some special manner. Metacharacters can themselves be matched directly by preceding them with the backslash (\) character. If the variable magic is turned off, all but two of the metacharacters are disabled; in this case, the backslash character must precede them to allow their use as metacharacters. See Appendix C, "Regular expressions (regexp)," on page 945 for examples.

\^  Matches the start of a line. This is only a metacharacter if it is the first character in the expression.

\$  Matches the end of a line. This is only a metacharacter if it is the last character in the expression.

.  Matches any single character.

\*  Matches zero or more occurrences of the previous expression.

\<  Matches the empty string preceding the start of a word. A word is a series of alphanumeric or underscore characters that are preceded by and followed by characters that are not alphanumeric or underscore.
\>  Matches the empty string following the end of a word. A word is a series of alphanumerical or underscore characters that are preceded by and followed by characters that are not alphanumeric or underscore.

[string]  Matches any of the characters in the class that is defined by string. For example, [aeiouy] matches any of the vowels. You can put a range of characters in a class by specifying the first and last characters of the range, with a hyphen (-) between them. For example, in ASCII [A-Za-z] matches any upper or lowercase letter. If the first character of a class is the caret (^), the class matches any character that is not specified inside the square brackets. Thus, in ASCII [a-z_][^0-9] matches a single alphabetic character or the underscore, followed by any nonnumeric character.

\(...\)  A set of characters in the pattern can be surrounded by escaped parentheses. See the description of \n in Replacement patterns. This is not affected by the setting of magic.

~  Matches the replacement part of the last substitute command.

Replacement patterns
A replacement pattern (indicated by repl in the command descriptions) describes what to put back in a line for the set of characters matched by the regular expression.

&  Is replaced by the entire string of matched characters.

~  Is replaced by the entire replacement pattern from the last substitute.

\n  Is replaced by the string that matched the nth occurrence of a \( \ldots \) in the regular expression. For example, consider:

\[
\text{s/\([a-zA-Z]*\)our/\1or/}
\]

The \1 represents the string that matched the regular expression \([a-zA-Z]*\). Thus, the previous command might change the word colour to color.

\u  Changes the next character in the replacement to uppercase.

\l  Changes the next character in the replacement to lowercase.

\U  Changes the following characters in the replacement to uppercase.

\L  Changes the following characters in the replacement to lowercase.

\E, \e  Turns off the effects of \U or \L.

Initializing the editor
Initialization code consists of one or more ex commands that run when the editor starts up. Initialization code can be obtained in several ways:

1. If there is an environment variable named EXINIT with a nonnull value, it is assumed to hold initialization code. vi executes this code using an ex source command.
2. If EXINIT does not exist or has a null value, the editor tries to find a file named .exrc. If you have an environment variable named HOME, the value of this variable is assumed to be the name of your home directory. vi runs the .exrc file using an ex source command.

3. If TEXINIT or the $HOME/.exrc file sets the option variable exrc and if there is a file named .exrc under the current directory, it is assumed to hold initialization code. vi runs this code using an ex source command.

All .exrc files must be owned by the same user ID that invoked the vi command, and must only be writable by that user ID. Typical permissions for a .exrc file would be 744.

The .exrc file is read as if it were a sequence of keystrokes typed at the beginning of an ex session. As a result, the contents of .exrc must be the same as the characters you would type if you were in vi or ex. In particular, if the input contains an unusual character (for example, a carriage return) that you would typically precede with Ctrl-V, there must be a Ctrl-V in the .exrc file. If you are creating an .exrc file with vi, you must type Ctrl-V Ctrl-V to put a Ctrl-V character into your initialization file, then Ctrl-V followed by the special character to put the special character into your initialization file. The .exrc file must show both Ctrl-V and the special character. A command specified in the .exrc file can be ignored (treated as a comment) by beginning that line with a double quotation mark (").

Files

vi uses the following files:

/tmp
  Directory used for temporary files if TMP_VI, TMPDIR, and TMP are not defined.

/tmp/VIInnnnn.mmm
  Temporary files.

.exrc
  Startup file.

Localization

vi uses the following localization environment variables:

• LANG
• LC_COLLATE
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH


Environment variables

vi uses the following environment variables:

_TEXT_CONV
  Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

CDPATH
  Contains a list of directories to be searched.

COLUMNS
  Contains the number of columns between the left and right margins (see option variable wrapmargin). This is also used as the horizontal screen size.
ENV
Contains the path name of a file containing KornShell commands. When you invoke sh, it executes this file before doing anything else.

EXINIT
Contains a list of vi commands to be run when the editor is started up.

HOME
Contains the directory to be searched for the editor startup file.

LINES
Contains the number of lines in a screen (see option variable windows). This is also used as the vertical screen size.

PATH
Contains a list of directories to be searched for the shell command specified in the ex commands read, write, and shell.

SHELL
Contains the name of the command interpreter for use in !, shell, read, write, and other ex commands with an operand of the form !string. The default is the sh utility.

TERM
Contains the name of the terminal type.

TERMINFO
Contains the path name of the terminfo database.

TMPDIR
Contains the path name that the shell uses as the directory for temporary files.

TMP_VI
Contains a directory path name that can be specified by an administrator as a location for vi's temporary files. This is useful if the current default directory for these files (typically /tmp) is implemented as a TFS. In this case, all vi's temporary files that the exrecover daemon uses for recovery would be gone after a system crash.

This environment variable should be set by a system administrator as opposed to a user setting it for their environment. If the latter occurs, and the user sets the TMP_VI directory to something different from what exrecover recognizes as TMP_VI, the user will need to run the exrecover daemon manually to allow the temporary files to be converted to the recoverable files used by vi.

(located in /etc/recover/$LOGNAME).

When you set TMP_VI, follow these restrictions:

1. Do not set TMP_VI to /etc/recover/$LOGNAME
2. Do not set TMP_VI to any directory where a path name component is an environment variable with a user's value different from the initialization process's value (for example $HOME). The temporary files that are connected with vi are converted into a form recoverable by vi when exrecover is run during IPL. Because exrecover is issued during IPL, it is owned by the initialization process and will contain different values for certain environment variables, if those environment variables are set. Throughout the file system, there might exist some temporary files that can only be converted by exrecover. This conversion can be done manually by a system administrator (to recover files that are owned by all users) or by a single user (to recover only their own files).

Exit values

0
Successful completion.

1
Failure due to any of the following reasons:

• The code set is not valid.
• Could not turn off automatic conversion.
• Could not perform requested text conversion.
• Unknown option.
• No such command from open/visual.
• Missing \textit{lhs}.
• Missing file name.
• System does not support job control.
• Write forms are \texttt{w} and \texttt{w>>}.
• Internal error: bad seek pointer.
• Internal error: Line out of range.
• Internal error: line too long.
• Nonzero address required on this command.
• No lines in the buffer.
• Nothing to undo.
• Cannot escape a newline in global from visual.
• Global command too long.
• Argument list too long.
• File is read-only.
• No previous command to substitute for !
• Command too long.
• No previous regular expression.
• Buffers are 1-9, a-z.
• Line too long.
• System does not support job control.
• Digits required after =.
• Nothing in buffer.
• Missing \textit{rhs}.
• Too many macros.
• Recursive map expansion.
• Nothing to repeat.
• Last repeatable command overflowed the repeat buffer.
• Bad tag.
• No tags file.
• No such tag in tags file.
• Negative address. The first buffer line is 1.
• Not an editor command.
• Unimplemented \texttt{ex} command.
• Wrong number of addresses for command.
• Mark requires following letter.
• Undefined mark referenced.
• Global within global not allowed.
• First address exceeds second.
• Cannot use open/visual unless open option is set.
• Regular expression \texttt{\textbackslash} must be followed by / or \texttt{?}. 
• No address allowed on this command.
• No more files to edit.
• No current file name.
• Extra characters at end of command.
• Not that many lines in buffer.
• Insufficient memory.
• Restricted environment.
• Command too long.
• Trailing address required.
• Destination cannot straddle source in m and t.
• No file name to substitute for %.
• No alternate file name to substitute for #.
• File name too long.
• Too many file names.
• Argument buffer overflow.
• Incomplete shell escape command.
• Regular expressions cannot be delimited by letters or digits.
• No previous scanning regular expression.
• No previous substitute to repeat.
• Cannot escape newlines into regular expressions.
• Missing [.
• Badly constructed regular expression.
• No remembered regular expression.
• Line overflow in substitute.
• Replacement pattern contains \d; cannot use in regular expression.
• Replacement pattern too long.
• Regular expression too complicated.
• Cannot escape newline in visual.
• No such set option.
• String too long in option assignment.

2
• Unknown command-line option.
• Missing or incorrect num in an -n option.

Limits
• Maximum number of lines: 65 279 (64 K - 256 - 1).
• Length of longest line: \{LINE_MAX\} bytes, including the newline.
• Longest command line: 256 bytes.
• Length of filenames: 128 bytes.
• Length of string options: 64 bytes.
• Length of remembered regular expressions: 256 bytes.
• Number of map, map!, and abbreviate entries: 64 each.
• Number of saved keystrokes for \ in \textit{vi}: 128.
• Length of the \texttt{lhs} of \texttt{map}, \texttt{map!}, or \texttt{abbreviate}: 10 bytes.
• Maximum number of characters in a tag name: 30.
• Number of characters in a \texttt{:} escape from \textit{vi}: 128.
• Requires 128K of memory plus the set option \texttt{maxbuffers} \texttt{K} of auxiliary memory. During startup, \texttt{maxbuffers} is changed to reflect available memory; at least 32K is required.
• Number of nested source files is 3.

\textbf{Portability}

POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.
The \texttt{-B}, \texttt{-e}, \texttt{-s}, \texttt{-v}, and \texttt{-W} options are extensions of the POSIX standard.

\textbf{Related information}

\texttt{ed}, \texttt{ex}, \texttt{fg}, \texttt{jobs}, \texttt{msg}, \texttt{sed}, \texttt{talk}, \texttt{tee}, \texttt{write}

See \texttt{Appendix C, "Regular expressions (regexp),"} on page 945 for more information about \texttt{regexp}.

\section*{wait - Wait for a child process to end}

\textbf{Format}

\begin{verbatim}
wait [pid|job-id ...]
\end{verbatim}

tcsh shell: \texttt{wait}

\textbf{Description}

\texttt{wait} waits for one or more jobs or child processes to complete in the background. If you specify one or more \texttt{job-id} arguments, \texttt{wait} waits for all processes in each job to end. If you specify \texttt{pid}, \texttt{wait} waits for the child process with that process ID (PID) to end. If no child process has that process ID, \texttt{wait} returns immediately.

If you specify neither a \texttt{pid} nor a \texttt{job-id}, \texttt{wait} waits for the process IDs known to the invoking shell to complete.

In the tcsh shell, the shell waits for all background jobs. If the shell is interactive, an interrupt disrupts the \texttt{wait} and cause the shell to print the names and job numbers of all outstanding jobs. See \texttt{"tcsh: Invoke a C shell"} on page 671.

\textbf{Localization}

\texttt{wait} uses the following localization environment variables:

• \texttt{LANG}
• \texttt{LC_ALL}
• \texttt{LC_CTYPE}
• \texttt{LC_MESSAGES}
• \texttt{NLSPATH}

See \texttt{Appendix F, "Localization,"} on page 965 for more information.

\textbf{Usage notes}

\texttt{wait} is a built-in shell command.
**Exit values**

If one or more arguments (pid or job-id) are specified, the exit status of `wait` is the exit status of the last argument.

If you specified a job-id that has terminated or is unknown by the invoking shell, an error message and a return code of 127 is returned. If you specified a pid that has terminated or is unknown to the shell, a return code of 127 is returned. If a signal ended the process abnormally, the exit status is a value greater than 128 unique to that signal. Otherwise, possible exit status values are:

0
   Successful completion or `wait` was invoked with no arguments, and all child processes known to the invoking shell have completed.

1–126
   An error occurred.

127
   A specified `pid` or `job-id` has terminated or is unknown by the invoking shell.

**Portability**

POSIX.2, X/Open Portability Guide, UNIX systems.

**Related information**

sleep, tcsh

---

**wall - Broadcast a message to logged-in users**

**Format**

`wall [message]`

**Description**

`wall` sends a message to all logged-in users. The `wall` command reads from the standard input (stdin). Type each line, pressing Enter after each. After you finish typing the message, enter End-of-File or an interrupt (typically, `<EscChar-D>` for End-of-File or `<EscChar-C>` for an interrupt, where EscChar is normally the cent sign; if you use rlogin or telnet to enter the shell, hold down the Ctrl key while pressing either D or C).

You must be a superuser to ensure permission to write to all the ttys that are logged in. If you are not a superuser, then writes to all ttys will fail (except your own) and those users will not receive the message. Superusers can also get failures if the `/etc/utmpx` file does not correctly represent the users who are currently logged in.

Recipients of the message receive a beep that announces the message. The message is displayed in this form:

```
Broadcast Message from SWEHR@AQFT (tty0006) at 10:43:54 (EDT5EST)...
This is the text of the message line1.
This is line2.
```

**Exit values**

0
   `wall` successfully sent the message to all users.

1
   Failure due to any of the following reasons:
• No message was entered in response to the prompt.
• You do not have permission to write to a user's terminal.

**wc - Count newlines, words, and bytes**

**Format**

```wc [-Blw] [-c|m] [file ... file]```

**Description**

*wc* tells you how large a text document is. It counts the number of newlines, words, characters, and bytes in text files. If you specify multiple files, *wc* produces counts for each file, plus totals for all files. If you do not specify any files, *wc* reads from the standard input (stdin).

**Options**

- **-B**
  Disables the automatic conversion of tagged files. This option is ignored if the *filecodeset* or *pgmcodeset* options (-W option) are specified.

- **-c**
  Prints a byte count. You cannot specify this option with -m.

- **-l**
  Prints a <newline> count

- **-m**
  Prints a character count. You cannot specify this option with -c.

- **-w**
  Prints a word count

- **-W option[,option]...**
  Specifies z/OS-specific options. The option keywords are case-sensitive. Possible options are:

  **filecodeset=** *codeset*
  Performs text conversion from one code set to another when reading from the file. The coded character set of the file is *codeset*. *codeset* can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). Note that the command `iconv -l` lists existing CCSIDs along with their corresponding code set names. The *filecodeset* and *pgmcodeset* options can be used on files with any file tag.

  If *pgmcodeset* is specified but *filecodeset* is omitted, then the default file code set is ISO8859-1 even if the file is tagged with a different code set. If neither *filecodeset* nor *pgmcodeset* is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

  If *filecodeset* or *pgmcodeset* is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See `z/OS UNIX System Services Planning` for more information about automatic conversion.

  When specifying values for *filecodeset*, use the values that Unicode Service supports. For more information about supported code sets, see `z/OS Unicode Services User's Guide and Reference`.

  **pgmcodeset=** *codeset*
  Performs text conversion from one code set to another when reading from the file. The coded character set of the program (command) is *codeset*. *codeset* can be a code set name that is known to the system or a numeric coded character set identifier (CCSID). The command `iconv -l` lists
existing CCSIDs along with their corresponding code set names. The filecodeset and pgmcodeset options can be used on files with any file tag.

If filecodeset is specified but pgmcodeset is omitted, then the default program code set is IBM-1047. If neither filecodeset nor pgmcodeset is specified, text conversion will not occur unless automatic conversion is enabled or the _TEXT_CONV environment variable indicates text conversion. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

If filecodeset or pgmcodeset is specified, then automatic conversion is disabled for this command invocation and the -B option is ignored if it is also specified. See z/OS UNIX System Services Planning for more information about automatic conversion.

The only supported values for pgmcodeset are IBM-1047 and 1047.

The order of options can dictate the order in which wc displays counts. For example, `wc -cwl` displays the number of bytes, then the number of words, then the number of <newline>s. If you do not specify any options, the default is `wc –lwc` (<newline> count, then words, then bytes).

A word is considered to be a character or characters delimited by white space.

**Note:** If you have a file containing double-byte characters, the byte count is higher than the character count.

**Examples**

1. To display the <newline> count, followed by the word count, followed by the byte count of a text file to the standard output (stdout):
   
   ```sh
   wc myTextFile
   ```

2. To display a byte count followed by a word count of a text file containing ASCII characters to the standard output (stdout), assuming that:
   
   - The text file is untagged and you do not want to tag it or enable automatic conversion, and
   - You cannot alter the tag (for example, you are displaying an untagged public text file or a read-only text file):

   ```sh
   wc -cw -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 myAsciiFile
   ```

3. To display the <newline> count of a text file containing EBCDIC characters, assuming that automatic conversion has been enabled but the text file is incorrectly tagged as UTF-8:

   ```sh
   wc -lB myMisTaggedFile
   ```

**Localization**

wc uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- NLSPATH


**Environment variables**

wc uses the following environment variable:
Contains text conversion information for the command. The text conversion information is not used when either the -B option or the filecodeset or pgmcodeset option (-W option) is specified. For more information about text conversion, see Appendix L, “Controlling text conversion for z/OS UNIX shell commands,” on page 993.

Exit values

0
Successful completion.

1
Failure due to any of the following reasons:
- Inability to open the input file.
- The code set is not valid.
- Could not turn off automatic conversion.
- Could not perform requested text conversion.

2
Failure because of an incorrect command-line option.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

The way the order of options -x, -l and -w affects the order of display is an extension to traditional implementations of wc. The -B and -W options are extensions of the POSIX standard.

Related information

awk, ed, vi

whence - Tell how the shell interprets a command name

Format

whence [-v] name ...

Description

whence tells how the shell would interpret each name if used as a command name. Shell keywords, aliases, functions, built-in commands, and executable files are distinguished. For executable files, the full path name is given. If the executable file is a tracked alias, the string identifies it as cached.

Options

-v
Gives a more verbose report.

Usage notes

whence is a built-in shell command.

Localization

whence uses the following localization environment variables:
- LANG
who

Exit values

0  
Successful completion.

1  
Command name could not be found.

2  
Failure due to an incorrect command-line argument.

Portability

POSIX.2.

Related information

command, sh

who - Display information about current users

Format

who[-AaBbHilmprsTtuw] [file]
who -q[file]
who am I|i

Description

who displays information about users who are logged in to the system. By default, the output contains the user's login name, terminal name, and the time that the user logged in. Normally, who consults the file /etc/utmpx for information, but you can use the file argument to specify another accounting file.

When called as:

who am i

or

who am I

who displays your login name, terminal, and login time. This command works only in the POSIX locale.

Options

-A  
Displays all accounting entries.

-a  
Displays all types of entries. This is equivalent to specifying -AbdHilmprsTtuw.

-b  
Displays all entries that are written at system boot time.
-d Displays entries that are produced after the death of a process that is spawned from /usr/sbin/init.

-H Displays column headings above the output.

-i Displays idle time for users. The idle time is the hours:minutes since the last activity; a dot (.) means that the terminal has been used in the last minute, and the string old means that the terminal has not been used in more than 24 hours, or hasn't been used since boot time.

-l Displays logged-out user entries.

-m Displays information about current terminal only.

-p Displays entries for processes that are spawned from /usr/sbin/init.

-q Displays a quick list with the number of users and their names; other options are ignored.

-r Displays all run-level change entries.

-s Displays only the three fields: user name, terminal, and time of entry.

-T Displays the state of each terminal as a plus sign (+) if the terminal allows write access to other users, and a minus sign (-) if write access is denied. who displays a question mark (?) if the write access cannot be determined.

-t Displays all time change entries (both old and new time).

-u Displays only entries that are associated with logged-in users. who enables this option when you do not provide any options on the command line.

-w Displays the terminal state; this indicates whether the terminal can be written to.

Files
who uses the following files:

/etc/utmpx
   Displays the current status file.

Localization
who uses the following localization environment variables:
   • LANG
   • LC_ALL
   • LC_CTYPE
   • LC_MESSAGES
   • LC_TIME
   • NLSPATH
Exit values

0
  Successful completion.

2
  Failure because of an incorrect command-line option, or because of too many command-line arguments.

Portability

POSIX.2 User Portability Extension, X/Open Portability Guide.
The utmpx file format, the options, and the output of who are compatible with UNIX System V.
The -A, -a, -b, -d, -l, -p, -r, -s, -t, w, and am I options are extensions to the POSIX standard.

Related information

See the utmpx file format description in Appendix H, “File formats,” on page 969 for more information.

whoami - Display your effective user name

Format

whoami

Description

whoami displays a user name associated with the effective user ID. To display your login name, use who am i.

For example, if you login as user1, then use the su command to change to user2:

<table>
<thead>
<tr>
<th>Command</th>
<th>Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>who am I</td>
<td>user1</td>
</tr>
<tr>
<td>who am I</td>
<td>user2</td>
</tr>
</tbody>
</table>

Exit values

0
  Successful completion.

1
  Incorrect command-line argument.

2
  Error getting effective user name; displays effective UID.

Related information

who, id

write - Write to another user

Format

write user_name [terminal]
Description

**write** lets you send a message directly to the terminal of someone else logged in to the system. It reads from the standard input (stdin) and writes to the terminal of another user.

Options

**user_name**

Specifies the user to whom you want to send your message.

**terminal**

Is an optional identifier for use when the other user is logged in on more than one terminal. The format of the terminal name is the same as returned by **who**.

Usage notes

1. When you issue a **write** command to send a message to another user, the other user receives a message of the form:

   ```
   Message from your_name (terminal) [date] ...
   ```

   After the system establishes the connection to the other user, it sends two alert characters (typically beeps) to your terminal to tell you that it is ready to send your message. You can then type your message, which will appear on the other user's terminal. To end your message, enter end-of-file or an interrupt (typically, <EscChar-D> for end-of-file or <EscChar-C> for an interrupt, where EscChar is normally the cent sign; if you use rlogin or telnet to enter the shell, you hold down the Ctrl key while you press either D or C). When **write** receives an indication for end-of-message, it tells the other user that the message is over and breaks the connection.

   The other user can reply to your message with:

   ```
   write your_user_name
   ```

   However, if both of you are trying to write on each other's terminal at the same time, the messages might get interleaved on your screens, making them difficult to read. For two-way conversations, use **talk** instead of **write**.

2. You can add the output of a command to the material that you write. To do this, start a line with an exclamation mark (!) and put a standard system command on the rest of that line. **write** calls your shell to execute the command, and sends the standard output (stdout) from the command to the other user. The other user does not see the command itself or any input to the command. For example, you might write:

   ```
   Here is what my file contains:
   !cat file
   ```

3. The **mesg** command lets you refuse **write** messages. With:

   ```
   mesg n
   ```

   you can tell the system that you don't want to be interrupted by **write** messages. If people try to **write** to you, they are denied immediately; the system does not inform you about such attempts. For further details, see **mesg**.

Localization

**write** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
Exit values

0
write successfully wrote a message, or the intended recipient used mesg to refuse messages (either before you start sending a message or as you are sending the message).

1
Failure due to any of the following reasons:

• user_name is not signed on.
• You do not have permission to write on that user's terminal.
• write cannot open the target terminal for writing.
• The command line had an incorrect number of options.

Portability
POSIX.2 User Portability Extension, X/Open Portability Guide, UNIX systems.

Related information
mailx, mesg, talk, who

**writedown - Set or display user's write-down mode**

Format

writedown -a | -d | -i [-p]
writedown -p

Description

writedown sets or displays the user's write-down mode for the current address space. Setting or querying the write-down mode is only allowed if multilevel security is active and the user has "write-down" privilege. See *z/OS Planning for Multilevel Security and the Common Criteria* for more information about multilevel security.

Options

-a
 Activate write-down mode. This allows the user to write data to a resource protected by a multilevel security label of lower-labeled classification than the user's security label.

-d
 Set the write-down mode from the default value in the user's security profile.

-i
 Deactivate write-down mode. This prevents the user from writing data to a resource protected by a multilevel security label of lower labeled classification than the user's security label.

-p
 Print the user's current write-down mode setting to stdout. The output is "active" or "inactive". If used with -a, -d, or -i, the new value is displayed.
Usage notes

1. This command is only supported when the user has at least READ access to the IRR.WRITEDOWN.BYUSER resource in the FACILITY class and SETR MLS is active.

2. Write-down mode affects the address space of the current process. When the write-down mode is changed, all processes running in the same address space will get the new write-down setting, until the shell (where writedown was invoked) exits.

3. **writedown** is a built-in shell command in **sh** and **tcsh**. It affects the security setting for commands that are issued by the current shell, and by child processes, such as shell scripts.

4. See **z/OS Planning for Multilevel Security and the Common Criteria** for more information about write-down mode, multilevel security, and seclabels.

Exit values

The exit values for **/bin/sh** are as follows:

0  Successful completion.

1  Failure due to any of the following reasons:
   • SETR MLS is not active.
   • User does not have at least READ access to IRR.WRITEDOWN.BYUSER resource in the FACILITY class.

2  Command syntax error.

The exit values for **/bin/tcsh** are as follows:

0  Successful completion.

1  Failure due to any of the following reasons:
   • SETR MLS is not active.
   • User does not have at least READ access to IRR.WRITEDOWN.BYUSER resource in the FACILITY class.
   • Command syntax error.

Examples

1. To display your current write-down mode:

   ```
   > writedown -p
   inactive
   ```

2. To activate and display your current write-down mode:

   ```
   > writedown -ap
   active
   ```

Related information

'id', 'sh', 'tcsh'

**xlc:** Compiler invocation using a customizable configuration file
Invocation commands

The xlc utility provides two basic compiler invocation commands, `xlc` and `xlC (xlc++)`, along with several other compiler invocation commands to support various C/C++ language levels and compilation environments. In most cases, you would use the `xlc` command to compile C source files and `xlC (xlc++)` command to compile C++ source files.

You can however, use other forms of the command if your particular environment requires it. The various compiler invocation commands for C are:

- `xlc`
- `cc`
- `c89`
- `c99`
- `xlc_x`
- `cc_x`
- `c89_x`
- `c99_x`
- `xlc_64`
- `cc_64`
- `c89_64`
- `c99_64`

The various compiler invocation commands for C++ are:

- `xlC (xlc++)`
- `cxx`
- `c++`
- `xlc_x (xlc++_x)`
- `c++_x`
- `cxx_x`
- `xlC_64 (xlc++_64)`
- `c++_64`
- `cxx_64`

The two basic compiler invocation commands appear as the first entry of each of these list items. Select an invocation command using the following criteria:

**xlc**

Invokes the compiler for C source files with a default language level of ANSI, the compiler option `-qansialias` to allow type-based aliasing, and the compiler option `-qcpluscmt` to allow C++ style comments (/ /).

**xlC (xlc++)**

Invokes the compiler so that source files are compiled as C++ language source code.

Files with .c suffixes, assuming you have not used the -+ compiler option, are compiled as C language source code with a default language level of ANSI, and compiler option `-qansialias` to allow type-based aliasing.

If any of your source files are C++, you must use this invocation to link with the correct runtime libraries.

**cc**

Invokes the compiler for C source files with a default language level of extended and compiler options `-qnoro` and `-qnozconst` (to provide placement of string literals or constant values in read/write storage).
Use this invocation for legacy C code that does not require compliance with ANSI C. This invocation is intended to provide the same compiler behavior as when invoked by the `cc` command name of the c89 utility.

**c89**

Invokes the compiler for C source files, with a default language level of ANSI, and specifies compiler options `-qansi` (to allow type-based aliasing) and `-qnolonglong` (disabling use of long long). Use this invocation for strict conformance to the ISO/IEC 9899:1990 standard. This invocation is intended to provide the same compiler behavior as when invoked by the **c89** command name of the c89 utility.

**c99**

Invokes the compiler for C source files, with a default language level of STDC99 and specifies compiler option `-qansi` (to allow type-based aliasing). Use this invocation for strict conformance to the ISO/IEC 9899:1999 standard.

**cxx/c++**

The `cxx` and `c++` commands invoke the compiler for C++ language source code. Both are intended to provide the same compiler behavior as when invoked using the `cxx` and `c++` command names of the c89 utility.

You can combine the previously described command names with the following suffixes:

**_-x**

Command invocations using command names with suffix `_x` are the same as invocations using names without suffixes, except the `-qxplink` option is also specified and appropriate XPLINK libraries are used in the link step. If you are building an XPLINK application, you no longer need to use command names with suffix `_x` to link with the correct runtime libraries. This can be achieved through the new configuration attributes that have been introduced to enable XPLINK behavior without the use of suffixes. See “Configuration file attributes” on page 854 for further information.

**_-64**

Command invocations using command names with suffix `_64` are the same as invocations using names without suffixes, except the `-q64` option is also specified and appropriate 64-bit libraries are used in the link step. If you are building a 64-bit application, you no longer need to use command names with suffix `_64` to link with the correct runtime libraries. This can be achieved through the new configuration attributes that have been introduced to enable 64-bit behavior without the use of suffixes. See “Configuration file attributes” on page 854 for further information.

**Notes:**

1. Suffixes are used as a naming convention and do not enforce behavior. The content of the command line will take precedence over the suffixes.
2. When compiling and linking a C++ application using a single command line invocation, the application will be correctly link edited with any stanza if at least one C++ source file is specified on the command line. If only object files or a mix of C sources and C++ object files are specified on the command line, a C++ stanza must be used to correctly link edit the application.

**Setting up the compilation environment**

Before you compile your C and C++ programs, you must set up the environment variables and the configuration file for your application.

**Environment variables**

You can use environment variables to specify necessary system information.

Before using the compiler, you must install the message catalogs and set the environment variables:

**LANG**

Specifies the national language for message and help files.

**NLS_PATH**

Specifies the path name of the message and help files.
The LANG environment variable can be set to any of the locales provided on the system. See the description of locales in z/OS XL C/C++ Programming Guide for more information.

The national language code for United States English may be En_US or C. If the Japanese message catalog has been installed on your system, you can substitute Ja_JP for En_US.

To determine the current setting of the national language on your system, see the output from both of the following echo commands:

- echo $LANG
- echo $NLSPATH

The LANG and NLSPATH environment variables are initialized when the operating system is installed, and may differ from the ones you want to use.

Environment variables for OpenMP

If you use OpenMP constructs for parallelization, you can specify runtime options using the OMP environment variables.

OpenMP runtime options affecting parallel processing are set by specifying OMP environment variables. These environment variables use syntax of the form:

$$\text{env_variable} = \text{option_and_args}$$

If an OMP environment variable is not explicitly set, its default setting is used.

For information about the OpenMP specification, see OpenMP (www.openmp.org).

**OMP_DYNAMIC**

The OMP_DYNAMIC environment variable enables or disables dynamic adjustment of the number of threads available for running parallel regions.

If it is set to TRUE, the number of threads available for executing parallel regions can be adjusted at run time to make the best use of system resources.

If it is set to FALSE, dynamic adjustment is disabled.

The default setting is TRUE.

**OMP_MAX_ACTIVE_LEVELS**

Use OMP_MAX_ACTIVE_LEVELS to set the `max-active-levels-var` internal control variable. This controls the maximum number of active nested parallel regions. The syntax is as follows:

$$\text{OMP_MAX_ACTIVE_LEVELS} = n$$

where \( n \) is the maximum number of nested active parallel regions. It must be a positive scalar integer. The maximum number that you can specify is 5.

In programs where nested parallelism is disabled, the initial value is 1. In programs where nested parallelism is enabled, the initial value is greater than 1. The function `omp_get_max_active_levels` can be used to retrieve this value at run time.

**OMP_NUM_THREADS**

The OMP_NUM_THREADS environment variable specifies the number of threads to use for parallel regions.

The syntax of the environment variable is as follows:

$$\text{OMP_NUM_THREADS} = \text{num_list}$$
num_list

A list of one or more positive integer values separated by commas.

If you do not set OMP_NUM_THREADS, the number of processors available is the default value to form a new team for the first encountered parallel construct. If nested parallelism is disabled, any nested parallel constructs are run by one thread by default.

If num_list contains a single value, dynamic adjustment of the number of threads is enabled (OMP_DYNAMIC is set to true), and a parallel construct without a num_threads clause is encountered, the value is the maximum number of threads that can be used to form a new team for the encountered parallel construct.

If num_list contains a single value, dynamic adjustment of the number of threads is not enabled (OMP_DYNAMIC is set to false), and a parallel construct without a num_threads clause is encountered, the value is the exact number of threads that can be used to form a new team for the encountered parallel construct.

If num_list contains multiple values, dynamic adjustment of the number of threads is enabled (OMP_DYNAMIC is set to true), and a parallel construct without a num_threads clause is encountered, the first value is the maximum number of threads that can be used to form a new team for the encountered parallel construct. After the encountered construct is entered, the first value is removed and the remaining values form a new num_list. The new num_list is in turn used in the same way for any closely nested parallel constructs inside the encountered parallel construct.

If num_list contains multiple values, dynamic adjustment of the number of threads is not enabled (OMP_DYNAMIC is set to false), and a parallel construct without a num_threads clause is encountered, the first value is the exact number of threads that can be used to form a new team for the encountered parallel construct. After the encountered construct is entered, the first value is removed and the remaining values form a new num_list. The new num_list is in turn used in the same way for any closely nested parallel constructs inside the encountered parallel construct.

**Note:** If the number of parallel regions is equal to or greater than the number of values in num_list, the omp_get_max_threads function returns the last value of num_list in the parallel region.

If the number of threads requested exceeds the system resources available, the program stops.

The omp_set_num_threads function sets the first value of num_list. The omp_get_max_threads function returns the first value of num_list.

If you specify the number of threads for a given parallel region more than once with different settings, the compiler uses the following precedence order to determine which setting takes effect:

1. The number of threads set using the num_threads clause takes precedence over that set using the omp_set_num_threads function.
2. The number of threads set using the omp_set_num_threads function takes precedence over that set using the OMP_NUM_THREADS environment variable.

See the following example:

```bash
export OMP_NUM_THREADS=3,4,5
export OMP_DYNAMIC=false

// omp_get_max_threads() returns 3
#pragma omp parallel
{
  // Three threads running the parallel region
  // omp_get_max_threads() returns 4
  #pragma omp parallel if(0)
  {
    // One thread running the parallel region
    // omp_get_max_threads() returns 5
    #pragma omp parallel
    {
      // Five threads running the parallel region
      // omp_get_max_threads() returns 5
    }
  }
}
```
OMP_PROC_BIND

The OMP_PROC_BIND environment variable controls whether OpenMP threads can be moved between processors. The syntax is as follows:

```
OMP_PROC_BIND=\{TRUE\, FALSE\}
```

By default, the OMP_PROC_BIND environment variable is not set. If you set OMP_PROC_BIND to TRUE, the threads are bound to processors. If you set OMP_PROC_BIND to FALSE, the threads may be moved between processors.

**Note:** The OMP_PROC_BIND environment variable provides a portable way to control whether OpenMP threads can be migrated.

OMP_SCHEDULE

The OMP_SCHEDULE environment variable specifies the scheduling algorithm used for loops with the `omp schedule(runtime)` clause.

For example:

```
OMP_SCHEDULE="guided, 4"
```

Valid options for `algorithm` are:

- auto
- dynamic[n]
- guided[n]
- runtime
- static[n]

If specifying a chunk size with `n`, the value of `n` must be a positive integer.

The default scheduling algorithm is **auto**.

OMP_STACKSIZE

The OMP_STACKSIZE environment variable indicates the stack size of threads created by the OpenMP runtime. OMP_STACKSIZE sets the value of the `stacksize-var` internal control variable. OMP_STACKSIZE does not control the stack size of the master thread. The syntax is as follows:

```
OMP_STACKSIZE=\{size\}
```

By default, the size value is represented in Kilobytes. You can also use the suffixes B, K, M, or G if you want to indicate the size in Bytes, Kilobytes, Megabytes, or Gigabytes respectively. White space is allowed between and around the size value and the suffix. For example, the following examples both indicate a stack size of 10 Megabytes.

```
setenv OMP_STACKSIZE 10M
setenv OMP_STACKSIZE " 10 M "
```

If OMP_STACKSIZE is not set, the initial value of the `stacksize-var` internal control variable is set to the default value. The default value for 32-bit mode is 256M. For 64-bit mode, the default is up to the limit imposed by system resources.

If the compiler cannot use the stack size specified or if OMP_STACKSIZE does not conform to the correct format, the compiler sets the environment variable to the default value.
**OMP_THREAD_LIMIT**

The OMP_THREAD_LIMIT environment variable sets the number of OpenMP threads to use for the whole program. The syntax is as follows:

\[ \text{OMP\_THREAD\_LIMIT} = n \]

- **n** - The number of OpenMP threads to use for the whole program. It must be a positive scalar integer.

The value for OMP_THREAD_LIMIT is a positive integer. When nested parallelism is enabled, the value you specify for OMP_THREAD_LIMIT can affect the behavior of a parallel region. For example, if the value of OMP_THREAD_LIMIT is much smaller than the number of threads required in the program, say OMP_THREAD_LIMIT=1, the parallel region is run sequentially rather than in parallel.

If the OMP_THREAD_LIMIT environment variable is not set and the OMP_NUM_THREADS environment variable is set to a single value, the default value for OMP_THREAD_LIMIT is the value of OMP_NUM_THREADS or the number of available processors, whichever is greater.

If the OMP_THREAD_LIMIT environment variable is not set and the OMP_NUM_THREADS environment variable is set to a list, the default value for OMP_THREAD_LIMIT is the multiplication of all the numbers in the list or the number of available processors, whichever is greater.

If the OMP_THREAD_LIMIT and OMP_NUM_THREADS environment variables are both not set, the default value for OMP_THREAD_LIMIT is the number of available processors.

**OMP_WAIT_POLICY**

The OMP_WAIT_POLICY environment variable gives hints to the compiler about the preferred behavior of waiting threads during program run time. The OMP_WAIT_POLICY environment variable sets the *wait-policy-var* internal control variable value.

The syntax is as follows:

\[ \text{OMP\_WAIT\_POLICY} = \text{PASSIVE} \quad \text{ACTIVE} \]

- The default value for OMP_WAIT_POLICY is PASSIVE.

Use ACTIVE if you want waiting threads to be mostly active. With ACTIVE, the thread consumes processor cycles while waiting, if possible.

Use PASSIVE if you want waiting threads to be mostly passive. That is, the preference is for the thread to not consume processor cycles while waiting. For example, you prefer waiting threads to sleep or to yield the processor to other threads.

**Setting up a configuration file**

The configuration file specifies information that the compiler uses when you invoke it. This file defines values used by the compiler to compile C or C++ programs. You can make entries to this file to support specific compilation requirements or to support other C or C++ compilation environments.

A configuration file is a UNIX file consisting of named sections called stanzas. Each stanza contains keywords called configuration file attributes, which are assigned values. The attributes are separated from their assigned value by an equal sign. A stanza can point to a default stanza by specifying the "use" keyword. This allows specifying common attributes in a default stanza and only the deltas in a specific stanza, referred to as the local stanza.

For any of the supported attributes not found in the configuration file, the xlc utility uses the built-in defaults. It uses the first occurrence in the configuration file of a stanza or attribute it is looking for. Unsupported attributes, and duplicate stanzas and attributes are not diagnosed.

**Note:**
1. The difference between specifying values in the stanza and relying on the defaults provided by the xlc utility is that the defaults provided by the xlc utility will not override pragmas.

2. Any entry in the configuration file must occur on a single line. You cannot continue an entry over multiple lines.

Configuration file attributes

A stanza in the configuration file can contain the following attributes:

acceptable_rc
Enables you to specify a number that represents a return code value for a program invoked by the xlc utility. The xlc utility does not place any restriction on the value assigned to the acceptable_rc attribute. acceptable_rc can appear as part of any stanza in the configuration file.

Note: If the acceptable_rc attribute is not specified in the configuration file, the xlc utility will assign the value from a c89 prefix_ACCEPTABLE_RC environment variable, if it is exported, to the acceptable_rc, otherwise it will default to 4. The command name used to invoke the xlc utility determines the prefix that the xlc utility will use when looking for a prefix_ACCEPTABLE_RC environment variable. For example, if the xlc utility is invoked using the xlC command name, the xlc utility will look for _CXX_ACCEPTABLE_RC and, if found, use it. If the acceptable_rc attribute is specified in the configuration file, the xlc utility will use the value specified in the configuration file and will ignore an exported prefix_ACCEPTABLE_RC environment variable.

as
Path name to be used for the assembler. The default is /bin/c89.

asopt
The list of options for the assembler and not for the compiler. These override all normal processing by the compiler and are directed to the assembler specified in the as attribute. Options are specified following the c89 utility syntax.

asuffix
The suffix for archive files. The default is a.

asuffix_host
The suffix for archive data sets. The default is LIB.

ccomp
The C compiler. The default is usr/lpp/cbclib/xlc/exe/ccndrvr.

cinc
A comma separated list of directories or data set wild cards used to search for C header files. The default for this attribute is: -I'/CEE.SCEEH.+'. For further information on the list of search places used by the compiler to search for system header files, see the note at the end of this list of configuration file attributes.

classversion
The USL class library version. The default matches the current release, as described in the TARGET compiler option description in z/OS XL C/C++ User’s Guide.

cppcomp
The C++ compiler. The default is /usr/lpp/cbclib/xlc/exe/ccndrvr.

cppinc
A comma separated list of directories or data set wild cards used to search for C++ header files. The default for this attribute is: -I'/CEE.SCEEH.+','-I'/CBC.SCLBH.+'. For further information on the list of search places used by the compiler to search for system header files, see the note at the end of this list of configuration file attributes.

csuffix
The suffix for source programs. The default is c (lowercase c).

csuffix_host
The suffix for C source data sets. The default is C (uppercase C).
**cversion**

The compiler version. The default matches the current release, as described in the TARGET compiler option description in *z/OS XL C/C++ User's Guide*. The oldest release supported is z/OS V1R6.

**cxxsuffix**

The suffix for C++ source files. The default is C (uppercase C).

**cxxsuffix_host**

The suffix for C++ source data sets. The default is CXX.

**exportlist**

A colon separated list of data sets with member names indicating definition side-decks to be used to resolve symbols during the link-editing phase. This attribute is only used for compatibility with configuration files that are defined using the z/OS V1R6 release. Attributes with an appropriate suffix should be used instead (see descriptions for exportlist attributes with a suffix). The default for this attribute should match the type of stanza for which it is specified.

Suffix-less C stanzas do not have a default.

The default for suffix-less C++ stanzas is:

```
CEE.SCEELIB(C128N):CBC.SCLBSID(IOSTREAM,COMPLEX)
```

The default for C stanzas with an _x suffix is:

```
CEE.SCEELIB(CELHS003,CELHS001)
```

The default for C++ stanzas with an _x suffix is:

```
CEE.SCEELIB(CELHS003,CELHSCPP,CELHS001,C128):CBC.SCLBSID(IOSTREAM,COMPLEX)
```

The default for C stanzas with a _64 suffix is:

```
CEE.SCEELIB(CELQS003)
```

The default for C++ stanzas with a _64 suffix is:

```
CEE.SCEELIB(CELQS003,CELQSCPP,C64):CBC.SCLBSID(IOSQ64)
```

**exportlist_c**

A colon separated list of data sets with member names indicating definition side-decks to be used to resolve symbols during the link-editing phase of non-XPLINK C applications. The default for this attribute is NONE.

**exportlist_cpp**

A colon separated list of data sets with member names indicating definition side-decks to be used to resolve symbols during the link-editing phase of non-XPLINK C++ applications. The default for this attribute is:

```
CEE.SCEELIB(C128n):CBC.SCLBSID(IOSTREAM,COMPLEX)
```

**exportlist_c_x**

A colon separated list of data sets with member names indicating definition side-decks to be used to resolve symbols during the link-editing phase of XPLINK C applications. The default for this attribute is:

```
CEE.SCEELIB(CELHS003,CELHS001)
```
**exportlist_cpp_x**
A colon separated list of data sets with member names indicating definition side-decks to be used to resolve symbols during the link-editing phase of XPLINK C++ applications. The default for this attribute is:

```
CEE.SCEELIB(CELHS003,CELHSCPP,CELHS001,C128):CBC.SCLBSID
(ISTREAM,COMPLEX)
```

**exportlist_c_64**
A colon separated list of data sets with member names indicating definition side-decks to be used to resolve symbols during the link-editing phase of 64-bit C applications. The default for this attribute is:

```
CEE.SCEELIB(CELQS003)
```

**exportlist_cpp_64**
A colon separated list of data sets with member names indicating definition side-decks to be used to resolve symbols during the link-editing phase of 64-bit C++ applications. The default for this attribute is:

```
CEE.SCEELIB(CELQS003,CELQSCPP,C64):CBC.SCLBSID(IOSQ64)
```

**isuffix**
The suffix for C preprocessed files. The default is `i`.

**isuffix_host**
The suffix for C preprocessed data sets. The default is CEX.

**ilsuffix**
The suffix for IPA output files. The default is `I`.

**ilsuffix_host**
The suffix for IPA output data sets. The default is IPA.

**ixxsuffix**
The suffix for C++ preprocessed files. The default is `i`.

**ixxsuffix_host**
The suffix for C++ preprocessed data sets. The default is CEX.

**ld**
The path name to be used for the binder. The default is `/bin/c89`.

**ld_c**
The path name to be used for the binder when only C sources appear on the command line invoked with a C stanza. The default is: `/bin/c89`.

**ld_cpp**
The path name to be used for the binder when at least one C++ source appears on the command line, or when a C++ stanza is used. The default is: `/bin/cxx`.

**libraries**
A string of option flags, separated by commas, to be processed by the compiler as if they had been entered on the command line.

**libraries2**
A string of option flags, separated by commas, to be processed by the compiler as if they had been entered on the command line.

**osuffix**
The suffix for object files. The default is `.o`.
**osuffix_host**
The suffix for object data sets. The default is OBJ.

**psuffix**
The suffix for prelinked files. The default is p.

**psuffix_host**
The suffix for prelinked data sets. The default is CPOBJ.

**pversion**
The runtime library version. The default matches the current release, as described in the TARGET compiler option description in *z/OS XL C/C++ User’s Guide*.

**ssuffix**
The suffix for assembler files. The default is .s.

**ssuffix_host**
The suffix for assembler data sets. The default is ASM.

**steplib**
A colon separated list of data sets or keyword NONE used to set the STEPLIB environment variable. The default is NONE, which causes all programs to be loaded from LPA or linklist.

**syslib**
A colon separated list of data sets used to resolve runtime library references. Data sets from this list are used to construct the SYSLIB DD for the IPA Link and the binder invocation for non-XPLINK applications. For compatibility with configuration files defined using the z/OS V1R6 release, this attribute is also used with XPLINK applications as a fallback when the syslib_x attribute is not specified. When the syslib_x attribute is not specified, the default for this attribute should match the type of stanza for which it is specified. When the syslib_x attribute is specified, the default for this attribute matches the default for suffix-less stanzas.

The default for suffix-less stanzas is:

```
CEE.SCEELKEX:CEE.SCEELKED:CBC.SCCNOBJ:SYS1.CSSLIB
```

The default for stanzas with _x and _64 suffixes is:

```
CEE.SCEEBND2:CBC.SCCNOBJ:SYS1.CSSLIB
```

**syslib_x**
A colon separated list of data sets used to resolve runtime library references. Data sets from this list are used to construct the SYSLIB DD for the IPA Link and the binder invocation when building XPLINK applications (31-bit and 64-bit).

The default for this attribute is:

```
CEE.SCEEBND2:CBC.SCCNOBJ:SYS1.CSSLIB
```

**sysobj**
A colon separated list of data sets containing object files used to resolve runtime library references. Data sets from this list are used to construct the LIBRARY control statements and the SYSLIB DD for the IPA Link and the binder invocation. This attribute is ignored for XPLINK and 64-bit applications.

The default is:

```
CEE.SCEE0BJ:CEE.SCEECPP
```

**use**
Values for attributes are taken from the named stanza and from the local stanza. For single-valued attributes, values in the use stanza apply if no value is provided in the local, or default stanza. For comma-separated lists, the values from the use stanza are added to the values from the local stanza.

**usuffix**
The suffix for make dependency file names. The default make dependency file name suffix is ".u", but it is overwritten by the value assigned to this attribute.
There is no host version of this attribute, because make dependency feature only applies to z/OS UNIX files.

**xlc**
The path name of the C++ compiler invocation command. The default is /usr/lpp/cbclib/xlc/bin/xlc.

**xlCcopt**
A string of option flags, separated by commas, to be processed when the xlc command is used for compiling a C file.

**xsuffix**
The suffix for definition side-deck files. The default is x.

**xsuffix_host**
The suffix for definition side-deck data sets. The default is EXP.

**Note:** When using the xlc utility to invoke the compiler, the compiler uses the following list of search places to search for system header files:

- If the `-qnosearch` option is not specified on the command line or in the configuration file:
  1. search places defined in the customizable defaults module (CCNEDFLT)
  2. followed by those specified on the command line using the `-I` flag option
  3. followed by those specified in the configuration file
- If the `-qnosearch` is specified only in the configuration file:
  1. search places specified on the command line using the `-I` flag option
  2. followed by those specified in the configuration file
- If the `-qnosearch` option is specified on the command line:
  1. search places specified on the command line following the last specified `-qnosearch` option
  2. followed by those specified in the configuration file

**Tailoring a configuration file**
The default configuration file is installed in /usr/lpp/cbclib/xlc/etc/xlc.cfg.

You can copy this file and make changes to the copy to support specific compilation requirements or to support other C or C++ compilation environments. The `-F` option is used to specify a configuration file other than the default. For example, to make `-qnoro` the default for the xlc compiler invocation command, add `-qnoro` to the xlc stanza in your copied version of the configuration file.

You can link the compiler invocation command to several different names. The name you specify when you invoke the compiler determines which stanza of the configuration file the compiler uses. You can add other stanzas to your copy of the configuration file to customize your own compilation environment.

Only one stanza, in addition to the one referenced by the "use" attribute, is processed for any one invocation of the xlc utility. By default, the stanza that matches the command name used to invoke the xlc utility is used, but it can be overridden using the `-F` flag option as described in the example below.

**Example:** You can use the `-F` option with the compiler invocation command to make links to select additional stanzas or to specify a stanza or another configuration file:

```
xlc myfile.C -Fmyconfig:SPECIAL
```

would compile myfile.C using the SPECIAL stanza in a myconfig configuration file that you had created.

**Default configuration file**
The default configuration file, (/usr/lpp/cbclib/xlc/etc/xlc.cfg), specifies information that the compiler uses when you invoke it. This file defines values used by the compiler to compile C or C++ programs. You can make entries to this file to support specific compilation requirements or to support other C or C++ compilation environments. Options specified in the configuration file override the default settings of the
option. Similarly, options specified in the configuration file are in turn overridden by options set in the source file and on the command line. Options that do not follow this scheme are listed in “Specifying compiler options” on page 866.

Example: The following example shows a default configuration file:

```plaintext
* *
* FUNCTION: z/OS 2.1.1 XL C/C++ Compiler Configuration file
*
* Licensed Materials - Property of IBM
* 5650-ZOS Copyright IBM Corp. 2004, 2014
* US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
* *
* C compiler, extended mode
  xlc: use = DEFLT
  * XPLINK C compiler, extended mode
    xlc_x: use = DEFLT
  * 64 bit C compiler, extended mode
    xlc_64: use = DEFLT
*
* C compiler, common usage C
  cc: use = DEFLT
  * XPLINK C compiler, common usage C
    cc_x: use = DEFLT
  * 64 bit C compiler, common usage C
    cc_64: use = DEFLT
*
* Strict ANSI C 89 compiler
  c89: use = DEFLT
  * XPLINK Strict ANSI C 89 compiler
    c89_x: use = DEFLT
  * 64 bit Strict ANSI C 89 compiler
    c89_64: use = DEFLT
*
* ISO/IEC 9899:1999 Standard Compliant C Compiler
  c99: use = DEFLT
  * XPLINK ISO/IEC 9899:1999 Standard Compliant C Compiler
    c99_x: use = DEFLT
  * 64 bit ISO/IEC 9899:1999 Standard Compliant C Compiler
    c99_64: use = DEFLT
*
* ANSI C++ compiler
  cxx: use = DEFLT
    xlc = /usr/lpp/cbclib/xlc/bin/.orig/xlc
    ipa = /bin/cxx
  * XPLINK ANSI C++ compiler
    cxx_x: use = DEFLT
    xlc = /usr/lpp/cbclib/xlc/bin/.orig/xlc
    ipa = /bin/cxx
  * 64 bit ANSI C++ compiler
    cxx_64: use = DEFLT
    xlc = /usr/lpp/cbclib/xlc/bin/.orig/xlc
    ipa = /bin/cxx
*
* ANSI C++ compiler
  c++: use = DEFLT
    xlc = /usr/lpp/cbclib/xlc/bin/.orig/xlc
    ipa = /bin/cxx
  * XPLINK ANSI C++ compiler
    c++_x: use = DEFLT
    xlc = /usr/lpp/cbclib/xlc/bin/.orig/xlc
    ipa = /bin/cxx
  * 64 bit ANSI C++ compiler
    c++_64: use = DEFLT
    xlc = /usr/lpp/cbclib/xlc/bin/.orig/xlc
    ipa = /bin/cxx
```
**Invoking the compiler**

The z/OS XL C/C++ compiler is invoked using the following syntax, where *invocation* can be replaced with any valid z/OS XL C/C++ invocation command:

```
invocation command_line_options input_files
```

The parameters of the compiler invocation command can be names of input files, compiler options, and linkage-editor options. Compiler options perform a wide variety of functions such as setting compiler characteristics, describing object code and compiler output to be produced, and performing some preprocessor functions.

To compile without binding, use the `-c` compiler option. The `-c` option stops the compiler after compilation is completed and produces as output, an object file `file_name.o` for each `file_name.c` input source file, unless the `-o` option was used to specify a different object filename. The binder is not invoked. You can bind the object files later using the invocation command, specifying the object files without the `-c` option.
Note:
1. Any object files produced from an earlier compilation with the same name as expected object files in this compilation are deleted as part of the compilation process, even if new object files are not produced.
2. By default, the invocation command calls both the compiler and the binder. It passes binder options to the binder. Consequently, the invocation commands also accept all binder options.

Invoking the binder

All invocation commands invoke the binder using the c89 utility, so all binder options must follow the syntax supported by the c89 utility. Standard libraries required to bind your program are controlled by the sysobj, syslib, and exportlist attributes in the configuration file.

The specified object files are processed by the binder to create one executable file. Invoking the compiler with one of the invocation commands, automatically calls the binder unless you specify one of the following compiler options: -E, -c, -P, -qsyntaxonly, -qpponly, or -#.

All input and output files supported by the c89 utility are valid for all invocation commands.

Supported options

In addition to -W syntax for specifying keyword options, the xlc utility supports AIX -q options syntax and several new flag options.

Syntax for the -q option

The following principles apply to the use of z/OS option names with -q syntax:

- Any valid abbreviation of a z/OS option name that matches (in full or in part) the spelling of the corresponding option on AIX, can be specified by using -q syntax. For example, ATTRIBUTE can be specified as -qatt, -qattr, -qattrib, -qattribu, -qattribute, and -qattrib. This is true even if the AIX option name is longer, as in the case of -qbitfields, which can be specified as -qbitf, -qbitfi, -qbitfile, -qbitfield, and -qbitfields. This is the common case that applies to most z/OS options except any suboptions.

- Any z/OS-specific option name and its valid abbreviation can also be specified using -q syntax; for example, DBRMLIB.

- Any z/OS option name that has a different spelling from the corresponding AIX option name cannot be specified using -q syntax. For example, CHECKOUT, EXH, ILP32, LP64, SSCOMM, and TEST cannot be specified using -q syntax. Instead, use, -qinfo, -qeh, -q32, -q64, -qcpluscmt, and -qdebug=format=isd. For historical reasons, OBJECTMODEL and PHASEID are exceptions to this principle, as both can be specified with the -q syntax. However, -qobjmodel and -qphsinfo should be used instead to enhance portability with AIX.

Options that do not exist on AIX, and are not required to accomplish a z/OS-specific task, and their effect can be accomplished by other means, are not supported by -q syntax. For example, use -D instead of DEFINE, -U instead of UNDEFINE, and -co instead of OBJECT.

Suboptions with negative forms of -q options are not supported, unless they cause an active compiler action, as in the case of -qnokeyword=<keyword>.

Compiler options for AIX that do not apply to z/OS are accepted and ignored with a diagnostic message. For a brief description of the compiler options that can be specified with xlc, type xlc or any other supported command name. For detailed descriptions of the compiler options that can be specified with xlc, refer to z/OS XL C/C++ User's Guide.

The following syntax diagram shows how to specify keyword options using -q syntax:
In the diagram, **option_keyword** is an option name and the optional **suboption** is a value that is associated with the option. Keyword options with no suboptions represent switches that can be either on or off. The **option_keyword** by itself turns on the switch, and the **option_keyword** preceded by the letters NO turns the switch off. For example, `-qLIST` tells the compiler to produce a listing and `-qNOLIST` tells the compiler not to produce a listing. If an option that represents a switch is set more than once, the compiler uses the last setting.

Some keyword options only have values. Keywords that have values are specified as keyword=value pairs. In `-qfloat=ieee`, for instance, `ieee` is a value.

Some keyword options have suboptions, which in turn have values. Suboptions that have values are specified as suboption=value pairs. In `-qipa=level=2`, for instance, `level` is a suboption and `2` is a value.

Keyword options and suboptions can appear in mixed case letters in the command that invokes the **xlc** utility. Keyword options that have suboptions can also be preceded by the letters NO in which case they are similar to off switches and do not allow suboptions. This is a noticeable departure from the z/OS options, which allow suboptions even if they are preceded by the letters NO. However, the function that the z/OS behavior provides can easily be emulated by specifying all of the suboptions with an **option_keyword** followed by the same **option_keyword** that is preceded by the letters NO. The subsequent specification of the same **option_keyword** unlocks all previously specified suboptions.

For example, `NODEBUG(FORMAT(DWARF))` is equivalent to `-qdebug=format=dwarf -qnodebug`.

The compiler recognizes all AIX `-q` options, but only those that have a matching z/OS native option are accepted and processed. All other AIX `-q` options are ignored with an informational message.

**Note:** The GENASM compiler option is not supported with `-q` syntax. Use the `-S` flag option instead, which is described in “Flag options syntax” on page 862.

**Flag options syntax**

Except for the `-W`, `-D`, and `-U` flag options, all flag options that are supported by the `c89` utility are supported by the `xlc` utility with the same semantics. The `xlc` utility does not recognize constructs such as `-Wl,I` or `-Wl,p`. All other aspects of the `-W` flag are the same as with the `c89` utility. `-D` and `-U` flag options are not preprocessed by the `xlc` utility. Instead, they are converted to the DEFINE and UNDEFINE native options and are passed to the compiler. The `xlc` utility also supports several additional flag options, which are described below:

-`-#`
  - Displays language processing commands but does not invoke them; output goes to stdout.

-`-B`
  - Determines substitute path names for programs such as the assembler and binder, where program can be:
    - a (assembler)
    - c (z/OS XL C/C++ compiler)
    - l (binder)
    - L (IPA Link)
Note:

1. The optional prefix defines part of a path name to the new programs. The compiler does not add a / between the prefix and the program name.

2. To form the complete path name for each program, the xlc utility adds prefix to the program names indicated by the -t option. The program names can be any combination of z/OS XL C/C++ compiler, assembler, IPA Link and binder.

3. If -Bprefix is not specified, or if -B is specified without the prefix, the default path (/usr/lpp/cbclib/xlc/bin/) is used.

4. -tprograms specifies the programs for which the path name indicated by the -B option is to be applied.

5. -Bprefix and -tprograms options override the path names of the programs that are specified inside the configuration file indicated by the -Fconfig_file option.

Example: To compile myprogram.c using a substitute compiler and binder from /lib/tmp/mine/, enter:

```
xlc myprogram.c -B/lib/tmp/mine/
```

Example: To compile myprogram.c using a substitute binder from /lib/tmp/mine/, enter:

```
xlc myprogram.c -B/lib/tmp/mine/ -tl
```

-F

Names an alternative configuration file (.cfg) for the xlc utility.

Suboptions are:

- config_file (specifies the name of an xlc configuration file.)
- stanza (specifies the name of the command used to invoke the compiler. This directs the compiler to use the entries under stanza in the config_file to set up the compiler environment.)

```
-M
```

Instructs the compiler to generate a dependency file or dependency files that can be used by the make utility. By default, the dependency file name and path is the same as the -o compile option but with .u suffix. Dependency file name can be overridden by the -MF option. The default suffix can be modified through "usuffix" configuration file attribute.

The compiler will generate as many dependency files as there are source files specified.

- M is the equivalent of specifying -qmakedep.
**Example:** To compile myprogram.c and create an output file named myprogram.u, enter:

```
xlc -c -M myprogram.c
```

**Example:** The following is a simple makefile that uses -M feature. You can refer to [z/OS UNIX System Services Programming Tools](https://www.ibm.com/support/knowledgecenter/SSEU52_2.2.0/com.ibm.zos.v2r2.doc/mac000000m.html) for more information about the make utility and makefiles.

```plaintext
CFLAGS = -M -qLSE=lib1 -qfloat=ieee
all: program
# Compile recipe; will also regenerate dependencies, used on the next compile
%.o: %.c
 xlc -c $(CFLAGS) $^;
program: file1.o file2.o
 xlc $(CFLAGS) -o program $&
#This line will include the generated file1.u and file2.u only
if they exist .INCLUDE .IGNORE : file1.u file2.u
```

- **MF**

  If **-M** or **-qmakedep** is specified, this option is used to override the file path name of the dependency file.

- **Note:**

  1. The argument of **file** can not be the name of a dataset.
  2. If the file specified by **-MF** already exists, it will be overwritten. Moreover, if the output path specified does not exist or is write-protected, an error message will be issued.
  3. If you specify a single file name for the **-MF** option when compiling multiple source files, each generated dependency file overwrites the previous one. Only a single output file will be generated for the last source file specified on the command line.

**Example:** You can refer to the following table for detail usage of **-M** and **-MF**:

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Dependency File</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-MF</strong> is not specified</td>
<td><code>xlc -c -M t.c</code></td>
<td>./t.u is generated</td>
</tr>
<tr>
<td></td>
<td><code>xlc -M -c -o obj.o t.c</code></td>
<td>./obj.u is generated</td>
</tr>
<tr>
<td></td>
<td><code>xlc -c -M -o dir/ t.c</code></td>
<td>./dir/t.u is generated if ./dir is writable</td>
</tr>
<tr>
<td><strong>-MF</strong> specifies a file</td>
<td><code>xlc -c -qmakedep -MF dep.u t.c</code></td>
<td>./dep.u is generated</td>
</tr>
<tr>
<td></td>
<td><code>xlc -c -o obj.o -M ..../dep.x t.c</code></td>
<td>../dep.x is generated</td>
</tr>
<tr>
<td></td>
<td><code>xlc -c -M -MF dir/dep.d a.c b.c</code></td>
<td>../dir/dep.d is generated for b.c only.</td>
</tr>
</tbody>
</table>
Table 36: Example of using -M and -MF. (continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Dependency File</th>
</tr>
</thead>
<tbody>
<tr>
<td>-MF specifies a directory</td>
<td><code>xlc -c -M -MF dir/ a.c b.c</code></td>
<td>./dir/a.u and ./dir/b.u are generated for a.c and b.c respectively if ./dir/ is writable</td>
</tr>
</tbody>
</table>

-O
Optimizes generated code.

-O2
Same as -O.

-O3
Performs memory and compile-time intensive optimizations in addition to those executed with -O2. The -O3 specific optimizations have the potential to alter the semantics of a user's program. The compiler guards against these optimizations at -O2 and the option -qstrict is provided at -O3 to turn off these aggressive optimizations.

-O4
Equivalent to -O3 -qipa and -qhot.

-O5
Equivalent to -O3 -qipa=level=2 and -qhot.

-P
Produces preprocessed output in a file that has a suffix that is defined by isuffix, isuffix_host, ixxsuffix, and ixxsuffix_host. The default for host files is .CEX and for z/OS UNIX files is .i. As with the -E option, the -C option can be combined with the -P option to preserve the comments.

-S
Produces an assembler source file for C source that is compiled with the METAL compiler option. The -o option can be used to override the default file name produced by -S. The default file name is the C source file name with the suffix determined by the ssuffix and ssuffix_host attributes in the configuration file.

When you specify the -o option, the assembler source file name is based on the name specified with the option. For example, when you specify `xlc -S -qmetal -c -o foo.x hello.c`, the output assembler source file name is foo.x. The following specifications have the same result:

- xlC -S -qmetal hello.c
- xlC -S -qmetal -o hello.s hello.c
- xlC -S -qmetal -c hello.c
- xlC -S -qmetal -c -o hello.s hello.c

-t
Adds the prefix specified by the -B option to the designated programs, where programs are:
- a (assembler)
- c (z/OS XL C/C++ compiler)
- L (Interprocedural Analysis tool - link phase)
• 1 (binder)

**Note:** This option must be used together with the -B option.

If -B is specified but the prefix is not, the default prefix is /usr/lpp/cbclib/xlc/bin/. If **-B prefix** is not specified at all, the prefix of the standard program names is /usr/lib/cbclib/xlc/bin/.

If -B is specified but **-t programs** is not, the default is to construct path names for all of the standard program names: a, c, L, and l.

**Example:** To compile myprogram.c so that the name /u/new/compilers/ is prefixed to the binder and assembler program names, enter:

```
xlc myprogram.c -B/u/new/compilers/ -tla
```

**-W**

Passes the listed options to a designated compiler program where programs are:

- a (assembler)
- c (z/OS XL C/C++ compiler)
- I (Interprocedural Analysis tool - compile phase)
- l (binder)

**Note:** When used in the configuration file, the -W option requires the escape sequence back slash comma (\,) to represent a comma in the parameter string.

**Example:** To compile myprogram.s so that the option map is passed to the binder and the option list is passed to the assembler, enter:

```
xlc myprogram.s -Wl,map -Wa,list
```

**Example:** In a configuration file, use the \, sequence to represent the comma (,):

```
-Wl\,map,-Wa\,list
```

**Specifying compiler options**

Compiler options perform a wide variety of functions, such as setting compiler characteristics, describing the object code and compiler output to be produced, and performing some preprocessor functions. You can specify compiler options in one or more of the following ways:

- On the command line
- In your source program
• In a configuration file

The compiler uses default settings for the compiler options not explicitly set by you in these listed ways. The defaults can be compiler defaults, installation defaults, or the defaults set by the c89 utility or the xlc utility. The compiler defaults are overridden by installation defaults, which are overridden by the defaults set by the c89 utility or the xlc utility.

When specifying compiler options, it is possible for option conflicts and incompatibilities to occur. z/OS XL C/C++ resolves most of these conflicts and incompatibilities in a consistent fashion, as follows:

<table>
<thead>
<tr>
<th>Source file overrides</th>
<th>Command line overrides</th>
<th>Configuration file overrides</th>
<th>Default settings</th>
</tr>
</thead>
</table>

Options that do not follow this scheme are summarized in the following table:

<table>
<thead>
<tr>
<th>Option</th>
<th>Conflicting Options</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>-qxref</td>
<td>-qxref=FULL</td>
<td>-qxref=FULL</td>
</tr>
<tr>
<td>-qattr</td>
<td>-qattr=FULL</td>
<td>-qattr=FULL</td>
</tr>
<tr>
<td>-E</td>
<td>-o</td>
<td>-E</td>
</tr>
<tr>
<td>-#</td>
<td>-v</td>
<td>-#</td>
</tr>
<tr>
<td>-F</td>
<td>-B</td>
<td>-t</td>
</tr>
<tr>
<td>-qpath</td>
<td>-B</td>
<td>-t</td>
</tr>
</tbody>
</table>

In general, if more than one variation of the same option is specified (with the exception of xref and attr), the compiler uses the setting of the last one specified. Compiler options specified on the command line must appear in the order you want the compiler to process them.

If a command-line flag is valid for more than one compiler program (for example -B, -W, or -I applied to the compiler, binder, and assembler program names), you must specify it in options, or asopt in the configuration file. The command-line flags must appear in the order that they are to be directed to the appropriate compiler program.

Three exceptions to the rules of conflicting options are the -I directory or -I // dataset_name, -1 library, and -L directory options, which have cumulative effects when they are specified more than once.

**Specifying compiler options on the command line**

There are two kinds of command-line options:

- **-q option_keyword** (compiler-specific)
- Flag options (available to z/OS XL C/C++ compilers in z/OS UNIX System Service environment)

Command-line options in the **-q option_keyword** format are similar to on and off switches. For most -q options, if a given option is specified more than once, the last appearance of that option on the command line is the one recognized by the compiler. For example, qsource turns on the source option to produce a compiler listing, and -qnosource turns off the source option so that no source listing is produced.

The following example would produce a source listing for both MyNewProg.C and MyFirstProg.C because the last source option specified (-qsource) takes precedence:

```
xlC -qsource MyFirstProg.C -qsource MyNewProg.C
```

You can have multiple **-q option_keyword** instances in the same command line, but they must be separated by blanks. Option keywords can appear in mixed case, but you must specify the **-q** in lowercase.
You can specify any `-q option_keyword` before or after the file name:

```
xlc -qLIST -qnomaf file.c
xlc file.c -qxref -qsource
```

Some options have suboptions. You specify these with an equal sign following the `-qoption`. If the option permits more than one suboption, a colon (:) must separate each suboption from the next.

The following example compiles the C source file `file.c` using the option `-qipa` to specify the interprocedural analysis options. The suboption `level=2` tells the compiler to use the full interprocedural data flow and alias analysis, `map` tells the compiler to produce a report, and the `noobj` tells the compiler to produce only an IPA object without a regular object. The option `-qattr` with suboption `full` will produce an attribute listing of all identifiers in the program.

```
xlc -qipa=level=2:map:noobj -qattr=full file.c
```

**Specifying flag options**

The z/OS XL C/C++ compilers use a number of common conventional flag options. Lowercase flags are different from their corresponding uppercase flags. For example, `-c` and `-C` are two different compiler options:

- `-c` specifies that the compiler should only preprocess, compile, and not invoke the binder
- `-C` can be used with `-E` or `-P` to specify that user comments should be preserved

Some flag options have arguments that form part of the flag. Here is an example: `xlc stem.c -F/home/tools/test3/new.cfg:myc -qflag=w` where `new.cfg` is a custom configuration file.

You can specify flags that do not take arguments in one string; for instance, `xlc -Ocv file.c` has the same effect as `xlc -O -v -c test.c`.

**Specifying compiler options in a configuration file**

The default configuration file, `(/usr/lpp/cbclib/xlc/etc/xlc.cfg)`, specifies information that the compiler uses when you invoke it. This file defines values used by the compiler to compile C or C++ programs. You can make entries to this file to support specific compilation requirements or to support other C or C++ compilation environments.

Options specified in the configuration file override the default settings of the option. Similarly, options specified in the configuration file are in turn overridden by options set in the source file and on the command line.

**Specifying compiler options in your program source files**

You can specify compiler options within your program source by using `#pragma` directives. Options specified with `pragma` directives in program source files override all other option settings.

**Specifying compiler options for architecture-specific 32-bit or 64-bit compilation**

You can use z/OS XL C/C++ compiler options to optimize compiler output for use on specific processor architectures. You can also instruct the compiler to compile in either 32-bit or 64-bit mode.

The compiler evaluates compiler options in the following order, with the last allowable one found determining the compiler mode:

1. Compiler default (32-bit mode)
2. Configuration file settings
3. Command line compiler options (`-q32`, `-q64`, `-qarch`, `-qtune`)
4. Source file statements (`#pragma options(ARCH(suboption),TUNE(suboption))`)

The compilation mode actually used by the compiler depends on a combination of the settings of the `-q32`, `-q64`, `-qarch`, and `-qtune` compiler options, subject to the following conditions:
• Compiler mode is set according to the last-found instance of the -q32, or -q64 compiler options. If neither of these compiler options is chosen, the compiler mode is set to 32-bit.

• Architecture target is set according to the last-found instance of the -qarch compiler option, provided that the specified -qarch setting is compatible with the compiler mode setting. If the -qarch option is not set, the compiler assumes a -qarch setting of 5.

• Tuning of the architecture target is set according to the last-found instance of the -qtune compiler option, provided that the -qtune setting is compatible with the architecture target and compiler mode settings. If the -qtune option is not set, the compiler assumes a default -qtune setting according to the -qarch setting in use.

Possible option conflicts and compiler resolution of these conflicts are described below:

• -q32 or -q64 setting is incompatible with user-selected -qarch option.
  Resolution: -q32 or -q64 setting overrides -qarch option; compiler issues a warning message, sets -qarch to 5, and sets the -qtune option to the -qarch setting's default -qtune value.

• -q32 or -q64 setting is incompatible with user-selected -qtune option.
  Resolution: -q32 or -q64 setting overrides -qtune option; compiler issues a warning message, and sets -qtune to the -qarch setting's default -qtune value.

• -qarch option is incompatible with user-selected -qtune option.
  Resolution: Compiler issues a warning message, and sets -qtune to the -qarch setting's default -qtune value.

• Selected -qarch and -qtune options are not known to the compiler.
  Resolution: Compiler issues a warning message, sets -qarch to 5, and sets -qtune to the -qarch setting's default -qtune setting. The compiler mode (32 or 64-bit) is determined by the -q32 or -q64 compiler settings.

---

**xlc - Compile C and C++ source code, link-edit and create an executable file**

See xlc.

When working in the shell, to view man page information about xlc, type man xlc.

**xlc++ - Compile C and C++ source code, link-edit and create an executable file**

See xlc.

When working in the shell, to view man page information about xlc++, type man xlc.

**xargs - Construct an argument list and run a command**

**Format**

```
xargs [-I placeholder] [-i [placeholder]] [-L number] [-l [number]] [-n number] [-ptx] [-E [eofstr]] [-e [eofstr]] [-s size] [command [argument ...]]
```

**Description**

The **xargs** command line typically contains the skeleton, or *template*, of another command. This template looks like a normal command, except that it lacks some arguments. **xargs** adds arguments from standard input (the standard input) to complete the command, then runs the resulting command. If more input remains, it repeats this process.
In a double-byte locale, some options may accept a double-byte string as an argument. In these cases, an incorrect double-byte string would be detected during command-line parsing.

The maximum length of a constructed command is LINE_MAX bytes.

Options

**xargs** gets the needed arguments from standard input (the standard input). Different options tell how the standard input is to be interpreted to obtain these arguments.

**-I placeholder**

Specifies that each line in the standard input (the standard input) is to be considered as a single argument. The *placeholder* following the -I is a string that can appear multiple times in the command template. **xargs** strips the input line of any leading white space characters and inserts it in place of the *placeholder* string. For example:

```bash
xargs -I '{}'
mv dir1/{} dir2/ '{}'
```

The standard input should consist of lines giving names of files that you want moved from *dir1* to *dir2*. **xargs** substitutes these names for the `{}` placeholder in each place that it appears in the command template.

When **xargs** creates arguments for the template command, no single argument can be longer than 255 characters after the input has replaced the placeholders. The -x option is automatically in effect if -I or -i is used. If you omit the *placeholder* string, it defaults to the string `{ }`. Thus we could write our preceding example as:

```bash
xargs -i mv dir1/ '{}' dir2/ '{}'
```

In a double-byte locale, *placeholder* may contain double-byte characters.

**-i placeholder**

Behaves like -I, except that the *placeholder* is optional. If you omit the *placeholder* string, it defaults to the string `{ }`. Thus, the previous example could be written as:

```bash
xargs -i mv dir1/ ' {' dir2/ ' '}
```

- **-L number**

Specifies that **xargs** read *number* lines from the standard input and concatenate them into one long string (with a blank separating each of the original lines). **xargs** then appends this string to the command template and runs the resulting command. This process is repeated until **xargs** reaches the end of the standard input if there are fewer than *number* lines left in the file the last time the command is run, **xargs** just uses what is there.

With this option, a line must contain at least one nonblank character; blank lines are skipped and do not count toward the number of lines being added to the template. **xargs** considers a line to end at the first newline character, unless the last character of the line is a blank or a tab; in this case, the current line is considered to extend to the end of the next non-empty line.

If you omit the -L or -l option, the default number of lines read from the standard input is 1. The -x option is automatically in effect if -l is used.

**-l number**

Acts like the -L option, but the *number* argument is optional. *number* defaults to 1.

**-n number**

Specifies **xargs** is to read the given number of arguments from the standard input and put them on the end of the command template. For example:

```bash
xargs -n
```

If you do not specify the -n option, the default number of arguments read from the standard input is 1.
Typically, an `xargs` command uses exactly one of the options just described. If you specify more than one, `xargs` uses the one that appears last on the command line. If the command has none of these options, `xargs` keeps reading input until it fills up its internal buffer, concatenating arguments to the end of the command template. When the buffer is full, `xargs` runs the resulting command, and then starts constructing a new command. For example:

```
ls | xargs echo
```

prints the names of files in the working directory as one long line. When you invoke `xargs` this way, the total length of all arguments must be less than the size specified by the `-s` option.

If no command template appears on the command line, `xargs` uses `echo` by default. When `xargs` runs a command, it uses your search rules to find the command; this means that you can run shell scripts as well as normal programs.

The command you want to execute should be in your search `$PATH`.

`xargs` ends prematurely if it cannot run a constructed command or if an executed command returns a nonzero status.

If an executed command is a shell program, it should explicitly contain an `exit` command to avoid returning a nonzero by accident; see `sh` for details.

You can use the following options with any of the three main options.

**-E [eofstr]**
Defines `eofstr` to represent end-of-file on the standard input. For example:

```
-E :::
```

```
tells xargs that ::: represents the end of the standard input, even if an input file continues afterward. If there is no -E or -e option, a single underscore (_) marks the end of the input.
```

```
In a double-byte locale, eofstr may contain double-byte characters.
```

**-e [eofstr]**
Acts like -E but the `eofstr` argument is optional. If you specify -e without `eofstr`, there is no end-of-file marker string, and _ is taken literally instead of as an end-of-file marker. `xargs` stops reading input when it reaches the specified end-of-file marker or the true end of the file.

**-p**
Prompts you before each command. This option turns on the -t option so that you see each constructed command before it is run. Then `xargs` displays ?…, asking if you really want to run this command. If you type a string beginning with y, `xargs` runs the command as displayed; otherwise, the command is not run, and `xargs` constructs a new command.

**-s size**
Sets the maximum allowable size of an argument list to `size` bytes (where `size` is an integer). The value of `size` must be less than or equal to the system variable `LINE_MAX`. If you omit the -s option, the default allowable size of an argument list is `LINE_MAX`. The length of the argument list is the length of the entire constructed command; this includes the length of the command name, the length of each argument, plus one blank for separating each item on the line.

**-t**
Writes each constructed command to stderr just before running the command.

**-x**
Kills `xargs` if it creates a command that is longer than the size given by the -s option (or `{LINE_MAX}` is -s was not specified). This option comes into effect automatically if you specify -i or -l.

**Examples**
The following displays file names in three columns:

```
ls | xargs -n 3 echo
```

```
prints the names of files in the working directory as one long line. When you invoke xargs this way, the total length of all arguments must be less than the size specified by the -s option.
```

```
If no command template appears on the command line, xargs uses echo by default. When xargs runs a command, it uses your search rules to find the command; this means that you can run shell scripts as well as normal programs.
```

```
The command you want to execute should be in your search $PATH.
```

```
xargs ends prematurely if it cannot run a constructed command or if an executed command returns a nonzero status.
```

```
If an executed command is a shell program, it should explicitly contain an exit command to avoid returning a nonzero by accident; see sh for details.
```

```
You can use the following options with any of the three main options.
```

```
-E [eofstr]
Defines eofstr to represent end-of-file on the standard input. For example:
```

```
-E :::
```

```
tells xargs that ::: represents the end of the standard input, even if an input file continues afterward. If there is no -E or -e option, a single underscore (_) marks the end of the input.
```

```
In a double-byte locale, eofstr may contain double-byte characters.
```

```
-e [eofstr]
Acts like -E but the `eofstr` argument is optional. If you specify -e without `eofstr`, there is no end-of-file marker string, and _ is taken literally instead of as an end-of-file marker. xargs stops reading input when it reaches the specified end-of-file marker or the true end of the file.
```

```
-p
Prompts you before each command. This option turns on the -t option so that you see each constructed command before it is run. Then xargs displays ?…, asking if you really want to run this command. If you type a string beginning with y, xargs runs the command as displayed; otherwise, the command is not run, and xargs constructs a new command.
```

```
-s size
Sets the maximum allowable size of an argument list to `size` bytes (where `size` is an integer). The value of `size` must be less than or equal to the system variable `LINE_MAX`. If you omit the -s option, the default allowable size of an argument list is `LINE_MAX`. The length of the argument list is the length of the entire constructed command; this includes the length of the command name, the length of each argument, plus one blank for separating each item on the line.
```

```
-t
Writes each constructed command to stderr just before running the command.
```

```
-x
Kills xargs if it creates a command that is longer than the size given by the -s option (or `{LINE_MAX}` is -s was not specified). This option comes into effect automatically if you specify -i or -l.
```

```
Examples
The following displays file names in three columns:
```

```
ls | xargs -n 3 echo
```

```
prints the names of files in the working directory as one long line. When you invoke xargs this way, the total length of all arguments must be less than the size specified by the -s option.
```

```
If no command template appears on the command line, xargs uses echo by default. When xargs runs a command, it uses your search rules to find the command; this means that you can run shell scripts as well as normal programs.
```

```
The command you want to execute should be in your search $PATH.
```

```
xargs ends prematurely if it cannot run a constructed command or if an executed command returns a nonzero status.
```

```
If an executed command is a shell program, it should explicitly contain an exit command to avoid returning a nonzero by accident; see sh for details.
```

```
You can use the following options with any of the three main options.
```

```
-E [eofstr]
Defines eofstr to represent end-of-file on the standard input. For example:
```

```
-E :::
```

```
tells xargs that ::: represents the end of the standard input, even if an input file continues afterward. If there is no -E or -e option, a single underscore (_) marks the end of the input.
```

```
In a double-byte locale, eofstr may contain double-byte characters.
```

```
-e [eofstr]
Acts like -E but the `eofstr` argument is optional. If you specify -e without `eofstr`, there is no end-of-file marker string, and _ is taken literally instead of as an end-of-file marker. xargs stops reading input when it reaches the specified end-of-file marker or the true end of the file.
```

```
-p
Prompts you before each command. This option turns on the -t option so that you see each constructed command before it is run. Then xargs displays ?…, asking if you really want to run this command. If you type a string beginning with y, xargs runs the command as displayed; otherwise, the command is not run, and xargs constructs a new command.
```

```
-s size
Sets the maximum allowable size of an argument list to `size` bytes (where `size` is an integer). The value of `size` must be less than or equal to the system variable `LINE_MAX`. If you omit the -s option, the default allowable size of an argument list is `LINE_MAX`. The length of the argument list is the length of the entire constructed command; this includes the length of the command name, the length of each argument, plus one blank for separating each item on the line.
```

```
-t
Writes each constructed command to stderr just before running the command.
```

```
-x
Kills xargs if it creates a command that is longer than the size given by the -s option (or `{LINE_MAX}` is -s was not specified). This option comes into effect automatically if you specify -i or -l.
```

```
Examples
The following displays file names in three columns:
```

```
ls | xargs -n 3 echo
```
Localization

**xargs** uses the following localization environment variables:

- LANG
- LC_ALL
- LC_CTYPE
- LC_MESSAGES
- LC_SYNTAX
- NLSPATH


Exit values

0
  Successful completion of all commands

1-125
  Failure due to any of the following:
  - **xargs** could not assemble a command line.
  - One or more invocations of **command** returned a nonzero exit status.
  - Some other error occurred.

126
  **xargs** found **command** but could not invoke it.

127
  **xargs** could not find **command**.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.
The -e, -E, -i, -l, -L, and -p options are extensions of the POSIX standard.

Related information

echo, find, sh

**yacc** - Use the yacc compiler

Format


Description

**yacc** converts a context-free LALR(1) grammar that is found in the input file **gram.y** into a set of tables that together with additional C code constitute a parser to recognize that grammar. If you specify an input file that is named -, **yacc** reads the grammar from the standard input. By default, **yacc** places the parsing tables and associated C code into the file y.tab.c.

You can find detailed information about writing parsers that use **yacc** in z/OS UNIX System Services Programming Tools.
Options

- `b file_prefix` Uses `file_prefix` instead of `y` as the prefix for all output file names. For example, `yacc` names the parsing table `file_prefix.tab.c` rather than `y.tab.h`.

- `D file.h` Generates the file `file.h`, which contains the constant definition statements for token names. This lets other modules of a multmodule program access these symbolic names. This is the same as `-d`, except that the user specifies the include file name.

- `d` Generates the file `y.tab.h`, which contains the constant definition statements for token names. This lets other modules of a multmodule program access these symbolic names. This is the same as `-D`, except that the user does not specify the header file name.

- `h` Displays a brief list of the options and quits.

- `l` Disables the generation of `#line` statements in the parser output file, which are used to produce correct line numbers in compiler error messages from `gram.y`.

- `m` Displays memory usage, timing, and table size statistics on the standard output.

- `o file.c` Places the generated parser tables into `file.c` instead of the default `y.tab.c`.

- `P yyparse.c` Indicates that the C parser template is found in the file `yyparse.c`. If you do not specify this option, this parser template is located in `/etc/yyparse.c`.

- `p prefix` By default, `yacc` prefixes all variables and defined parameters in the generated parser code with the two letters `yy` (or `YY`). In order to have more than one `yacc`-generated parser in a single program, each parser must have unique variable names. `-p` uses the string `prefix` to replace the `yy` prefix in variable names. `prefix` should be entirely in lowercase because `yacc` uses an uppercase version of the string to replace all `YY` variables. We recommend a short prefix (such as `zz`) because some C compilers have name length restrictions for identifiers. You can also set this identifier with a `%prefix` directive in the grammar file.

- `q` Disables the printing of warning messages.

- `t` Enables debugging code in the generated parser. `yacc` does not normally compile this code because it is under the control of the preprocessor symbol `YYDEBUG`.

  This option is therefore equivalent to either setting `YYDEBUG` on the C compiler command line or specifying `#define YYDEBUG` statement in the first section of the grammar.

- `V stats` Writes a verbose description of the parsing tables and any possible conflicts to the file `stats`.

  This is the same as `-v` except that the user specifies the file name.

- `v` writes a verbose description of the parsing tables and any possible conflicts to the file `y.output`.

Files

`yacc` uses the following files:

`/usr/lib/liby.a`
  yacc function library.
/usr/lib/libyxp.a
  yacc archive library with functions that are compiled with XPLINK. Includes two versions: 64-bit addressing mode and 31-bit addressing mode.

y.output
  Default statistics file when you specify -v.

y.tab.c
  Default file for the generated parser.

y.tab.h
  Default header file when you specify -d.

/etc/yyparse.c
  Default parser template.

Localization

yacc uses the following localization environment variables:

• LANG
• LC_ALL
• LC_CTYPE
• LC_MESSAGES
• LC_SYNTAX
• NLSPATH

For more information, see Appendix F, “Localization,” on page 965.

Usage notes

In a double-byte environment, yacc can use double-byte characters, although this practice is possibly nonportable.

1. Comments and rule names can contain double-byte characters.
2. double-byte characters can be used in symbolic token names (generated by %token statements only if the C preprocessor and compiler will interpret them correctly. Symbolic token names are converted directly into #define statements and are then interpreted by the preprocessor and the compiler.
3. You can use double-byte characters as literal token definitions (a double-byte character that is surrounded by apostrophes), although this will generate a warning and might create a conflict with an assigned token name.

Exit values

0
  Successful completion

1
  Failure due to any of the following reasons:
  • number rules never reduced
  • Reduce-reduce conflict
  • Shift-reduce conflict
  • NAME should have been defined earlier
  • \000 not permitted
  • EOF encountered while processing %union
  • EOF in string or character constant
  • EOF inside comment
  • Use of $number not permitted
• Nonterminal number, entry at number
• Action does not terminate
• Bad %start construction
• Bad syntax in %type
• Bad syntax on $<ident> clause
• Bad syntax on first rule
• Inability to find parser
• Inability to open input file
• Inability to open table file
• Inability to open temporary file
• Inability to open y.output
• Inability to place goto
• Inability to reopen action temporary file
• Default action causes potential type clash
• EOF before %3
• %prec syntax not permitted
• \nnn construction not permitted
• Comment not permitted
• Option not permitted
• Incorrect or missing ' or"
• Incorrect rule: missing semicolon, or |?
• Internal yacc error
• Incorrect escape, or incorrect reserved word
• Item too big
• More than number rules
• Must return a value, since LHS has a type
• Must specify type for name
• Must specify type of $number
• Newline in string.
• No space in action table
• Nonterminal symbol not permitted after %prec
• Nonterminal symbol never derives any token string
• Nonterminal symbol not defined
• Optimizer cannot open temporary file
• Out of space in optimizer
• Out of state space
• Redeclaration of precedence of symbol
• Redeclaration of type of symbol
• Syntax error
• Token incorrect on LHS of grammar rule
• Too many characters in ID's and literals
• Too many look-ahead sets
• Too many nonterminals
Messages

Possible error messages include:

**No input file**
You did not specify a grammar file `gram.y` on the command line.

**No parser produced**
Analysis of the input grammar shows that it contains inaccessible or ungrounded nonterminal symbols. Check the preceding report and revise the grammar.

**Out of memory at size bytes**
The specified grammar is too complex to process within the memory resources of the current configuration.

Limits

`yacc` dynamically allocates all internal tables so that grammar size and complexity are limited only by available memory.

Portability

POSIX.2, POSIX.2 C-Language Development Utilities Option, X/Open Portability Guide, UNIX systems. The `-D`, `-h`, `-m`, `-p`, `-q`, `-S`, `-s`, and `-v` options are extensions of the POSIX standard.

Related information

`lex`

*z/OS UNIX System Services Programming Tools*

**zcat - Uncompress and display data**

**Format**

`zcat -DVv [file ...]`

**Description**

`zcat` takes one or more compressed data files as input. The data files should be compressed with the `compress` command. If no data files are specified on the command line, `zcat` reads standard input (stdin). You can also pass stdin to `zcat` by specifying `-` as one of the files on the command line.

`zcat` uncompresses the data of all the input files, and writes the result on standard output (stdout). `zcat` concatenates the data in the same way `cat` does.

The names of compressed input files are expected to end in `.Z`. If a specified input file name does not end in this suffix, `zcat` automatically adds the `.Z`. For example, if the command line specifies file `abc`, `zcat` looks for `abc.Z`. 
**zcat** is equivalent to:

```
uncompress -c
```

**Options**

- **-D**
  Uncompresses files that were compressed using the dictionary option of **compress**.

- **-V**
  Prints the version number of **uncompress** that **zcat** calls.

- **-v**
  Prints the name of each file as it is uncompressed.

**Localization**

**zcat** uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_MESSAGES**
- **NLSPATH**


**Exit values**

0

Successful completion.

1

Failure due to any of the following reasons:

- Unknown command line option.
- File is not in compressed format.
- File was compressed with a number of bits **zcat** cannot handle.
- There is no space for decompress tables.
- The compressed file is corrupt.

**Portability**

UNIX systems

**Related information**

**cat, compress, uncompress**

---

**zlsof - Display information about open files, sockets, and pipes**

**Format**

```
```
Description
The zlsof utility displays information about open files, sockets, and pipes (including named pipes, which are also known as FIFO special files). The display includes the file name or inode number, associated PID, user, file system, and whether the file was locked by the byte range lock manager (BRLM).

Filters are provided to limit output. Filters include specific processes, ASIDs, jobs, users, files, file systems, sockets, and pipes.

The default output for an unauthorized invoker consists of open file information for processes that are associated with the user. If the invoker is authorized, the default output consists of open file information for all processes in the system.

You can also use the -rw option to monitor file system usage.

Operands
pathname|socket|pipe
If a pathname is given, only the usage of that file is listed. If pathname is a mount point, all files in use within that file system are listed.
If socket or pipe is entered, the output shows only the socket or pipe and FIFO summary usage.

Options
-a asid
ASID filter: an optional list of ASIDs that are separated by commas. The output shows the ASIDs. For example:

```
-a 23,1d
```

-c
Omits the current directory and root.

-d
Displays all deleted, but open files. This option also sets -i -s. To use this option, you must have UID=0 or be permitted to the BPX.SUPERUSER resource in the FACILITY class.

-i
Shows the inode numbers instead of file names.

-j job
Job filter: an optional list of job names that are separated by commas. The output shows the job names. For example:

```
-j resolver,cea
```

-json
Shows data in JSON format.

-l
Indicates files with byte range locks. To use this option, you must have UID=0 or be permitted to BPX.SUPERUSER resource in the FACILITY class.

-m maxatime
No longer derives the file names after maxtime seconds. The default is 60. Use this option with the -n option.

-n
Derives the real path names for files. This option is disabled after maxtime seconds (see the -m option).
-p pid
PID filter: an optional list of process IDs that are separated by commas. The output shows the process IDs. -p is the default output format. For example:

```
-p 11,22,37
```

-rw
Monitors matching file systems for read and write activity for a specified time. If a length of time is not specified, 20 seconds is used. To use this option, you must have UID=0 or be permitted to BPX.SUPERUSER resource in the FACILITY class.

The output consists of two parts. In the first part, the initial file or file system usage information that matches the input PIDs, ASIDs, jobs, users, and path name filters is displayed. In the second part, the summary read and write activity for the corresponding file systems is provided.

This option causes the command to be suspended for the specified time period while I/O counts are accumulated.

-su
Sets the effective UID to 0, if it is not already 0. To use this option, you must have UID=0 or be permitted to BPX.SUPERUSER resource in the FACILITY class.

-t
Shows the tally of file types open by process.

-u user
User filter: a list of user IDs that are separated by commas. For example:

```
-u wjs,jhc
```

-v
Verbose mode. Prints running status and information.

-x
Shows extended process information. The information includes UID with the user name, start time, elapsed time, CPU time, ppid, thread number, controlling TTY information, state of the process, and read/write open mode.

Examples
1. The following display shows output with a PID filter applied:

```
SY1:/> zlsof -p 1,2,3,16777220
zlsof 120515: Searching for all file usage by process 1,2,3,16777220
Command     PID User     File System    Mountpoint  Inode/file
BPXPINPR     1 IBMUSER  ZOS21.ROOT.HFS /           r 3
             ZOS21.ROOT.HFS /           c 3
EZBREINI     2 RESOLVER ZOS21.ROOT.HFS /           r 3
             ZOS21.ROOT.HFS /           c 3
EZBREUPS     3 RESOLVER ZOS21.ROOT.HFS /           r 3
             ZOS21.ROOT.HFS /           c 3
CEAPSRVR 16777220 CEA ZOS21.ROOT.HFS /           r 3
             ZOS21.ROOT.HFS /           c 3
CEAPSRVR 16777220 CEA socket: 1
End of output
```

2. The following display shows output with a user filter applied.
SY1:/> zlsof -u wellie1
zlsof 120515: Searching for all file usage by user wellie1
Command       PID User    File System    Mountpoint  Inode/file
obrowse  83886099 WELLIE1 ZOS21.ROOT.HFS /           r 3
          ZOS21.ROOT.HFS / c 3
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / r 3
          ZOS21.ROOT.HFS / c 3
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / 1301 //tmp/.sh_history
-sh    83886091 WELLIE1 ZOS21.ROOT.HFS /           r 3
          ZOS21.ROOT.HFS / c 3
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / 1259 /dev/ttyp0001
          ZOS21.ROOT.HFS / 1301 /tmp/.sh_history
OMVS   83886095 WELLIE1 ZOS21.ROOT.HFS /           r 3
          ZOS21.ROOT.HFS / 21 tmp
          ZOS21.ROOT.HFS / 1258 /dev/ptyp0001
          ZOS21.ROOT.HFS / 1323 //instruct.txt

3. The following display lists deleted, but open, files. The -d option is used.

SY1:/> zlsof -d
zlsof 120515: Searching for all file usage for deleted files
Command       PID User    File System    Mountpoint Size Inode/file
OMVS   83886095 WELLIE1 ZOS21.ROOT.HFS /           1 1323 //instruct.txt
End of output

4. When you issue the -x option, you will see output that is similar to the following one:

SY1:/> zlsof -u WELLIE1 -x
zlsof version=180323
Searching for all file usage by user WELLIE1
Command       PID User        File System    Mountpoint  Inode/file/process info
OMVS   67108885 WELLIE1(11)                            GMT Start: 04/06 17:46 ET: 00:04:11 CPU:
          Parent pid: 1 Threads: 4 TTY: State: MULPROCESS
          Command: OMVS
          ZOS22.ROOT.ZFS / r 1
          ZOS22.ROOT.ZFS / 16 tmp
          ZOS22.ROOT.ZFS / rd 1274 //mytmp/output
          ZOS22.ROOT.ZFS / rw 1280 /dev/ptyp0002
-sh    83886103 WELLIE1(11) GMT Start: 04/06 17:46 ET: 00:04:04 CPU:
          Parent pid: 67108885 Threads: 1 TTY: /dev/ttyp0002 State:
          Command: -sh
          ZOS22.ROOT.ZFS / r 1
          ZOS22.ROOT.ZFS / 1260 mytmp
          ZOS22.ROOT.ZFS / rw 1281 /dev/ptyp0002
          ZOS22.ROOT.ZFS / rw 1281 /dev/ptyp0002
          ZOS22.ROOT.ZFS / rw 1281 /dev/ptyp0002
          ZOS22.ROOT.ZFS / 1278 /tmp/.sh_history
OMVS   83886104 WELLIE1(11) GMT Start: 04/06 17:48 ET: 00:02:38 CPU:
          Parent pid: 83886103 Threads: 1 TTY: /dev/ttyp0002 State:
          Command: obrowse output
          ZOS22.ROOT.ZFS / r 1
          ZOS22.ROOT.ZFS / 1268 mytmp
          ZOS22.ROOT.ZFS / rw 1281 /dev/ptyp0002
          ZOS22.ROOT.ZFS / rw 1281 /dev/ptyp0002
          ZOS22.ROOT.ZFS / rw 1281 /dev/ptyp0002
zlsof End of output

Usage notes

1. In the output, the inode or file can be prefixed by one of the following letters:
   c
   Indicates the current directory for the process.
   r
   Indicates the current root for the process.
This process is waiting on a byte range lock for the file.

This process is holding a byte range lock for the file.

2. If the -c option is used, lines with c and r are suppressed.
3. If the -l option is not used, w and h are not displayed.
4. The -p, -j and -a options are mutually exclusive.
5. If neither -n nor -i are specified, the output format will contain any portion of the file name that was retained by the system in addition to the inode number.
6. Use -- to separate the last option flag from the operand if the operand starts with - or the last option flag takes an optional parameter that is not specified. For example, zlsof -j pipe will filter on job name PIPE but zlsof -j -- pipe will filter on pipe and show job names in the output.

As a special notation, most z/OS shell commands let you specify a double minus sign (--) to separate the options from the non-option arguments; -- means that there are no more options. Thus, if you have a directory named –F, you could enter ls -- -F to list the contents of that directory or the file attributes.

7. zlsof is available as a TSO/E command and a shell command. It can also be executed as a system REXX exec by using the F AXR,ZLSOF system command.
8. The -x option displays the command and its parameters. The parameters can be nonprintable characters.

**Exit values**

0  
The command was successfully processed.

1  
A failure occurred.

**Related information**

fuser
zlsof
Chapter 3. TSO/E commands

This part describes the Time Sharing Option Extensions (TSO/E) OMVS command that you use to invoke the shell and the TSO/E commands that you can use to work with the z/OS UNIX file system.

The OMVS command invokes the z/OS shell. You can enter the OMVS command from TSO/E or from the ISPF command processor panel.

You can use the man command to view descriptions of TSO/E commands. To do this, you must prefix all commands with `tso`. For example, to view a description of the MOUNT command, enter:

```bash
man tsomount
```

The commands for working with the file system are:

- BPXBATCH
- BPXMTEXT
- BPXTRACE
- BPXWMIGF
- ISHELL
- MKDIR
- MKNOD
- MOUNT
- OBROWSE
- OCOPY
- OEDIT
- OGET
- OGETX
- OPUT
- OPUTX
- OSHELL
- OSTEPLIB
- UNMOUNT
- ZLSOF

You can enter these TSO/E commands from:

- TSO/E
- The Interactive System Productivity Facility (ISPF) command processor panel (typically, option 6 on the ISPF menu).

Option 6 is usually preferable, because it does not convert into uppercase the commands that you enter. You should enter a TSO/E command from an ISPF panel that does not convert all the parameters into uppercase; some panels, such as the main ISPF panel, convert what you enter into uppercase. z/OS UNIX System Services is case-sensitive.

- The shell

**Note:**

1. The relative path name is relative to the working directory (usually the HOME directory) of the TSO/E session, not the shell session.
2. Use absolute path names when entering any TSO/E commands.
3. Avoid using spaces or single quotation marks within path names.

BPXBATCH - Run shell commands, shell scripts, or executable files

**Format**

```plaintext
BPXBATCH SH [program_name] | PGM program_name
```

where `program_name` is an optional parameter that is passed to SH or a required parameter when PGM is used.

**Description**

BPXBATCH makes it easy for you to run, from your TSO/E session, shell scripts or z/OS XL C/C++ executable files that reside in z/OS UNIX files. To use it when running shell scripts and executable files that reside in files through job control language (JCL), see BPXBATCH.

With BPXBATCH, you can allocate stdin only as z/OS UNIX files for passing input. You can allocate stdout, stderr or stdenv as MVS data sets or z/OS UNIX text files. The stdenv file for containing environment variables or the stderr and stdout files for saving job output can be allocated as SYSOUT, PDSE, PDS or sequential data sets. If you do not allocate them, stdin, stdout, stderr, and stdenv default to `/dev/null`. Allocate the standard files using the data definition PATH keyword options, or standard data definition options for MVS data sets, for stdenv, stdout and stderr.

**Note:** The stream of data in the file associated with the STDIN DD is supplied as input to either the target shell program (for the SH case) or target z/OS UNIX program (PGM case) via File Descriptor 0, which is the traditional UNIX standard input (stdin). This data is then read and handled by the shell or the designated program. BPXBATCH itself does nothing with the data in the file.

To display BPXBATCH syntax using multiple commands, use one of the following:

```plaintext
BPXBATCH SH
PXBATCH SH [program_name]
PXBATCH PGM program_name
```

In addition to using BPXBATCH, a user who wants to perform a local spawn without being concerned about environment setup (that is, without having to set specific environment variables which could be overwritten if they are also set in the user's profile) can use BPXBATSL. It provides users with an alternate entry point into BPXBATCH, and forces a program to run using a local spawn instead of fork/exec as BPXBATCH does. This ultimately allows a program to run faster.

The following example contains DD statements that are accessible to a program that was given control from BPXBATSL:

```plaintext
//jobname JOB ...
//stepname EXEC PGM=BPXBATSL,PARM='PGM program_name'
/* The following 2 DDs are still available in the program which gets */
/* control from BPXBATSL.
//DD1   DD DSN=MVSDSN.FOR.APPL1,DISP=SHR
//DD2   DD DSN=MVSDSN.FOR.APPL2,DISP=SHR
/* The following DDs are processed by BPXBATSL to create file descriptors */
/* for stdin, stdout, stderr
//STDIN  DD PATH='./stdin-file-pathname',PATHOPTS=(ORDONLY)
//STDOUT DD PATH='./stdout-file-pathname',PATHOPTS=(OWRONLY,OCREAT,OTRUNC
//PATHMODE=SIRWXU
//STDERR DD PATH='./stderr-file-pathname',PATHOPTS=(OWRONLY,OCREAT,OTRUNC
PATHMODE=SIRWXU
```

BPXBATSL is also useful when the user wants to perform a local spawn of their program, but also needs subsequent child processes to be fork/exec'ed. Formerly, with BPXBATCH, this could not be done because BPXBATCH and the requested program shared the same environment variables.
BPXBATSL is an alias of BPXBATCH.

BPXBATA2 and BPXBATA8 are provided as APF-authorized alternatives to BPXBATSL. They provide the capability for a target APF-authorized z/OS UNIX program to run in the same address space as the originating job, allowing it to share the same allocations and job log, and so on. BPXBATA2 is specifically intended to provide the capability for a PSW Key 2 APF-authorized z/OS UNIX program to be started. To ensure that the target program receives control PSW Key 2, a PPT entry for BPXBATA2 must be set up that specifies that BPXBATA2 starts up PSW Key 2. The same restrictions that apply to BPXBATSL also apply to BPXBATA2 and BPXBATA8, in addition to the following:

- The PGM keyword is the only invocation type that is supported. The SH keyword is not supported.
- The interfaces can only be used from started task address spaces.
- The z/OS UNIX program that is the target of the BPXBATA2 and BPXBATA8 job must be marked as an APF-authorized executable file.

Any other usage of the BPXBATA8 and BPXBATA2 interfaces than what is described is not supported and will cause the invoking job to fail.

**Parameters**

**SH|PGM**

Specifies whether BPXBATCH is to run a shell script or command, or a z/OS C/C++ executable file located in a z/OS UNIX file.

If neither SH nor PGM is specified, BPXBATCH assumes that the shell is to be started in order to run the shell script allocated by stdin.

**SH**

Instructs BPXBATCH to start the shell and to run shell commands or scripts provided from stdin or the specified *program_name*. BPXBATCH passes all of the argument data, blanks included as is, to the shell as one parameter.

**Example:**

BPXBATCH PARM='SH command string'

If you specify SH with no *program_name* information, BPXBATCH attempts to run anything read in from stdin.

SH is the default.

**PGM**

Instructs BPXBATCH to run the specified *program_name* as a called program. This is done either via a spawn or a fork and exec. BPXBATCH creates a process for the program to run in and then calls the program. BPXBATCH breaks each argument up that is separated by one or more blanks into multiple parameters passed to the target program, taking out the blanks.

**Example:**

BPXBATCH PARM='PGM arg1 ... argn'

**Rule:** If you specify PGM, you must also specify *program_name*.

All environment variables read from stdenv are set when the program is run, if stdenv was allocated. If the HOME and LOGNAME environment variables are not specified in the stdenv file, or stdenv was not allocated, then HOME and LOGNAME, if possible, are set when the program is run.

**Restriction:** When using PGM, the *program_name* parameter cannot contain any shell-specific functions because they will not be resolved. If shell specific functions must be specified, then SH should be used to avoid possible errors or unpredictable results.

**program_name**

Specifies the shell command name or path name for the shell script or z/OS XL C/C++ executable file that you want to run. *program_name* can also contain option information. *program_name* must be in uppercase and lowercase letters.
When PGM and program_name are specified and the specified program name does not begin with a slash character (/), BPXBATCH prefixes the user's initial working directory information to the program path name.

Arguments that may be passed to the program specified by program_name are determined by the program being passed to PGM as a parameter. For more information about arguments that may be passed to the program, refer to the documentation of the program.

Examples

1. You want to run the shell script you specify with stdin.
   
   ```sh
   ALLOCATE FILE(STDIN) PATH('/stdin_file_pathname')
   PATHOPTS(ORDONLY)
   BPXBATCH SH
   ```

2. You want to run the program /usr/bin/payroll.
   
   ```sh
   BPXBATCH PGM /usr/bin/payroll
   ```

3. You want to run the script shellscriptA and put its output into the file a.out in a temporary directory.
   
   ```sh
   BPXBATCH SH /u/usr/joe/shellscriptA > /tmp/a.out
   ```

**BPXMTEXT - Display reason code text**

See “bpxmaxtext - Display reason code text” on page 67.

**BPXTRACE - Activate or deactivate traces for processes**

See “bpxmaxtrace - Activate or deactivate traces for processes” on page 68.

**BPXWMIGF - Migrate HFS file systems to zFS**

See “bpxmaxmigf - Migrate HFS file systems to zFS” on page 74.

**ISHELL - Invoke the ISPF shell**

**Format**

```
ISHELL [initial_path] [-d]
```

**Note:** An alias of ISHELL is:

```
ISH
```

**Description**

ISHELL invokes the ISPF shell, a panel interface that helps you to set up and manage z/OS UNIX System Services functions.

You can use the ISHELL command to:

- List files in a directory
- Create, delete, or rename directories, files, and special files
• Browse files
• Edit files
• Copy files
• Display file attributes
• Search files for text strings
• Compare files or directories
• Run executable files
• Display the attributes and contents of a symbolic link
• Mount and unmount a z/OS UNIX file system
• Create a z/OS UNIX file system
• Set up character special files
• Set up standard directories for a root file system
• Set up existing users and groups for access to z/OS UNIX System Services

For more information about setting up TSO/E users, see z/OS UNIX System Services Planning.

Some of these tasks require specific authority. For example:

• Mount authority is needed for mounting and unmounting file systems. See the section on mounting file systems in z/OS UNIX System Services Planning for an explanation about the mount authority that is needed for mounting and unmounting file systems.
• Superuser authority is needed when setting up character special files and setting up existing users and groups for z/OS UNIX access.
• RACF SPECIAL attribute is needed when setting up existing users and groups for z/OS UNIX access.

The last path name used on the main panel of ISHELL is kept and displayed again on the next invocation of ISHELL. In order to switch back to the home directory, erase the path name shown and press ENTER.

Field level and panel help are available throughout the dialog. For more information about ISHELL, see z/OS UNIX System Services User's Guide and the online help panels.

Parameters

initial_path
The path that you want to appear in ISHELL's main panel. For example: ishell /tmp/

-d
Prevents ISHELL from suppressing ISPF server dialog errors. Because this option will cause ISHELL to terminate on errors, use it only at the direction of IBM Support.

Usage notes

1. In z/OS V1R11, OEDIT and OBROWSE were changed to use the ISPF edit and browse dialog services by default. To have ISHELL use the original dialog service, export the environment variable BPXWISHISPF=NO from /etc/profile or $HOME/.profile.
2. ISHELL starts a shell process in the background to find out the user's TZ setting from /etc/profile and $HOME/.profile. If the TZ setting is found, this value is used when displaying file time stamps. If not found, GMT is assumed. The environment variable BPXWISHTZ can be set to a time zone value to have ISHELL use a local time zone that is different from your TZ setting. BPXWISHTZ must be specified in /etc/profile or in .profile.
Example

If the TZ setting does not specify GMT, to allow ISHELL users to return to GMT add the following line to etc/profile or .profile:

```
export BPWISHTZ=GMT
```

**MKDIR - Make a directory**

**Format**

```
MKDIR 'directory_name'  MODE(directory_permission_bits) STICKY|NOSTICKY
```

**Description**

Use the MKDIR command to create a directory in the file system.

**Parameters**

**directory_name**

Specifies the name of the directory to be created. The name can be a relative path name or an absolute path name. You must enclose it in single quotation marks. A relative path name is relative to the working directory of the TSO/E session (usually the HOME directory). Therefore, you should usually specify an absolute path name. The name can be up to 1023 characters long. The name is case-sensitive; the system stores each character in the case entered.

All directories in the path name prior to the specified directory must already exist. If the specified directory already exists, no new directory is created.

**MODE(directory_permission_bits)**

Specifies the directory permission bits as three octal numbers, from 0 to 7, separated by commas or blanks. The octal values represent read (r), write (w), and search (x) access for: user, group, and other.

User permission is the permission given to the directory owner. Group permission is the permission given to the group the owner is a member of. Other permission is the permission given to any other user.

The mode for a directory created by MKDIR is determined by the mode itself and applying a umask to it.

The access indicated by each of the numbers 0–7 is:

0  No access
1  Search (x) access
2  Write-only (w) access
3  Write and search (wx) access
4  Read-only (r) access
5  Read and search (rx) access
6  Read and write (rw) access
Read, write, and search (rwx) access

The default permissions set when a directory is created are 755, representing:

- User: read, write, and search permission.
- Group: read and search permission.
- Other: read and search permission.

**STICKY**

Specifies that the sticky bit is to be set on for a directory so a user cannot remove or rename a file in the directory unless one or more of these conditions are true:

- The user owns the file.
- The user owns the directory.
- The user has superuser authority.

**NOSTICKY**

Specifies that the sticky bit is to be set off in the directory. NOSTICKY is the default.

**Return codes**

0

Processing successful.

12

Processing unsuccessful. An error message has been issued.

**Examples**

1. You want to create a directory using an absolute path name giving read, write, and search access to the directory owner and no access to the group and other classes. The new directory name is to be `/tmp/bin`. The directory `/tmp` already exists. Enter:

   ```
   MKDIR '/tmp/bin' MODE(7,0,0)
   ```

2. You want to create a new directory under the working directory of your TSO/E session; therefore you can specify a relative path name. You want to name the new directory `u2`, and to set it up with the default permissions (755). Enter:

   ```
   MKDIR 'u2'
   ```

**MKNOD - Create a character special file**

**Format**

```
MKNOD 'pathname'
MAJOR(device_major_number)
MINOR(device_minor_number)
MODE(file_permission_bits)
```  

**Description**

MKNOD creates a character special file in a file system.

MKNOD can be used only by a superuser.
Parameters

pathname
Specifies the name of the character special file to be created. The name can be a relative path name or an absolute path name. It must be enclosed in single quotes. A relative path name is relative to the working directory of the TSO/E session (usually the HOME directory). Therefore, you should usually specify an absolute path name. The name can be up to 1023 characters long. The name is case-sensitive; the system stores each character in the case entered. This operand is required.

All directories in the path name must exist. If the specified file already exists, no new file is created.

MAJOR(device_major_number)
Specifies the device major number, which can be a decimal number between 0 and 65,535 (64K minus 1). See z/OS UNIX System Services Planning for information about specifying the device major number. This operand is required.

MINOR(device_minor_number)
Specifies the device minor number, which can be a decimal number between 0 and 65,535 (64K minus 1). See z/OS UNIX System Services Planning for information about specifying the device minor number. This operand is required.

MODE(file_permission_bits)
Specifies the file permission bits as three octal numbers, from 0 to 7, separated by commas or blanks. The octal values represent read (r), write (w), and execute (x) access for: user, group, and other.

User permission is the permission given to the file owner. Group permission is the permission given to the group the owner is a member of. Other permission is the permission given to any other user.

The access indicated by each of the numbers from 0 to 7 is:

0
 No access
1
 Search (x) access
2
 Write-only (w) access
3
 Write and execute (wx) access
4
 Read-only (r) access
5
 Read and execute (rx) access
6
 Read and write (rw) access
7
 Read, write, and execute (rwx) access

When the MKNOD command is issued in the TSO interactive environment, the file is created with default permissions of 666, regardless of the user’s umask setting, representing:

6
 User: read and write access
6
 Group: read and write access
6
 Other: read and write access
Examples

1. You want to create a character special file using an absolute path name, giving read, write, and execute access to the file owner and no access to others. The file name is tty1 in the existing directory /dev. The device major number is 2; the minor number is 1. Enter:

   MKNOD '/dev/tty1' MAJOR(2) MINOR(1) MODE(7,0,0)

2. You want to create a character special file named ptty2 in the existing directory /dev. The device major number is 1; the device minor number is 457. You want the default permissions. Enter:

   MKNOD '/dev/ptty2' MAJOR(1) MINOR(457)

3. You want to create a new tty pair using an absolute path name. The file name is ttyp0042 in the existing directory /dev. The device minor number is 42. You want the default permissions. Enter:

   MKNOD '/dev/ptyp0042' MAJOR(1) MINOR(42)
   MKNOD '/dev/ttyp0042' MAJOR(2) MINOR(42)

MOUNT - Logically mount a file system

Format

MOUNT FILESYSTEM(file_system_name)
   MOUNTPOINT(pathname)
   TYPE(file_system_type)
   MODE(RDWR|READ)
   PARM(parameter_string)
   TAG(NOTEXT|TEXT,ccsid)
   SETUID|NOSETUID
   WAIT|NOWAIT
   SECURITY|NOSECURITY
   SYSNAME (sysname)
   AUTOMOVE|AUTOMOVE(indicator,sysname1,sysname2,...,sysnameN)|
   NOAUTOMOVE|UNMOUNT

The Indicator is either INCLUDE or EXCLUDE, which can also be abbreviated as I or E

Description

Use the MOUNT command to logically mount, or add, a mountable file system to the file system hierarchy. You can unmount any mounted file system using the UNMOUNT command. For descriptions of the valid MOUNT parameters for the zFS file system, see Mount in z/OS Distributed File Service zFS Administration.

For options that are specific to the temporary file system (TFS), see Mounting the TFS in z/OS UNIX System Services Planning.

Rule: You must have mount authority before you can issue the MOUNT command. See the section on mounting file systems in z/OS UNIX System Services Planning. The TSO MOUNT and UNMOUNT commands performs privileged operations if the user has read access to the BPX.SUPERUSER resource in the FACILITY class.

filesystem(file_system_name)

Specifies the name of the file system to be added to the file system hierarchy.

file_system_name

For the z/OS UNIX file system, this is the fully qualified name of the z/OS UNIX file system data set that contains the file system. It cannot be a partitioned data set member.

The file system name that is specified must be unique among previously mounted file systems. The file system name that is supplied is changed to all uppercase characters. You can enclose it in single quotation marks, but they are not required.
If file system("file_system_name") is specified, the file system name is not translated to uppercase.

**MOUNTPOINT(pathname)**
Specifies the path name of the mount point directory, the place within the file hierarchy where the file system is to be mounted. This operand is required.

**pathname**
Specifies the path name of the mount point. The path name must be enclosed in single quotation marks. The name can be a relative path name or an absolute path name. A relative path name is relative to the working directory of the TSO/E session (usually the HOME directory). Therefore, you should usually specify an absolute path name. It can be up to 1023 characters long. Path names are case-sensitive, so enter the path name exactly as it is to appear.

When specifying the path name, remember the following rules:
1. The mount point must be a directory. Any files in that directory are inaccessible while the file system is mounted.
2. Only one file system can be mounted to a mount point at any time.

**TYPE(file_system_type)**
Specifies the type of file system that will perform the logical mount request. The system converts the TYPE operand value to uppercase letters. This operand is required.

**file_system_type**
This name must match the TYPE operand of the FILESYSTYPE statement that activates this physical file system in the BPXPRMxx parmlib member. The file_system_type value can be up to 8 characters long.

**MODE(RDWR|READ)**
Specifies the type of access the file system is to be opened for.

**RDWR**
Specifies that the file system is to be mounted for read and write access. RDWR is the default if MODE is omitted.

**READ**
Specifies that the file system is to be mounted for read-only access.

The z/OS UNIX file system allows a file system that is mounted using the MODE(READ) option to be shared as read-only with other systems that share the same DASD.

**PARM('parameter')**
Specifies a parameter string to be passed to the file system type. The parameter format and content are specified by the file system type.

Refer to the following documentation for the appropriate file system-specific options:
- For HFS-specific options, see the BPXPRMxx section in *z/OS MVS Initialization and Tuning Reference*.
- For zFS-specific options, see Mount, in *z/OS Distributed File Service zFS Administration*.
- For NFS-specific options, see Mount processing parameters, in *z/OS Network File System Guide and Reference*.
- For TFS-specific options, see Mounting the TFS, in *z/OS UNIX System Services Planning*.

**TAG(NOTEXT|TEXT,ccsid)**
Specifies whether the file tags for untagged files in the mounted file system are implicitly set. File tagging controls the ability to convert a file's data during file reading and writing. Implicit, in this case, means that the tag is not permanently stored with the file. Rather, the tag is associated with the file during reading or writing, or when stat() type functions are issued. Either TEXT or NOTEXT, and ccsid must be specified when TAG is specified.

When the file system is unmounted, the tags are lost.
NOTE
  Specifies that none of the untagged files in the file system are automatically converted during file reading and writing.

TEXT
  Specifies that each untagged file is implicitly marked as containing pure text data that can be converted.

ccsid
  Identifies the coded character set identifier to be implicitly set for the untagged file. ccsid is specified as a decimal value from 0 to 65535. However, when TEXT is specified, the value must be between 0 and 65535. Other than this, the value is not checked as being valid and the corresponding code page is not checked as being installed.

SETUID|NOSETUID
  Specifies whether the SETUID and SETGID mode bits on executables in this file system are respected. Also determines whether the APF extended attribute or the Program Control extended attribute is honored.

SETUID
  Specifies that the SETUID and SETGID mode bits be respected when a program in this file system is run. SETUID is the default.

NOSETUID
  Specifies that the SETUID and SETGID mode bits not be respected when a program in this file system is run. The program runs as though the SETUID and SETGID mode bits were not set. Also, if you specify the NOSETUID option on MOUNT, the APF extended attribute and the Program Control extended attribute are not honored.

WAIT|NOWAIT
  Specifies whether to wait for an asynchronous mount to complete before returning.

WAIT
  Specifies that MOUNT is to wait for the mount to complete before returning. WAIT is the default.

NOWAIT
  Specifies that if the file system cannot be mounted immediately (for example, a network mount must be done), then the command will return with a return code indicating that an asynchronous mount is in progress.

SECURITY|NOSECURITY
  Specifies whether security checks are to be enforced for files in this file system. When a z/OS UNIX file system is mounted with the NOSECURITY option enabled, any new files or directories that are created are assigned an owner of UID 0, no matter what UID issued the request.

SECURITY
  Specifies that normal security checking is done. SECURITY is the default.

NOSECURITY
  Specifies that security checking will not be enforced for files in this file system. A user can access or change any file or directory in any way.

  Security auditing will still be performed if the installation is auditing successes.

  The SETUID, SETGID, APF, and Program Control attributes may be turned on in files in this file system, but they are not honored while it is mounted with NOSECURITY.

SYSNAME (sysname)
  For systems participating in shared file system, SYSNAME specifies the particular system on which a mount should be performed. This system will then become the owner of the file system mounted. This system must be IPLed with SYSPLEX(YES).

  IBM recommends that you omit the SYSNAME parameter or specify SYSNAME(system_name) where system_name is the name of this system.

sysname
  sysname is a 1–8 alphanumeric name of a system participating in shared file system.
AUTOMOVE(indicator,sysname1,...,sysnameN)|NOAUTOMOVE|UNMOUNT

These parameters apply only in a sysplex where systems are exploiting the shared file system capability. They specify what is to happen to the ownership of a file system when a shutdown, PFS termination, dead system takeover, or file system move occurs. The default setting is AUTOMOVE where the file system will be randomly moved to another system (no system list used).

Indicator is either INCLUDE or EXCLUDE, which can also be abbreviated as I or E

AUTOMOVE
AUTOMOVE indicates that ownership of the file system can be automatically moved to another system participating in a shared file system. AUTOMOVE is the default.

AUTOMOVE(INCLUDE,sysname1,sysname2,...,sysnameN) or AUTOMOVE(I,sysname1,sysname2,...,sysnameN)
The INCLUDE indicator with a system list provides an ordered list of systems to which the file system's ownership could be moved. sysnameN may be a system name, or an asterisk (*). The asterisk acts as a wildcard to allow ownership to move to any other participating system and is only permitted in place of a system name as the last entry of a system list.

AUTOMOVE(EXCLUDE,sysname1,sysname2,...,sysnameN) or AUTOMOVE(E,sysname1,sysname2,...,sysnameN)
The EXCLUDE indicator with a system list provides a list of systems to which the file system's ownership should not be moved.

NOAUTOMOVE
NOAUTOMOVE prevents movement of the file system's ownership in some situations.

UNMOUNT
UNMOUNT allows the file system to be unmounted in some situations.

Follow these guidelines when unmounting the file system:
1. Define your version and sysplex root file systems as AUTOMOVE, and define your system-specific file systems as UNMOUNT.
2. Do not define a file system as NOAUTOMOVE or UNMOUNT and a file system underneath is as AUTOMOVE; in this case, the file system defined as AUTOMOVE will not be recovered after a system failure until the failing system is restarted.

See z/OS UNIX System Services Planning for more information about Sharing file systems in a sysplex.

Usage notes
1. The /samples directory contains sample MOUNT commands (called mountx).
2. When the mount is done asynchronously (NOWAIT was specified and return code 4 was returned), you can determine if the mount has completed with one of the following:
   • The df shell command
   • The DISPLAY OMVS,F operator command (see z/OS MVS System Commands)
   • The MOUNT table option on the File Systems pull-down in the ISPF Shell (accessed by the ISHELL command)
3. In order to mount a file system as the system root file system, the caller must be a superuser. Also, a file system can only be mounted as the system root file system if the root file system was previously unmounted.
4. If you have previously unmounted the root file system, a dummy file system or SYSROOT is displayed as the current root file system. During the time when SYSROOT is displayed as the root, any operation that requires a valid file system will fail. When you subsequently mount a new root file system on mount point /, that new file system will replace SYSROOT. When a new root file system has been
mounted, you should terminate any current dubbed users or issue a `chdir` command, using a full path name to the appropriate directory. This way, the users can access the new root file system. Otherwise, an error will occur when a request is made requiring a valid file system.

5. Systems exploiting shared file system will have I/O to an OMVS couple data set. Because of these I/O operations to the CDS, each mount request requires additional system overhead. You will need to consider the effect that this will have on your recovery time if a large number of mounts are required on any system participating in shared file system.

6. The TAG parameter is intended for file systems that don't support storing the file tag, such as NFS remote file systems.

7. Do not use the TAG parameter simultaneously with the NFS Client Xlate option. If you do, the mount will fail.

8. The UNMOUNT keyword is not available to automounted file systems.

**File system recovery and TSO MOUNT**

File system recovery in a shared file system environment takes into consideration file system specifications such as AUTOMOVE | NOAUTOMOVE | UNMOUNT, and whether or not the file system is mounted read-only or read/write.

Generally, when an owning system fails, ownership over its AUTOMOVE mounted file systems is moved to another system and the file is usable. However, if a file system is mounted read/write and the owning system fails, then all file system operations for files in that file system will fail. This is because data integrity is lost when the file system owner fails. All files should be closed (BPX1CLO) and reopened (BPX1OPN) when the file system is recovered. (The BPX1CLO and BPX1OPN callable services are discussed in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*).

For file systems that are mounted read-only, specific I/O operations that were in progress at the time the file system owner failed may need to be re-attempted. Otherwise, the file system is usable.

In some situations, even though a file system is mounted AUTOMOVE, ownership of the file system may not be immediately moved to another system. This may occur, for example, when a physical I/O path from another system to the volume where the file system resides is not available. As a result, the file system becomes unowned (the system will issue message BPXF213E when this occurs). This is true if the file system is mounted either read/write or read-only. The file system still exists in the file system hierarchy so that any dependent file systems that are owned by another system are still usable.

However, all file operations for the unowned file system will fail until a new owner is established. The shared file system support will continue to attempt recovery of AUTOMOVE file systems on all systems in the sysplex that are enabled for shared file system. Should a subsequent recovery attempt succeed, the file system transitions from the unowned to the active state.

Applications using files in unowned file systems must close (BPX1CLO) those files and reopen (BPX1OPN) them after the file system is recovered.

File systems that are mounted NOAUTOMOVE will become unowned when the file system owner exits the sysplex. The file system will remain unowned until the original owning system restarts or until the unowned file system is unmounted. Since the file system still exists in the file system hierarchy, the file system mount point is still in use.

An unowned file system is a mounted file system that does not have an owner. The file system still exists in the file system hierarchy. As such, you can recover or unmount an unowned file system.

File systems associated with a 'never move' PFS will be unmounted during dead system recovery. For example, TFS is a 'never move' PFS and will be unmounted, as well as any file systems mounted on it, when the owning system leaves the sysplex.

As stated in “Usage notes” on page 894, the UNMOUNT keyword is not available to automounted file systems. However, during dead system recovery processing for an automounted file system (whose owner
is the dead system), the file system is unmounted if it is not being referenced by any other system in the sysplex.

Return codes

0
Processing successful.

4
Processing incomplete. An asynchronous mount is in progress.

12
Processing unsuccessful. An error message has been issued.

Examples

1. To mount HFS.WORKDS on the directory /u/openuser, enter:

   MOUNT filesystem('HFS.WORKDS') MOUNTPOINT('/u/openuser') TYPE(HFS)

2. The following example mounts the z/OS UNIX directory /u/shared_data, which resides on the remote host named mvshost1, onto the local directory /u/jones/mnt. The command may return before the mount is complete, allowing the mount to be processed in parallel with other work. The SETUID and SETGID bits are honored on any executable programs:

   MOUNT filesystem('MVSHOST1.SHARE.DATA') MOUNTPOINT('/u/jones/mnt') TYPE(NFSC) PARM('mvshost1:/hfs/u/shared_data') NOWAIT SETUID

3. Examples for using the TAG parameter are:

   TAG(TEXT,819) identifies text files containing ASCII (ISO-8859-1) data.

   TAG(TEXT,1047) identifies text files containing EBCDIC (ISO-1047) data.

   TAG(NOTEXT,65535) tags files as containing binary or unknown data.

   TAG(NOTEXT,0) is the equivalent of not specifying the TAG parameter at all.

   TAG(NOTEXT,273) tags files with the German code set (ISO-273), but is ineligible for automatic conversion.

OBROWSE - Browse a z/OS UNIX file

Format

OBrowse [-r xx] pathname

or

OBrowse -r xx 'pathname'

or

OBrowse

The path name is optional in the last example.
Description
The OBROWSE command browses files in the z/OS UNIX file system using the ISPF Browse facility.
If you enter the OBROWSE command without specifying a path name, the Browse Entry panel is displayed. From that panel, you can enter the directory name and file name of an existing file you want to browse. If you are browsing fixed-length records, you must also indicate the record length.

Parameters
pathname
Specifies the path name of the file to be browsed. The path name can be absolute or relative. It can be enclosed in single quotation marks. A relative path name is relative to the working directory of the TSO/E session (usually the HOME directory). Therefore, you should usually specify an absolute path name. If you enter the OBROWSE command from the shell, use the absolute path name. Avoid using spaces or single quotation marks within the path name.

Options
-o
By default, starting in V1R11, the ISPF browse dialog service is used when browsing z/OS UNIX files. Specify -o if you want OBROWSE to use the original dialog service.

-r xx
Sets the record length to be browsed for fixed-length text files. xx is length. If -r xx is specified, the file is processed as fixed-length records. This lets you convert a variable length file to fixed length for viewing.

OCOPY - Copy an MVS data set member or z/OS UNIX file to another member or file

Format

```
OCOPY INDD(ddname1) OUTDD(ddname2)
BINARY | TEXT
CONVERT(character_conversion_table | YES | NO)
PATHOPTS (USE|OVERRIDE)
TO1047 | FROM1047
```

Description
You can use the OCOPY command to copy data between an MVS data set and the z/OS UNIX file system. For OCOPY, you need to use CONVERT for these two situations:

- Conversion between code pages IBM-037 and IBM-1047
- Conversion between ASCII and code page IBM-1047

The z/OS shell uses code page 1047, and MVS uses a country extended code page. You can convert data to or from code page 1047 while it is being copied.

If you are copying a file with double-byte data, do not use the CONVERT option.

Before using the OCOPY command, you must allocate the data set or file you are working with. When using the TSO/E ALLOCATE command or a JCL DD statement to allocate a file or data set, you can specify PATHMODE and PATHOPTS parameters along with the PATH parameter. For information about the use of these parameters with the JCL statement, see z/OS MVS JCL Reference. For information about the TSO/E ALLOCATE command, see z/OS TSO/E Command Reference.

You can use OCOPY to copy:

- A member of a partitioned data set (PDS or PDSE) to a file
OCOPY

- An MVS sequential data set to a file
- A file to a member of a PDS or PDSE
- A file to a sequential data set
- A file to a file
- A member of a PDS or PDSE to another member of a PDS or PDSE
- A member of a PDS or PDSE to a sequential data set
- An MVS sequential data set to another sequential data set
- An MVS sequential data set to a member of a PDS or PDSE

Both INDD and OUTDD can represent an MVS data set or a file. If the source (INDD) is an MVS data set and the target (OUTDD) is a z/OS UNIX file, then OCOPY copies an MVS data set to a file; the operation is the same as the OPUT command. If the source (INDD) is a z/OS UNIX file and the target (OUTDD) is an MVS data set, then OCOPY copies a file to an MVS data set; the operation is the same as the OGET command.

Both the target and source can be an MVS data set or member of a partitioned data set, or both can be a file. This function is typically used for code page conversion.

If PATHMODE, which sets the permission bits for a new file, is specified during allocation, it is used when creating a new file. If PATHMODE is not specified during the allocation of a new file, the allocation creates a file with the default permission of 000, which means the user has no access to it.

Parameters

INDD(ddname1)
- Specifies the ddname of the source. The ddname is up to 8 characters long.

OUTDD(ddname2)
- Specifies the ddname of the target. The ddname is up to 8 characters long.

BINARY | TEXT
- Specifies that the data to be copied is a binary file or text file.
  - BINARY
    - Specifies that the data to be copied is a binary file. The default is binary when copying a data set of undefined record format to a file.
    - When you specify BINARY, OCOPY operates without any consideration for <newline> characters or the special characteristics of DBCS data. For example, double-byte characters might be split between MVS data set records, or a “shift-out” state might span records.
  - TEXT
    - Specifies that the data to be copied is a text file. The default is text except when copying a data set of undefined record format to a file.
    - If you are using a DBCS-supported terminal, you should use TEXT. It is assumed that double-byte data in the file system includes the <newline> character in order to delineate line boundaries. Data within these lines that are delineated by <newline> characters must begin and end in the “shift-in” state.

CONVERT(character_conversion_table | YES | NO)
- Specifies the character conversion table used to convert between the following:
  - Code pages IBM-037 and IBM-1047
  - The ASCII code page and IBM-1047
- If this optional operand is omitted, the system copies the data without conversion.
- Use this option for single-byte data only.
  - Specify the CONVERT value as one of the following:
character_conversion_table

Specify one of the following:

- **data_set_name(member_name)**. Specifies the name of the partitioned data set (library) and the name of the member that contains the character conversion table.

- **data_set_name**. Specifies the name of the partitioned data set (library) that contains the character conversion table as the default member. The default member name is BPXFX000. (BPXFX000 is an alias; when shipped by IBM, it points to BPXFX111.) A fully qualified data set name must be enclosed in single quotes.

- **(member_name)**. Specifies the name of the conversion table to be used. It is a member of a PDS. Since the data_set_name is omitted, the standard library concatenation is searched for the table. (The default library is SYS1.LINKLIB.)

The following list summarizes what you can specify when you want to convert data to a different code page when copying single-byte data:

- **BPXFX100**. Null character conversion table. Use this table if the square brackets at your workstation are at the same code points as the square brackets on code page 1047 (it is the default). Also use this table if you are using a DBCS terminal.

- **BPXFX111**. Specifies a non-APL conversion table to convert between code pages IBM-037 and IBM-1047.

- **BPXFX211**. Specifies an APL conversion table to convert between code pages IBM-037 and IBM-1047.

- **BPXFX311**. Specifies an ASCII-EBCDIC conversion table to convert between code pages ISO8859-1 and IBM-1047.

**YES**

Specifies that the system is to perform conversion and use the default conversion table (BPXFX000) in the system library concatenation. (BPXFX000 is an alias; when shipped by IBM, it points to BPXFX111.)

**NO**

Specifies that conversion not be done. NO is the same as omitting the CONVERT operand.

**PATHOPTS(USE | OVERRIDE)**

Specifies whether the OCOPY should use or override the PATHOPTS value specified during allocation. If the PATHOPTS is not specified in the allocation, OCOPY will open the file with the appropriate PATHOPTS.

**USE**

Specifies that the PATHOPTS value is to be enforced. If a file that was identified as read-only when it was allocated is identified as the output file for OCOPY, OCOPY fails. Similarly, if a write-only file is specified as the input file, OCOPY fails. USE is the default.

**OVERRIDE**

Specifies that the PATHOPTS value specified during allocation is to be ignored.

**TO1047 | FROM1047**

**TO1047**

Specifies that the TO section of the character conversion table is to be used. This is typically used to convert from code page IBM-037 or ASCII to code page IBM-1047.

**FROM1047**

Specifies that the FROM section of the conversion table is to be used. This is typically used to convert from code page IBM-1047 to code page IBM-037 or ASCII.

If the CONVERT operand is specified and this operand is omitted,

- Data copied from an MVS data set to a file uses the TO1047 section of the table.
- Data copied from a file to an MVS data set uses the FROM1047 section of the table.
If the CONVERT operand is specified for a copy from a file to a file or an MVS data set to an MVS data set, you must specify either TO1047 or FROM1047.

Usage notes

1. You can use OCOPY to copy a program object from a PDSE to the file system, and it will be executable there. If you have a load module in a partitioned data set, however, you must first use the IEBCOPY program to copy the load module from a partitioned data set to a PDSE and then subsequently use OCOPY to copy the module into the file system. The IEBCOPY converts the load module to a program object.

   Note: You can use the linkage editor to put the load module directly into the file system.

2. An executable file copied from the file system into an MVS data set is not executable under MVS. Some required directory information is lost during the copy. See z/OS UNIX System Services User’s Guide for a discussion of copying executable files.

3. Data sets with spanned records are not allowed.

4. When you are copying into an existing file, data is appended to the end of the file if OAPPEND is specified in PATHOPTS. Otherwise, the existing file is overwritten.

5. Copying from text files in the z/OS UNIX file system to MVS data sets:
   - For text files, all <newline> characters are stripped during the copy. Each line in the file ending with a <newline> character is copied into a record of the MVS data set. You cannot copy a text file to an MVS data set in an undefined record format.
     - For an MVS data set in fixed record format: Any line longer than the record size is truncated. If the line is shorter than the record size, the record is padded with blanks.
     - For an MVS data set in variable record format: Any line longer than the largest record size is truncated and the record length is set accordingly. A change in the record length also occurs if the line is short.

6. Copying from binary files in the z/OS UNIX file system to MVS data sets:
   - For binary files, all data is preserved.
     - For an MVS data set in fixed record format: Data is cut into chunks of size equal to the record length. Each chunk is put into one record. The last record is padded with spaces or blanks.
     - For an MVS data set in variable record format: Data is cut into chunks of size equal to the largest record length. Each chunk is put into one record. The length of the last record is equal to the length of the data left.
     - For an MVS data set in undefined record format: Data is cut into chunks of size equal to the block size. Each chunk is put into one record. The length of the last record is equal to the length of the data left.

7. When you copy MVS data sets to text files in the z/OS UNIX file system, a <newline> character is appended to the end of each record. If trailing blanks exist in the record, the <newline> character is appended after the trailing blanks. MVS fixed block data sets have a fixed record length, which means that trailing blanks could exist up to the end of each record.

8. When you copy MVS data sets to binary files in the z/OS UNIX file system, the <newline> character is not appended to the record.

Return codes

0
   Processing successful.

12
   Processing unsuccessful. An error message has been issued.
Examples

1. The following commands copy an MVS sequential data set to a z/OS UNIX file. This is text data, and there is no code page conversion.
   - SYSUT1 is the ddname of the source data set, EMPLOYEE.DATA.
   - PATHNAME is the ddname of the target, which is the existing file /u/admin/employee/data.

   ```
   ALLOCATE FILE(sysut1) DATASET('employee.data')
   ALLOCATE FILE(pathname) PATH('/u/admin/employee/data')
   OCOPY INDD(sysut1) OUTDD(pathname) TEXT
   ```

2. The following commands copy a binary file into a member of a partitioned data set:
   - BINARY is the ddname of the source file, bin/payroll. This file is in the working directory.
   - MVSPDS is the ddname of the target data set member, APPL.CODES(PAYROLL)

   ```
   ALLOCATE FILE(binary) PATH('/bin/payroll')
   ALLOCATE FILE(mvspds) DATASET('appl.codes(payroll)')
   OCOPY INDD(binary) OUTDD(mvspds) BINARY
   ```

3. The following commands copy system input from the MVS SYSIN data set to the file system and perform code page conversion:
   - SYSIN is the ddname of the source, IBMUSR.EMPLOYEE.DATA.
   - PATHNAME is the ddname of the target, /u/admin/employee/data. This file does not currently exist and is created by ALLOCATE.
   - This is text data.
   - The character conversion table is the default table, member BPXFX000 of the SYS1.BPXLATE data set. (BPXFX000 is an alias; when shipped by IBM, it points to BPXFX111.)
   - Because this is a copy from an MVS data set to a file, the section TO1047 of the conversion table is used by default.

   ```
   ALLOCATE FILE(sysin) DATASET('IBMUSR.EMPLOYEE.DATA')
   ALLOCATE FILE(pathname) PATH('/u/admin/employee/data')
   PATHMODE (sirwxu) PATHOPTS (ocreat, owronly)
   OCOPY INDD(sysin) OUTDD(pathname) TEXT CONVERT((BPXFX000))
   ```

4. The following OCOPY command copies data from one MVS sequential data set to another MVS sequential data set and performs code page conversion. This example shows just the OCOPY command; the necessary ALLOCATE commands are not included.
   - SYSUT1 is the ddname of the source data set.
   - TRANSDDD is the ddname of the target data set.
   - This is text data.
   - The data is converted using the user-specified character conversion table and the TO1047 section of the table.

   ```
   OCOPY INDD(sysut1) OUTDD(transdd) TEXT CONVERT('sys1.mylib(mytab)') TO1047
   ```

---

**OEDIT - Edit a z/OS UNIX file**

**Format**

```
OEDIT [-r xx] pathname
```
Description

The OEDIT command uses the ISPF Edit facility to edit files in the z/OS UNIX file system.

If you enter OEDIT without specifying a path name, the Edit Entry panel is displayed. From that panel, you can enter the directory name and file name of an existing file, or you can specify a directory name and file name for a new file. The Edit Entry panel also lets you specify an edit profile and an initial edit macro.

Parameters

**pathname**

Specifies the path name of the file to be edited. The path name can be absolute or relative. It can be enclosed in single quotation marks. A relative path name is relative to the working directory of the TSO/E session (typically the HOME directory). Therefore, you should typically specify an absolute path name. If you enter OEDIT from the shell, use the absolute path name. Do not use spaces or single quotation marks within path names.

Options

**-o**

By default, starting in V1R11, the ISPF edit dialog service is used when browsing z/OS UNIX files. Specify -o if you want OEDIT to use the original dialog service.

If the -r xx option flag is also specified, the -o option flag must be specified first.

**-r xx**

Set the record length to be edited for fixed length text files. xx is the record length.

If -r xx is specified, the file is processed as variable length but loaded into the editor as fixed-length records and saved as fixed-length records. This lets you convert a variable length file to fixed length. If any lines are longer than the specified record length, the edit session will not load the file and will issue the customary message that a line is too long.

Usage notes

1. All directories in pathname, starting from the root, must be searchable.
2. ASCII files must be tagged as ISO8859-1 in order for OEDIT to automatically convert the file. Do not enter the OEDIT session and type SOURCE ASCII.
3. OEDIT attempts to load the file into a VB255 session. If this is an ISPF that supports wide edit (such as ISPF 4.1) and any line exceeds 235 characters, the width for the new session is the length of the longest line plus 25% to allow for some expansion.
4. The COPY command cannot copy in files that have records wider than the edit session.
5. The TSO region size must be large enough to hold the size of the file to be edited.
6. Two ISPF variables are available to edit macros:
   - HFSCWD this variable contains the path name for the directory in which the file being edited resides.
   - HFSNAME this variable contains the name of the file that is being edited.
OGET - Copy z/OS UNIX files into an MVS data set

Format

```
OGET 'pathname'
mvs_data_set_name | mvs_data_set_name(member_name)
BINARY | TEXT
CONVERT(character_conversion_table | YES | NO)
```

Description

You can use the OGET command to copy a z/OS UNIX file:

- To a member of an MVS partitioned data set (PDS or PDSE)
- To an MVS sequential data set

and convert the data from code page 1047 to code page IBM-037 or ASCII while it is being copied. Do not use the CONVERT option when copying files that contain double-byte data. This option is used for single-byte data only, not for double-byte data.

Parameters

pathname

Specifies the path name of the file that is being copied to a data set. This operand is required. The path name is:

- A relative or absolute path name. A relative path name is relative to the working directory of the TSO/E session (usually the HOME directory). Therefore, you should usually specify an absolute path name.
- Up to 1023 characters long.
- Enclosed in single quotation marks.
- In uppercase or lowercase characters, which are not changed by the system.

mvs_data_set_name | mvs_data_set_name(member_name)

Specifies the name of an MVS sequential data set or an MVS partitioned data set member to receive the file that is being copied. One of these two operands is required. The data set name is:

- A fully qualified name that is enclosed in single quotation marks, or an unqualified name.
- Up to 44 characters long.
- Converted to uppercase letters by the system.

BINARY | TEXT

Specifies whether the file being copied contains binary data or text.

BINARY

Specifies that the file being copied contains binary data.

When you specify BINARY, OGET operates without any consideration for <newline> characters or the special characteristics of DBCS data. For example, double-byte characters might be split between MVS data set records, or a “shift-out” state might span records.

TEXT

Specifies that the file being copied contains text. This is the default.

If you are using a DBCS-supported terminal, you should use TEXT. It is assumed that double-byte data in the file system includes the <newline> character in order to delineate line boundaries. Data within these lines that are delineated by <newline> characters must begin and end in the “shift-in” state.
CONVERT(character_conversion_table | YES | NO)

Specifies that the data being copied is to be converted from IBM-1047 to IBM-037 or ASCII. This operand is optional. If it is omitted, the system copies the data without conversion.

Use this option for single-byte data only.

Specify the CONVERT value as one of the following:

**character_conversion_table**

Specify one of the following:

- `data_set_name(member_name)`. Specifies the name of the partitioned data set (PDS) and the name of the member that contains the character conversion table.
- `data_set_name`. Specifies the name of the partitioned data set (PDS) that contains the character conversion table. The table is the FROM1047 part in member BPXF000. (This is an alias; when shipped by IBM, it points to BPXF111.)
- `(member_name)`. Specifies the name of the conversion table to be used. It is a member of a PDS. Since the `data_set_name` is omitted, the standard library concatenation is searched for the table. (The default library is SYS1.LINKLIB.)

The following list summarizes what you can specify when you want to convert data to a different code page when copying single-byte data:

- BPXF100. Null character conversion table. Use this table if the square brackets at your workstation are at the same code points as the square brackets on code page 1047 (it is the default). Also use this table if you are using a DBCS terminal.
- BPXF111. Specifies a non-APL conversion table to convert between code pages IBM-037 and IBM-1047.
- BPXF211. Specifies an APL conversion table to convert between code pages IBM-037 and IBM-1047.
- BPXF311. Specifies an ASCII-EBCDIC conversion table to convert between code pages ISO8859-1 and IBM-1047.

**YES**

Specifies that the system is to perform conversion and use the default conversion table (BPXF000) in the standard library concatenation. (BPXF000 is an alias; when shipped by IBM, it points to BPXF111.)

**NO**

Specifies that conversion not be done. NO is the same as omitting the CONVERT operand.

Do not use the CONVERT parameter on files containing double-byte data. double-byte data in the file system is in code page 939. If you need to convert to a code page other than 939, you use the `iconv` command.

**Usage notes**

1. For text files, all `<newline>` characters are stripped during the copy. Each line in the file ending with a `<newline>` character is copied into a record of the MVS data set. You cannot copy a text file to an MVS data set in an undefined record format.

   - For an MVS data set in fixed record format: Any line longer than the record size is truncated. If the line is shorter than the record size, the record is padded with blanks.
   - For an MVS data set in variable record format: Any line longer than the largest record size is truncated; the record length is set to the length of the line. A change in the record length also occurs if the line is short.

   For text mode transfer, if the line is longer than the record size, the line is truncated (for DBCS, perhaps in the middle of a double-byte character or in “shift-in” state). If the line is shorter than the record size, the record is padded with blanks.

2. For binary files, all data is preserved.
• For an MVS data set in fixed record format: Data is cut into chunks of size equal to the record length. Each chunk is put into one record. The last record is padded with spaces or blanks.

• For an MVS data set in variable record format: Data is cut into chunks of size equal to the largest record length. Each chunk is put into one record. The length of the last record is equal to the length of the data left.

• For an MVS data set in undefined record format: Data is cut into chunks of size equal to the block size. Each chunk is put into one record. The length of the last record is equal to the length of the data left.

For binary mode transfers, double-byte characters might be split between MVS data set records, or a "shift-out" state might span records.

3. If the MVS data set does not exist, OGET allocates a new data set, a sequential data set of variable record format. However, OGET does not allocate a new partitioned data set. The record length of the new data set is either 255 or the size of the longest line in the z/OS UNIX file system file, whichever is larger. Dynamic allocation services determine the block size and space, based on installation-defined defaults. If the defaults are not sufficient, you should allocate a new MVS data set and then specify it on OGET.

A simple method of allocating a sufficient size is to specify a primary extent size and a secondary extent size equal to the number of bytes in the file being copied.

4. An executable file copied into an MVS data set is not executable under MVS, because some required directory information is lost during the copy to the partitioned data set.

5. Data sets with spanned records are not allowed.

6. If you are using a DBCS-supported terminal, the target MVS data set should be defined or defaulted to variable record format. The record length of the data set must be greater than or equal to the longest line in the z/OS UNIX file system file. (OGET can determine how long the longest line is if you ask it to allocate the target data set.)

7. OGET cannot be used to copy a load module out of a partitioned data set and into a file system. You have to use a binder to "flatten" the load module. For more information about copying load modules, see z/OS UNIX System Services User's Guide

Return codes

0
  Processing successful.

12
  Processing unsuccessful. An error message has been issued.

Examples

1. The following command copies a text file to an MVS sequential data set, converting from code page 1047 to code page IBM-037 using the default table BPXFX000. (BPXFX000 is an alias; when shipped by IBM, it points to BPXFX111.)

   • The path name of the file is /u/admin/employee/data.
   • The unqualified name of the sequential data set is EMPLOYEE.DATA.

   OGET '/u/admin/employee/data' EMPLOYEE.DATA TEXT CONVERT(YES)

2. The following command copies a text file to an MVS sequential data set, converting from code page 1047 to code page IBM-037 using conversion table BPXFX111 in the user's library data set.

   • The path name of the file is /u/admin/employee/data.
   • The fully qualified name of the sequential data set is IBMUSR.EMPLOYEE.DATA.

   OGET '/u/admin/employee/data' 'IBMUSR.EMPLOYEE.DATA' CONVERT('MY.LOADLIB(BPXFX111)')
OGETX - Copy z/OS UNIX files from a directory to an MVS PDS or PDSE

Format

<table>
<thead>
<tr>
<th>OGETX</th>
<th>hfs_directory</th>
<th>hfs_file_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mvs_PDS_name</td>
<td>mvs_data_set_name(member_name)</td>
</tr>
<tr>
<td>ASIS</td>
<td>BINARY</td>
<td>TEXT</td>
</tr>
<tr>
<td>CONVERT(character_conversion_table</td>
<td>YES</td>
<td>NO)</td>
</tr>
<tr>
<td>LC</td>
<td>QUIET</td>
<td></td>
</tr>
<tr>
<td>SUFFIX(suffix)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

You can use the OGETX command to:

- Copy files in a z/OS UNIX system directory to members of a partitioned data set (PDS) or PDSE
- Copy an individual file to a sequential data set or member of a partitioned data set

and convert the data from code page 1047 to code page IBM-037 or ASCII while it is being copied.

Do not use the CONVERT option when copying files that contain double-byte data. This option is used for single-byte data only, not for double-byte data.

This command uses the ISPF/PDF Edit facility.

Parameters

**hfs_directory | hfs_file_name**

Specifies the path name of the z/OS UNIX directory or file name that is being copied to an MVS PDS or PDSE. The files are copied into members of the PDS or PDSE.

Use hfs_directory when a PDS is specified. When a sequential data set or PDS member is specified, then the file name must be used.

These limitations apply to an MVS data set name:

- It can use uppercase alphabetic characters A through Z, but not lowercase letters.
- It can use numeric characters 0 through 9, and the special characters @, #, and $.
- It cannot begin with a numeric character.
- The member name cannot be more than 8 characters. If a file name is longer than 8 characters or uses characters that are not allowed in an MVS data set name, the file is not copied.

The LC operand lets you copy z/OS UNIX system file names that are lowercase, mixed case, or uppercase.

Single quotation marks around the directory name or file name are optional.

**mvs_PDS_name | mvs_data_set_name(member_name)**

mvs_PDS_name specifies the name of an MVS PDS or PDSE to receive the z/OS UNIX system files that are being copied. mvs_data_set_name(member_name) specifies the name of an MVS partitioned data set member to receive the file that is being copied. The name is:

- A fully qualified name that is enclosed in single quotation marks, or an unqualified name
- Up to 44 characters long, with an additional 8 characters for the member name
- Converted to uppercase letters

**ASIS**

Specifies that the _ character in path names not be translated to the @ character in member names. (It is a common convention to use @ symbols in PDS member names to correspond with the _ symbol in path names.)
**BINARY | TEXT**

Specifies whether the files in the directory being copied contains binary data or text. For more information, see Note “7” on page 908.

**BINARY**

Specifies that the files in the directory being copied contains binary data.

When you specify BINARY, OGET operates without any consideration for <newline> characters or the special characteristics of DBCS data. For example, double-byte characters might be split between MVS data set records, or a “shift-out” state might span records.

**TEXT**

Specifies that the files in the directory being copied contains text. This is the default.

If you are using a DBCS-supported terminal, you should use TEXT. It is assumed that double-byte data in the file system includes the <newline> character in order to delineate line boundaries. Data within these lines that are delineated by <newline> characters must begin and end in the “shift-in” state.

**CONVERT(character_conversion_table | YES | NO)**

Specifies that the data being copied is to be converted from code page 1047 to code page IBM-237 or ASCII. That is, the FROM1047 part of the specified character conversion table is used. This operand is optional. If it is omitted, the system copies the data without conversion.

Use this option for single-byte data only.

Specify the CONVERT value as one of the following:

**character_conversion_table**

Specify one of the following:

- `data_set_name(member_name)`. Specifies the name of the partitioned data set (PDS) and the name of the member that contains the character conversion table.
- `data_set_name`. Specifies the name of the partitioned data set (PDS) that contains the character conversion table. The table is the FROM1047 part in member BPXFX000. (This is an alias; when shipped by IBM, it points to BPXFX111.)
- `(member_name)`. Specifies the name of the conversion table to be used. It is a member of a PDS. Because the `data_set_name` is omitted, the standard library concatenation is searched for the table. (The default library is SYS1.LINKLIB.)

The following list summarizes what you can specify when you want to convert data to a different code page when copying single-byte data:

**BPXFX100**

Null character conversion table. Use this table if the square brackets at your workstation are at the same code points as the square brackets on code page 1047 (it is the default). Also use it if you are using a DBCS terminal.

**BPXFX111**

Specifies a non-APL conversion table to convert between code pages IBM-037 and IBM-1047.

**BPXFX211**

Specifies an APL conversion table to convert between code pages IBM-037 and IBM-1047.

**BPXFX311**

Specifies an ASCII-EBCDIC conversion table to convert between code pages ISO8859-1 and IBM-1047.

**NO**

Specifies that conversion not be done. NO is the same as omitting the CONVERT operand.

Do not use the CONVERT parameter on files containing double-byte data. Double-byte data in the file system is in code page 939. If conversion to a code page other than 939 is required, you should use the `iconv` command.
YES
Specifies that the system is to perform conversion and use the default conversion table (BPXFX000) in the standard library concatenation. (BPXFX000 is an alias; when shipped by IBM, it points to BPXFX111.)

LC
Specifies that lowercase or mixed-case z/OS UNIX file names are to be converted to uppercase member names. If LC is not specified, file names containing lowercase characters are not found.

QUIET
Turns off the echoing of the OGET command before a file is copied.

SUFFIX(suffix)
Specifies that files with suffix are to be copied. The suffix and the separating period are dropped from the file name when the PDS members are created.

A suffix is an optional additional file identifier that is appended to suffix the file name following the last period (.). It is typically used to identify the type of file.

Usage notes
1. Avoid using OGETX with path names that contain single quotation marks or spaces.
2. For text files, all <newline> characters are stripped during the copy. Each line in the file ending with a <newline> character is copied into a record of the MVS data set. You cannot copy a text file to an MVS data set in an undefined record format.
   - For an MVS data set in fixed record format, any line longer than the record size is truncated. If the line is shorter than the record size, the record is padded with blanks.
   - For an MVS data set in variable record format, any line is longer than the largest record size is truncated; the record length is set accordingly. A change in the record length also occurs if the line is short.
3. For binary files, all data is preserved.
   - For an MVS data set in fixed record format, data is cut into chunks of size equal to the record length. Each chunk is put into one record. The last record is padded with spaces or blanks.
   - For an MVS data set in variable record format, data is cut into chunks of size equal to the largest record length. Each chunk is put into one record. The length of the last record is equal to the length of the data left.
   - For an MVS data set in undefined record format, data is cut into chunks of size equal to the block size. Each chunk is put into one record. The length of the last record is equal to the length of the data left.
4. Data sets with spanned records are not allowed.
5. Before the copy, the OGET command for a file is echoed, unless you specify the QUIET option. If you did not specify QUIET and if the command is not echoed for a file, it has not met the copy criteria and is not copied.
6. If more than one file name is the same, the file is overwritten on each subsequent copy. For example, if you specify a copy of Pgma and pgma and use LC, the first file copied is overwritten. Or if you copy pgma.h and pgma.c and specify SUFFIX, the first file copied is overwritten.
7. If the target data set is a PDS with an undefined record format, the files may be treated as load modules. A load module is copied by link-editing it into the target library. For the program to be able to execute, the entry point must be at the beginning of the load module.
   For OGETX to treat the file as a load module, do not specify either TEXT nor BINARY.
8. If the source for the copy is a file, the target can be specified as a PDS. The member name used is the file name, which is in uppercase and has had any suffixes removed. Any remaining characters in the member name that are not valid in member names cause the copy to fail. You do not have to specify a file as a target with a sequential data set, or a directory as a target with a PDS. The ASIS option is not affected.
Examples

1. The following command copies the files in the z/OS UNIX system directory /usr/sbllib to the MVS PDS named DATAFILE, removing any suffixes appended to the z/OS UNIX system files and accepting lowercase file names.

   OGETX /usr/sbllib/ DATAFILE LC SUFFIX

   The members /usr/sbllib/program1.c, /usr/sbllib/list.prg, and usr/sbllib/program2.c become DATAFILE(PROGRAM1), DATAFILE(LIST), and DATAFILE(PROGRAM2).

2. The following command copies the files with the suffix of c in the z/OS UNIX system directory /usr/sbllib to the MVS PDS named DATAFILE, removing the .c suffix appended to the z/OS UNIX system files and accepting lowercase file names.

   OGETX /usr/sbllib/ DATAFILE LC SUFFIX(c)

   The members /usr/sbllib/program1.c, /usr/sbllib/list.prg, and usr/sbllib/program2.c become DATAFILE(PROGRAM1) and DATAFILE(PROGRAM2).

OMVS - Invoke the z/OS shell

Format

OMVS ALARM | NOALARM
 AUTOSCROLL | NOAUTOSCROLL
 CONVERT(character_conversion_table)
 DBCS | NODBCS
 DEBUG(NO | YES | EVENT | DATA)
 ECHO | NOECHO
 ENDPASSTHROUGH(ATTN | CLEAR | CLEARPARTITION | ENTER | NO | PA1 | PA3 | PF1 | PF2 | PF3 ... PF24 | SEL)
 ESCAPE('escape-characters')
 LINES(n)
 PFn
 (ALARM | NOALARM | AUTOSCROLL | NOAUTOSCROLL | BACKSCR | BOTTOM | CLOSE | CONTROL | ECHO | NOECHO | FWDRETR | HALFSCR | HELP | HIDE | NOHIDE | NEXTSESS | NO | OPEN | PFSHOW | NOPFSHOW | PREVSESS | QUIT | QUITALL | REFRESH | RETRIEVE | RETURN | SCROLL | SUBCOMMAND | TOP | TSO )
 PFSHOW | NOPFSHOW
 RUNOPTS('LE/370-runtime-options')
 SESSIONS(n)
 SHAREAS | NOSHAREAS
 WRAPDEBUG(n)
Description

Use the OMVS command to invoke the z/OS shell. You can select options on the OMVS command to customize aspects of the shell interface, such as the function keys.

After you are working in a shell session, you can switch to subcommand mode, return temporarily to TSO/E command mode, or end the session by exiting the shell.

Parameters

**ALARM | NOALARM**

Controls the sounding of the 3270 alarm to alert you to particular events. The default is ALARM.

**ALARM**

Causes the 3270 alarm to sound when the <alert> character is encountered in data being sent to the workstation.

**NOALARM**

Prevents the 3270 alarm from sounding when the <alert> character is encountered in data being sent to the workstation.

**AUTOSCROLL | NOAUTOSCROLL**

Controls the setting of the autoscroll function. The default is AUTOSCROLL.

**AUTOSCROLL**

Specifies automatic scrolling of input and output written to the screen.

**NOAUTOSCROLL**

Specifies that there not be automatic scrolling.

**CONVERT(character_conversion_table)**

Specifies the character conversion table that is used to convert between the z/OS code page and the code page that is used in the shell.

**data_set_name(member_name)**

Specifies the name of the partitioned data set (PDS) and the name of the member that contains the character conversion table.

**data_set_name**

Specifies the name of the partitioned data set containing the character conversion table to be used.

**(member_name)**

Specifies the name of the character conversion table to be used. It is the name of a member in a partitioned data set.

If both the member_name and data_set_name are omitted, member FSUMQ000 in the default module search order is used as the character conversion table. Table 38 on page 910 lists the various formats of the OMVS CONVERT command:

<table>
<thead>
<tr>
<th>Command format</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMVS CONV((BPXFX111))</td>
<td>See Note 1.</td>
</tr>
<tr>
<td>OMVS CONV('SYS1.XXXX')</td>
<td>Looks for SYS1.XXXX(FSUMQ000). FSUMQ000 is an alias; when shipped by IBM, it points to BPXFX100, the default null character conversion table.</td>
</tr>
<tr>
<td>OMVS CONV('SYS1.XXXX(BPXFX111)')</td>
<td>Looks for SYS1.XXXX(BPXFX111)</td>
</tr>
<tr>
<td>OMVS CONV('prefix.XXXX')</td>
<td>Looks for prefix.XXXX(FSUMQ000)</td>
</tr>
<tr>
<td>OMVS CONV('prefix.XXXX(BPXFX111)')</td>
<td>Looks for prefix.XXXX(BPXFX111)</td>
</tr>
</tbody>
</table>

**Note:**
1. If the `data_set_name` is omitted, z/OS UNIX locates `member_name` using the default search order for modules in the system library concatenation. The located `member_name` is used as the character conversion table. For example, if you specify:

```omvs
OMVS CONVERT((BPXFX111))
```

the character conversion table is `BPXFX111` in the default module search order in the system library concatenation.

If the member name is omitted, the OMVS command looks in the specified `data_set_name` for member `FSUMQ000`, to use it as the character conversion table. For example, if you specify:

```omvs
OMVS CONVERT('SYS1.XLATE') ...
```

OMVS uses `SYS1.XLATE(FSUMQ000)` as the character conversion table.

Table 39 on page 911 lists the character conversion tables supplied with the OMVS command. It shows the locale name, the conversion table to specify, and the default escape character for that table. If you are using the `De_CH.IBM-500` locale, you must specify `BPXFX450` as the conversion table, and the default escape character for that particular table is the topic sign, §. To specify `BPXFX450` as the conversion table, issue:

```omvs
CONVERT((BPXFX450))
```

<table>
<thead>
<tr>
<th>Shell and Utilities locale</th>
<th>3270 code page</th>
<th>Shell code page</th>
<th>Conversion table</th>
<th>Default escape character</th>
</tr>
</thead>
<tbody>
<tr>
<td>De_CH.IBM-500</td>
<td>IBM-500</td>
<td>IBM-500</td>
<td>BPXFX450</td>
<td>§</td>
</tr>
<tr>
<td>De_DE.IBM-273</td>
<td>IBM-273</td>
<td>IBM-273</td>
<td>BPXFX473</td>
<td>§</td>
</tr>
<tr>
<td>De_DK.IBM-277</td>
<td>IBM-277</td>
<td>IBM-277</td>
<td>BPXFX477</td>
<td>§</td>
</tr>
<tr>
<td>En_GB.IBM.285</td>
<td>IBM-285</td>
<td>IBM-285</td>
<td>BPXFX485</td>
<td>(the overline character)</td>
</tr>
<tr>
<td>En_JP.IBM-1027</td>
<td>IBM-1047</td>
<td>IBM-1047</td>
<td>BPXFX100</td>
<td>¢</td>
</tr>
<tr>
<td>En_US.IBM-037</td>
<td>IBM-037</td>
<td>IBM-037</td>
<td>BPXFX437</td>
<td>¢</td>
</tr>
<tr>
<td>En_US.IBM-1047(Fox APL terminals)</td>
<td>IBM-037</td>
<td>IBM-1047</td>
<td>BPXFX211</td>
<td>¢</td>
</tr>
<tr>
<td>En_US.IBM-1047</td>
<td>IBM-037</td>
<td>IBM-047</td>
<td>BPXFX111</td>
<td>¢</td>
</tr>
<tr>
<td>Es_ES.IBM.284</td>
<td>IBM-284</td>
<td>IBM-284</td>
<td>BPXFX484</td>
<td>&quot;</td>
</tr>
<tr>
<td>Fi_FI.IBM-278</td>
<td>IBM-278</td>
<td>IBM-278</td>
<td>BPXFX478</td>
<td>§</td>
</tr>
<tr>
<td>Fr_BE.IBM-500</td>
<td>IBM-500</td>
<td>IBM-500</td>
<td>BPXFX450</td>
<td>§</td>
</tr>
<tr>
<td>Fr_CA.IBM-037</td>
<td>IBM-037</td>
<td>IBM-037</td>
<td>BPXFX437</td>
<td>¢</td>
</tr>
<tr>
<td>Fr_CA.IBM-1047</td>
<td>IBM-037</td>
<td>IBM-1047</td>
<td>BPXFX111</td>
<td>¢</td>
</tr>
<tr>
<td>Fr_CH.IBM-500</td>
<td>IBM-500</td>
<td>IBM-500</td>
<td>BPXFX450</td>
<td>§</td>
</tr>
<tr>
<td>Fr_FR.IBM-297</td>
<td>IBM-297</td>
<td>IBM-297</td>
<td>BPXFX497</td>
<td>§</td>
</tr>
<tr>
<td>Is_IS.IBM-871</td>
<td>IBM-871</td>
<td>IBM-871</td>
<td>BPXFX471</td>
<td>&quot;</td>
</tr>
<tr>
<td>It_IT.IBM-280</td>
<td>IBM-280</td>
<td>IBM-280</td>
<td>BPXFX480</td>
<td>§</td>
</tr>
<tr>
<td>Ja_JP.IBM-939</td>
<td>IBM-939</td>
<td>IBM-939</td>
<td>BPXFX100</td>
<td>¢</td>
</tr>
<tr>
<td>Ja_JP.IBM-1027</td>
<td>IBM-1027</td>
<td>IBM-1027</td>
<td>BPXFX100</td>
<td>¢</td>
</tr>
</tbody>
</table>
Table 39: Locales, their conversion tables, and default escape characters (OMVS command) (continued)

<table>
<thead>
<tr>
<th>Shell and Utilities locale</th>
<th>3270 code page</th>
<th>Shell code page</th>
<th>Conversion table</th>
<th>Default escape character</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL_BE.IBM-500</td>
<td>IBM-500</td>
<td>IBM-500</td>
<td>BPXFX450</td>
<td>§</td>
</tr>
<tr>
<td>NL_NL.IBM-037</td>
<td>IBM-037</td>
<td>IBM-037</td>
<td>IBM-037</td>
<td>¥</td>
</tr>
<tr>
<td>NL_NL.IBM-1047</td>
<td>IBM-037</td>
<td>IBM-1047</td>
<td>BPXFX111</td>
<td>¥</td>
</tr>
<tr>
<td>No_NO.IBM-277</td>
<td>IBM-277</td>
<td>IBM-277</td>
<td>BPXFX477</td>
<td>¥</td>
</tr>
<tr>
<td>Pt_PT.IBM-037</td>
<td>IBM-037</td>
<td>IBM-037</td>
<td>BPXFX437</td>
<td>¥</td>
</tr>
<tr>
<td>Pt_PT.IBM-1047</td>
<td>IBM-037</td>
<td>IBM-1047</td>
<td>BPXFX111</td>
<td>¥</td>
</tr>
<tr>
<td>Sv_SE.IBM-278</td>
<td>IBM-278</td>
<td>IBM-278</td>
<td>BPXFX478</td>
<td>§</td>
</tr>
<tr>
<td>Xx_XX.IBM-1047</td>
<td>IBM-1047</td>
<td>IBM-1047</td>
<td>BPXFX100</td>
<td>¥</td>
</tr>
</tbody>
</table>

**DBCS | NODBCS**

Specifies whether to use DBCS on 3270-type terminals. The default is DBCS processing.

**DBCS**

Causes OMVS to automatically determine whether the terminal supports DBCS. If so, DBCS processing takes place. It also enables the OMVS command to handle double-byte data in translated messages. This operand is ignored if you're not using a DBCS terminal.

Double-byte data, including escape character strings, cannot be supplied for any of the OMVS command operands. The following data strings used by OMVS must contain single-byte characters only:

- Escape characters
- Conversion table data set name
- Conversion table member name
- Password or password phrase used to access the conversion table, if one is required

OMVS supports only code pages 939, 1027, and 1047 on DBCS. The null character conversion table (BPXFX100) should be used with DBCS terminals. (It is the default.)

**NODBCS**

Specifies that OMVS operate in SBCS mode only. If you are logged on to a terminal that supports DBCS, this operand allows you to bypass DBCS processing.

**DEBUG(NO | YES | EVENT | DATA)**

Controls the collection and output of debugging information. The default is NO; change the default setting only if IBM requests it.

**NO**

Indicates that no debugging information is to be written.

**YES**

Indicates that debugging information is collected while the OMVS command runs.

**EVENT**

Causes additional debugging information to be written whenever certain internal events occur in the OMVS command.

**DATA**

Causes any data received from or sent to the workstation to be written. Also, debug information for internal events is recorded.

Also, the ddname for the OMVS debug data set is always SYSFSUMO.
**ECHO | NOECHO**  
Enables OMVS to control the visibility of the input area. The default is NOECHO.

**ECHO**  
Allows OMVS to hide or unhide the input area.

**NOECHO**  
Prevents OMVS from hiding and unhiding the input area.

**ENDPASSTHROUGH(ATTN | CLEAR | CLEARPARTITION | ENTER | NO | PA1 | PA3 | PF1 | PF2 | PF3 ... PF24 | SEL)**  
Specifies a 3270 key that ends TSO/3270 passthrough mode and forces OMVS to return to the shell session. Because this key would be used only during application development, the default is ENDPASSTHROUGH(NO). All 3270 keys can be used by the 3270 application.

**ATTN**  
Specifies the 3270 <Attention> key. In some 3270 applications, this key may be changed to <PA1> before it is seen by the TSO/E OMVS command. If so, OMVS will never see the <Attention> key; specify <PA1> instead of <ATTN>.

With some terminal connections, the <ATTN> key may not be available.

**CLEAR**  
Specifies the 3270 CLEAR key. In some TSO/3270 applications, the TSO/E OMVS command will not see <CLEAR> when the CLEAR key is pressed. In these cases, specifying ENDPASSTHROUGH(CLEAR) will have no effect.

**CLEARPARTITION**  
Specifies the 3270 <Clear Partition> key. This key is effective only if the application is using explicit 3270 partitions.

**ENTER**  
Specifies the 3270 ENTER key. This key is useful only if the 3270 application is completely driven by PF or PA keys.

**NO**  
No breakout key; this is the default.

**PA1**  
Specifies the 3270 <PA1> key. For some TSO/3270 applications, <PA1> is changed to <ATTN> before OMVS sees it. In these cases, you should specify ENDPASSTHROUGH(ATTN).

In general, the provider of the TSO/3270 application needs to tell the user whether <PA1>, <ATTN>, or <CLEAR> can be used for ENDPASSTHROUGH.

**PA3**  
Specifies the 3270 <PA3> key. The <PA3> key may not be available on some keyboards.

**PFN**  
Specifies the 3270 function keys 1-9.

**PFNn**  
Specifies the 3270 function keys 10-24.

**SEL**  
Specifies the 3270 Cursor Select key. This key is useful only when the 3270 application creates fields on the 3270 screen that can be selected by a light pen.

**ESCAPE('escape-characters')**  
Specifies an escape character as the first character in a two-character sequence that is the EBCDIC equivalent of an ASCII control character (for example, the EBCDIC “ød” is the equivalent of the ASCII “Ctrl-D”). When an escape character is typed in the input area, the next character typed is converted into a special character before it is passed to the shell.

You can enter a string up to eight escape characters, enclosed in single quotes with no space between them. (Do not use nonprintable EBCDIC characters.)
The default escape character depends on the character conversion table being used. (See Table 39 on page 911 for a list of default characters and the conversion tables they are used with.) To enter <Ctrl-D>, for example, type in ød or øD in the input area.

If the last character in the input area is one of the escape characters, the <newline> character normally appended to the input data is suppressed. For example, to enter only a <Ctrl-Q> with no final <newline>, type the string øQø in the input area, and press <Enter>.

LINES(n)
Controls the amount of output data the OMVS command keeps for scrolling. The default is roughly four screenfuls. You can specify that between 25 and 3000 lines should be kept in the output buffer.

PFn(ALARM | NOALARM | AUTOSCROLL | NOAUTOSCROLL | BACKSCR | BOTTOM | CLOSE | CONTROL | ECHO | NOECHO | FWDRETR | HALFSR | HELP | HIDE | NOHIDE | NEXTSESS | NO | OPEN | PFSHOW | NOPFSHOW | PREVSESS | QUIT | QUITALL | REFRESH | RETRIEVE | RETURN | SCROLL | SUBCOMMAND | TOP | TSO)
Customizes the settings for the function keys that you use while working in the z/OS shell or in subcommand mode. In <PFN> The n is a one- or two-digit function key number from 1 to 24. Do not use a leading zero for a one-digit number. More than one function key can be assigned the same function. For example, both <PF1> and <PF13> are assigned the Help function by default.

All PF keys can be abbreviated using the usual TSO/E rules. For example,

• OPEN can be abbreviated as O, OP, or OPE.
• NEXTSESS can be abbreviated as NE, NEX, NEXT, NEXTS, NEXTSE, or NEXTSES.
• PFSHOW can be abbreviated as PF, and NOPFSHOW can be abbreviated as NOPF.

ALARM | NOALARM
A toggle key used to turn on and off the 3270 alarm that sounds when an <alert> character is written to the output area (also available in subcommand mode).

The label for this PF key (in the PF key lines at the bottom of the screen) shows up as either ALARM or NOALARM, depending on the current toggle setting. If it is ALARM, pressing this PF key turns the alarm on. If it is NOALARM, pressing this PF key turns the alarm off.

AUTOSCROLL | NOAUTOSCROLL
A toggle key used to turn the autoscroll function on and off (also available in subcommand mode). The screen automatically scrolls forward when new input is written to the screen.

The label for this PF key (in the PF key lines at the bottom of the screen) shows up as either AUTOSCROLL or NOAUTOSCROLL, depending on the current toggle setting. If it is AUTOSCROLL, pressing this PF key turns the autoscroll function on. If it is NOAUTOSCROLL, pressing this PF key turns the autoscroll function off.

BACKSCR
Scrolls the screen backward one full screen, redisplaying previously displayed output lines. The scrolling ends when the oldest available saved line is reached. (This option is also available in subcommand mode.)

If you first move the cursor into the output area, the line with the cursor becomes the top line.

BOTTOM
Scrolls help information forward to the last panel of information, and scrolls output forward the last full screen (also available in subcommand mode).

CLOSE
Ends the displayed session and switches to another one, or returns to TSO/E if the only session was closed (also available in subcommand mode).

CONTROL
Treats all characters in the input area as if they were preceded by an escape character. Also, no trailing <newline> is appended to the data.
**ECHO | NOECHO**
A toggle key used to control whether the shell command can hide or unhide the OMVS command input area.

The label for this PF key (in the PF key lines at the bottom of the screen) shows up as either ECHO or NOECHO, depending on the current toggle setting. If it is ECHO, pressing this PF key allows the current shell command to hide or unhide the OMVS command input area. If it is NOECHO, pressing this PF key prevents the current shell command from hiding or unhideing the OMVS input area.

**FWDRETR**
Retrieves the oldest available input line from a stack of saved input lines, starting with the oldest and moving up to the most recent line (also available in subcommand mode).

**HALFSCR**
Scrolls half the displayed screen forward, allowing room for more output data. If the output area on the screen is not full, half the displayed lines are scrolled off the screen. If you first move the cursor into the output area, the line with the cursor becomes the middle line. (This option is also available in subcommand mode.)

**HELP**
Temporarily suspends the session and displays the help information for the OMVS command. The scrolling function keys can be used to look at the help information. To exit the help information, press the Return function key. (This option is also available in subcommand mode.)

**HIDE | NOHIDE**
Temporarily hides or unhides the input data you type on the shell command line. If you press this PF key while the input area is hidden, the input area is made visible. If it is not hidden, the input area is hidden.

The input area stays hidden or unhidden until:

- You press <Enter>.
- You press the HIDE | NOHIDE PF key.
- You switch to another session, escape to TSO/E and return, or enter subcommand mode and return.

If OMVS is running in NOECHO mode, the input area will be visible after you take one of these actions. If OMVS is running in ECHO mode, the visibility of the input area depends on the shell command you are running.

**NEXTSESS**
Switches to the next (higher-numbered) session (also available in subcommand mode).

**NO**
Deactivates a function key so that it doesn’t do anything (also available in subcommand mode).

**OPEN**
Starts a new shell session and switches to it (also available in subcommand mode).

**PFSHOW | NOPFSHOW**
Toggles on and off the display of the active function key settings at the bottom of the screen (also available in subcommand mode, and can be used as PF and NOPF).

**PREVSESS**
Switches to the previous (lower-numbered) session (also available in subcommand mode).

**QUIT**
Ends the displayed session and switches to another one, or returns to TSO/E if the only session was closed (also available in subcommand mode).

**QUITALL**
Ends all shell sessions and causes OMVS to end and to return to TSO/E (also available in subcommand mode).

**REFRESH**
Updates the screen with the latest output data. Use this function key if the display of output is incomplete, but the session is now displaying INPUT status. For more information about the status
field, see *z/OS UNIX System Services User's Guide*. (This option is also available in subcommand mode.)

**RETRIEVE**
Retrieves the most recently entered input line from a stack of saved input lines, starting with the most recent and moving down to the oldest available line (also available in subcommand mode).

**RETURN**
If help information is displayed, returns you to the session you were in. If you are in subcommand mode, returns you to the shell. (This option is also available in subcommand mode.)

**SCROLL**
Scrolls the last line of output data to the top of the screen, making room for more output data. If Help information is displayed, its data is scrolled. If you first move the cursor into the output area, the line with the cursor becomes the top line. (This option is also available in subcommand mode.)

**SUBCOMMAND**
If you press this key when the command line is blank, it leaves the shell session and enters subcommand mode.

To run a subcommand without switching to subcommand mode, type the subcommand at the command line and then press the function key. You can enter the OMVS subcommands at the command line when you are in subcommand mode.

**TOP**
Scrolls help information backward to the first panel, and scrolls output backward to a screen full of the oldest available output (also available in subcommand mode).

**TSO**
If you press this key when the command line is blank, it temporarily suspends a shell session or subcommand mode, and you are in a TSO/E session. You can enter TSO/E commands. Press <PA1> or the <Attention> key to exit TSO/E command mode and return to the session you were in. (This option is also available in subcommand mode.)

To run a TSO/E command without suspending the shell session or subcommand mode, type the command at the command line and then press the function key. When the command completes, you can continue working in the shell session or subcommand mode.

Function key defaults:

PF1(HELP)
PF2(SUBCOMMAND)
PF3(RETURN)
PF4(TOP)
PF5(BOTTOM)
PF6(TSO)
PF7(BACKSCR)
PF8(SCROLL)
PF9(NEXTSESS)
PF10(REFRESH)
PF11(FWDRETR)
PF12(RETRIEVE)
PF13(HELP)
PF14(SUBCOMMAND)
PF15(RETURN)
PF16(TOP)
PF17(BOTTOM)
PF18(TSO)
PF19(BACKSCR)
PF20(SCROLL)
PF21(NEXTSESS)
PF22(REFRESH)
PF23(FWDRETR)
PF24(RETRIEVE)

PF SHOW | NOPF SHOW
Specifies that the PF keys be shown at the bottom of the screen. The default is PF SHOW.

PF SHOW
Specifies that PF keys be shown at the bottom of the screen.

NOPF SHOW
Specifies that PF keys not be shown at the bottom of the screen.

RUNOPTS('run-time-options')
Specifies a string containing run-time options, which are passed to Language Environment when the TSO/E OMVS command starts up, and to the initial login shell program in the _CEE_RUNOPTS environment variable. These options are the same as those passed to other Language Environment programs run from the TSO READY prompt.

The options string can be from 1 to 1000 characters in length, and should contain valid run-time options. It should not contain options such as POSIX(OFF), TRAP(OFF), TRAP(ON,NOSPIE), or MSGFILE(), or characters such as slashes, unbalanced parentheses or quotes, or imbedded NULL characters. Specifying such options or using these characters will cause unpredictable problems when the TSO/E OMVS command runs.

If the RUNOPTS operand is omitted, OMVS uses the RUNOPTS string defined in the BPXPRMxx member of SYS1.PARMLIB that is active for the OMVS kernel. If no RUNOPTS string was defined in BPXPRMxx, no default run-time options are used when the TSO/E OMVS command starts up.

For more information, refer to z/OS Language Environment Programming Guide which contains a discussion about restrictions on _CEE_RUNOPTS environment variable settings.

SESSIONS(n)
Specifies the initial number of sessions to be started. The default is 1, and the allowed range is 1 to 100; most users will use two or three sessions.

Note: You can specify a number from 1 to 100 without getting a syntax error on the command. Normally, you cannot start more than several sessions before getting an error message. If you try to start too many sessions (the limit depends on the size of your TSO/E address space), your TSO/E user ID runs out of storage and various unpredictable errors may occur. You may have to log off your TSO/E user ID before you can continue.

SHAREAS | NOSHAREAS
Specifies whether to run the shell program in a separate address space. Both OMVS and the shell will run in the TSO/E address space when OMVS is invoked with the SHAREAS parameter.

OMVS will use SHAREAS as the default if the shell program is not a SETUID or SETGID program and the owning UID or GID is not the same as the current user.

SHAREAS
Runs the shell program in the same TSO/E address space as OMVS. SETUID and SETGID shell programs cannot be run with the SHAREAS option unless your UID or GID owns the shell program.

Note: If you end OMVS while in SHAREAS mode, the shell process ends immediately. (It may get killed, but it will usually end by itself when the TTY is closed.)

NOSHAREAS
Runs the shell program in a separate address space. SETUID and SETGID shell programs usually require this option.

WRAPDEBUG(n)
Controls how many lines of debug data OMVS writes out before wrapping around to the top of the debug data set. This option is effective only if the DEBUG(YES) DEBUG(EVENT), or DEBUG(DATA) options are used.

The WRAPDEBUG(n) value specifies how many lines of debug data OMVS writes out before wrapping around to the top of the debug data set. The default number of lines is 10 000. The value of n must be
between 100 and 1,000,000,000. The debug data set must be large enough to hold n 80-byte lines of debug data. If the debug data set is too small, debug recording stops when the data set fills up.

Subcommands

When the shell is active, you can enter subcommand mode by pressing the Subcommand function key. While in subcommand mode, you can enter subcommands on the command line or use function keys.

**ALARM**
Turns on the 3270 alarm which sounds when an <alert> character is written to the output area.

**AUTOSCROLL**
Activates automatic forward scrolling of output as new input is written to the screen.

**BACKSCR**
Scrolls the screen backward one full screen, redisplaying previously deleted output lines. The scrolling ends when the oldest available saved line is reached.

**BOTTOM**
If the help information is displayed, it is scrolled forward to the last panel of information. If output is displayed, it is scrolled forward to the last screen of output.

**CLOSE**
Ends the displayed session and switches to another one, or returns to TSO/E if the only session was closed.

**ECHO**
 Allows the current shell command to control whether the OMVS input area is visible or hidden. The HIDE subcommand, NOHIDE subcommand, and HIDE | NOHIDE PF keys can temporarily override the input area visibility set by the current shell command.

**HALFSCR**
Scrolls half the displayed screen forward, allowing room for more output data.

**HELP**
Displays help information for the OMVS command. To view the help information, use the scrolling function keys. To return from Help to the session, press the Return function key.

? is a short form for the Help subcommand.

**HIDE**
Temporarily hides the input data you type on the shell command line. The input area stays hidden until you do one of the following actions:

- Press <Enter>.
- Press the HIDE | NOHIDE PF key.
- Switch to another session, escape to TSO and return, or enter subcommand mode and return.

If OMVS is running in NOECHO mode, the input area will be visible after you take one of these actions. If it is running in ECHO mode, whether you can see the input area depends on the shell command you are running.

**NEXTSESS**
Switches to the next (higher-numbered) session.

**NOALARM**
Prevents the 3270 alarm from sounding when the <alert> character is encountered in data being sent to the workstation.

**NOAUTOSCROLL**
Turns off the automatic scrolling (AUTOSCROLL) function.

**NOECHO**
Causes the OMVS input area to remain visible regardless of the current shell command. You can use the HIDE subcommand and the PF key to temporarily hide the input area.
**NOHIDE**
Temporarily unhides the input data you type on the shell command line. The input area remains visible until you do one of the following actions:

- Press <Enter>.
- Press the HIDE | NOHIDE PF key.
- Switch to another session, escape to TSO and return, or enter subcommand mode and return.

If OMVS is running in NOECHO mode, the input area remains visible after you take one of these actions. If OMVS is running in ECHO mode, the visibility of the input area depends on the shell command you are running.

**NOPFSHOW**
Turns off the display of the function key settings and escape characters at the bottom of the screen.

**OPEN**
Starts a new shell session and switches to it.

**PFSHOW**
Displays the current function key settings and escape characters on the bottom two lines of the display screen. A maximum of two screen lines is used. If some function key settings do not fit on the two lines, they are not displayed.

**PREVSESS**
Switches to the previous (lower-numbered) session.

**QUIT**
Ends the displayed session and switches to another one, or returns to TSO/E if the only session was closed.

**QUITALL**
Ends all shell sessions and causes OMVS to end and to return to TSO/E.

**RETURN**
Returns from subcommand mode to the shell session. If help information is being displayed, the session returns to subcommand mode and you must enter the RETURN command again to return to the shell.

**SCROLL**
Scrolls forward the data displayed on the screen, approximately one full screen.

**TOP**
Scrolls help information backward to the first panel. Scrolls output backward to a display of the oldest available output.

**TSO**
Invokes TSO/E command mode. In this mode, you can enter TSO/E commands. Press <PA1> or the <Attention> key to return to subcommand mode.

**Usage notes**
1. The OMVS command is a Language Environment application. OMVS overrides the default MSGFILE ddname (SYSOUT) and uses ddname SYSFSUMM.

   Normally, any Language Environment error messages from the OMVS command are displayed on the TSO/E terminal. If you want to redirect these messages, you need to allocate the SYSFSUMM ddname instead of the SYSOUT ddname, as is usual with Language Environment applications.

2. The language of the OMVS command messages is determined by the PROFILE PLANGUAGE setting when OMVS is invoked. Do not change PROFILE PLANGUAGE while OMVS is running.

**Return codes**
0
Processing successful.
Processing unsuccessful. An error message has been issued.

Examples

These examples explain how to use the multi-session capability of OMVS:

1. To start 2 sessions automatically when starting OMVS, enter:

   ```
   OMVS SESSIONS(2)
   ```

2. To assign the NEXTSESS function to a PF key, enter:

   ```
   OMVS PF1(NEXTSESS)
   ```

OPUT - Copy an MVS data set member into a z/OS UNIX file

Format

```
OPUT  mvs_data_set_name | mvs_data_set_name(member_name)
   pathname
   BINARY | TEXT
   CONVERT(character_conversion_table | YES | NO)
```

Description

You can use the OPUT command to:

- Copy a member of an MVS partitioned data set (PDS or PDSE) to a file
- Copy an MVS sequential data set to a file

and convert the data from code page IBM-037 or ASCII to code page IBM-1047.

Do not use the CONVERT option when copying files that contain double-byte data. This option is used for single-byte data only, not for double-byte data.

Parameters

**mvs_data_set_name | mvs_data_set_name(member_name)**

Specifies the name of an MVS sequential data set or an MVS partitioned data set member that is being copied.

- A fully qualified name that is enclosed in single quotation marks, or an unqualified name (an unqualified name is not enclosed in single quotation marks)
- Up to 44 characters long, with an additional 8 characters for the member name
- Converted to uppercase characters by the system

**pathname**

Specifies the path name of the file to receive the data set member that is being copied. The target file cannot be a directory. All directories in the path name prior to the file name directory must already exist. The path name is:

- A relative or absolute path name. A relative path name is relative to the working directory of the TSO/E session (usually the HOME directory). Therefore, you should usually specify an absolute path name.
- Up to 1023 characters long.
- Enclosed in single quotation marks.
- In uppercase or lowercase characters, which are not changed by the system.
**BINARY | TEXT**

specifies that the data set being copied contains binary data or text.

**BINARY**

Specifies that the data set being copied contains binary data. This is the default for a data set of undefined record format.

**TEXT**

Specifies that the data set being copied contains text. This is the default for a data set of fixed record format or variable record format.

**CONVERT(character_conversion_table | YES | NO)**

Specifies that the data being copied is to be converted from IBM-037 or ASCII to EBCDIC Latin 1/Open Systems Interconnection code page 01047—that is, that the TO1047 part of the specified character conversion table will be used. This operand is optional. If this operand is omitted, the system copies the data without conversion.

You can use this option for single-byte data, but not for double-byte data.

Specify the CONVERT value as one of the following:

**character_conversion_table**

Specify one of the following:

- data_set_name(member_name). Specifies the name of the partitioned data set (PDS) and the name of the member that contains the character conversion table.
- data_set_name. Specifies the name of the partitioned data set (PDS) that contains the character conversion table. The table is the FROM1047 part in member BPXFX000. (This is an alias; when shipped by IBM, it points to BPXFX111.)
- (member_name). Specifies the name of the conversion table to be used. It is a member of a PDS. Because the data_set_name is omitted, the standard library concatenation is searched for the table. (The default library is SYS1.LINKLIB.)

The following list summarizes what you can specify when you want to convert data to a different code page when copying single-byte data:

- BPXFX100. Null character conversion table. Use this table if the square brackets at your workstation are at the same code points as the square brackets on code page 1047 (it is the default). Also use it if you are using a DBCS terminal.
- BPXFX111. Specifies a non-APL conversion table to convert between code pages IBM-037 and IBM-1047.
- BPXFX211. Specifies an APL conversion table to convert between code pages IBM-037 and IBM-1047.
- BPXFX311. Specifies an ASCII-EBCDIC conversion table to convert between code pages ISO8859-1 and IBM-1047.

**YES**

The system will perform conversion and use the default conversion table (BPXFX000) in the system library concatenation. (BPXFX000 is an alias; when shipped by IBM, it points to BPXFX111.)

**NO**

Specifies no conversion. NO is the same as omitting the CONVERT operand.

**Usage notes**

1. If the specified file does not exist, OPUT creates a new file. For a new text file, the mode (permission bits) is 600. When the mode is 600, the user has read and write access; others have nothing. For a new binary file, the mode (permission bits) is 700. When the mode is 700, the user has read, write, and execute access; others have nothing.
2. If the specified file exists, the new data overwrites the existing data. The mode of the file is unchanged.
3. You can use OPUT to copy a program object from a PDSE to the file system, and it will be executable there. If you have a load module in a partitioned data set, however, you must first use the IEBCOPY program to copy the load module from a partitioned data set to a PDSE and then subsequently use OPUT to copy the module into the file system. IEBCOPY converts load modules to a program object. See z/OS UNIX System Services User's Guide for a discussion of copying executable files.

4. Data sets with spanned record lengths are not allowed.

5. When you copy MVS data sets to text files in the z/OS UNIX file system, a \n character is appended to the end of each record. If trailing blanks exist in the record, the \n character is appended after the trailing blanks. MVS fixed block data sets have a fixed record length, which means that trailing blanks could exist up to the end of each record.

6. When you copy MVS data sets to binary files in the z/OS UNIX file system, the \n character is not appended to the record.

Return codes

0
Processing successful.

12
Processing unsuccessful. An error message has been issued.

Examples

1. This command copies an MVS sequential data set to a file, converting from code page IBM-037 to code page 1047.
   • The unqualified name of the sequential data set is EMPLOYEE.DATA.
   • The path name of the file is /u/admin/employee/data.

   OPUT EMPLOYEE.DATA '/u/admin/employee/data' TEXT CONVERT(YES)

2. This command copies an MVS sequential data set to a file converting to code page 1047 using the conversion table BPXFX000 in the user's library data set.
   • The fully qualified name of the sequential data set is IBMUSR.EMPLOYEE.DATA.
   • The path name of the file is /u/admin/employee/data.

   OPUT 'IBMUSR.EMPLOYEE.DATA' '/u/admin/employee/data'
   TEXT CONVERT(MY.LOADLIB(BPXFX000))

3. This command copies a binary file from a PDSE to a file in the file system.
   • APPL.LOADLIB(PAYROLL) is the fully qualified name of the member of the PDSE.
   • bin/payroll is the path name of the file; the directory bin is in the working directory.
   • There is no code page conversion.

   OPUT 'APPL.LOADLIB(PAYROLL)' '/bin/payroll' binary

OPUTX - Copy members from an MVS PDS or PDSE to a z/OS UNIX system directory

Format

OPUTX  mvs_PDS_name | mvs_data_set-name(member_name)
hfs_directory | hfs_file_name
ASIS
BINARY | TEXT
CONVERT(character_conversion_table | YES | NO)
Description
You can use the OPUTX command to:

• Copy members from an MVS partitioned data set (PDS) or PDSE to a directory in the z/OS UNIX file system.
• Copy a sequential data set or member of a PDS to a file.

and convert the data from code page IBM-037 or ASCII to code page IBM-1047 while it is being copied.

Restriction: Do not use the CONVERT option when copying files that contain double-byte data. This option is used for single-byte data only, not for double-byte data.

This command uses the ISPF/PDF Edit facility.

Parameters

hfs_directory | HFS_file_name
Specifies the directory name or file name of a file in the z/OS UNIX file system that is to receive the PDS members that are being copied. The name can be up to 1023 characters long. Single quotation marks around the directory name or file name are optional.

Use hfs_directory when a PDS is specified. When a sequential data set or PDS member is specified, then the file name must be used.

mvs_PDS_name | mvs_data_set_name(member_name)
Specifies the name of an MVS partitioned data set or an MVS partitioned data set member that is being copied into a z/OS UNIX file system. The data set name is:

• A fully qualified name that is enclosed in single quotation marks, or an unqualified name. An unqualified name is not enclosed in single quotation marks.
• Up to 44 characters long, with an additional 8 characters for the member name
• Converted to uppercase letters

ASIS
Specifies that the @ character in member names not be translated to the _ character in path names. (It is a common convention to use @ symbols in PDS member names to correspond with the _ symbol in path names.)

BINARY | TEXT
Specifies whether the data set being copied contains binary data or text.

BINARY
Specifies that the data set being copied contains binary data. BINARY is the default for a data set of undefined record format.

TEXT
Specifies that the data set being copied contains text. TEXT is the default for a data set of fixed record format or variable record format.

CONVERT(character_conversion_table | YES | NO)
Specifies that the data being copied be converted from code page IBM-037 to EBCDIC Latin 1/Open Systems Interconnection code page 01047. That is, the TO1047 part of the specified character conversion table is used. This operand is optional. If this operand is omitted, the system copies the data without conversion.

You can use this option for single-byte data, but not for double-byte data.

Specify the CONVERT value as one of the following:
**character_conversion_table**
Specify one of the following:

**data_set_name(member_name)**
Specifies the name of the partitioned data set (PDS) and the name of the member that contains the character conversion table.

**data_set_name**
Specifies the name of the partitioned data set (PDS) that contains the character conversion table. The table is the FROM1047 part in member BPXFX000. (This is an alias; when shipped by IBM, it points to BPXFX111.)

**(member_name)**
Specifies the name of the conversion table to be used. It is a member of a PDS. Because the data_set_name is omitted, the standard library concatenation is searched for the table. (The default library is SYS1.LINKLIB)

The following list summarizes what you can specify when you want to convert data to a different code page when copying single-byte data:

- **BPXFX100.** Null character conversion table. Use this table if the square brackets at your workstation are at the same code points as the square brackets on code page 1047 (it is the default). Also use it if you are using a DBCS terminal.
- **BPXFX111.** Specifies a non-APL conversion table to convert between code pages IBM-037 and IBM-1047.
- **BPXFX211.** Specifies an APL conversion table to convert between code pages IBM-037 and IBM-1047.
- **BPXFX311.** Specifies an ASCII-EBCDIC conversion table to convert between code pages ISO8859-1 and IBM-1047.

**NO**
Specifies that conversion not be done. NO is the same as omitting the CONVERT operand.

**YES**
Specifies that the system is to perform conversion and use the default conversion table (BPXFX000) in the standard library concatenation. (BPXFX000 is an alias; when shipped by IBM, it points to BPXFX111.)

**LC**
Specifies that the member name be converted to a lowercase file name.

**MODE(nnn)**
Specifies the file mode for any members that are copied into the z/OS UNIX file system.

*nnn*
Specified as three or four octal digits. The digits can be separated by commas. Invalid mode specifications are ignored.

If the specified file does not exist and the mode is not specified, OPUTX creates a new file. For a new text file, the mode (permission bits) is 600. When the mode is 600, the user has read and write access; others have none. For a new binary file, the mode (permission bits) is 700. When the mode is 700, the user has read, write, and search access; others have none.

**QUIET**
Turns off the echoing of the OPUTX command before the member or data set is copied.

**SUFFIX(suffix)**
Specifies that a suffix specified by (suffix) be appended to the member names in creating the file names for the z/OS UNIX system.

A suffix is an optional additional file identifier that is appended to the file name following a period (.). It is typically used to identify the type of file. For example, .c typically indicates a C language source file and .h indicates a C language header file. Suffixes can be any length and you can append as many as you want, but the file name, including suffixes, cannot exceed 255 characters for z/OS UNIX.
Usage notes

1. Do not use OPUTX with path names that contains quotation marks or spaces.
2. If the specified file does not exist, OPUTX creates a new file. For a new text file, the mode (permission bits) is 600. When the mode is 600, the user has read and write access; others have nothing. For a new binary file, the mode (permission bits) is 700. When the mode is 700, the user has read, write, and search access; others have nothing.
3. If the specified file exists, the new data overwrites the existing data. The mode of the file is unchanged.
4. Data sets with spanned records are not allowed.
5. When you copy MVS data sets to text files in the z/OS UNIX file system, a <newline> character is appended to the end of each record. If trailing blanks exist in the record, the <newline> character is appended after the trailing blanks. MVS fixed block data sets have a fixed record length, which means that trailing blanks could exist up to the end of each record.
6. When you copy MVS data sets to binary files in the z/OS UNIX file system, the <newline> character is not appended to the record.
7. Before the copy, the OPUTX command for a data set or member is echoed, unless you specify the QUIET option. If you did not specify QUIET and if the command is not displayed, the data set or member is not copied.
8. If the source data set is a PDS with an undefined record format, the members might be treated as load modules. A load module is copied by link-editing it into the target file in the file hierarchy. For the program to be able to run from the file hierarchy, the entry point must be at the beginning of the load module.
   For OPUTX to treat the file as a load module, neither BINARY or TEXT can be specified.
9. If the source for the copy is a sequential data set or a PDS member and the target is a directory, the file name that is used is the last qualifier of the data set name or the member name. You do not have to specify a file as the target with a sequential data set, or a directory as the target with a PDS. The LC and ASIS options are not affected.

Examples

The following command copies files in a PDS into a directory in the z/OS UNIX file system and specifies that:

- The name of the partitioned data set (PDS) is DATAFILE
- The directory is /usr/sbllib
- The files are given a suffix of .c

OPUTX DATAFILE '/usr/sbllib/' LC SUFFIX(c)

Assuming the PDS has members PROGRAM1, PROGRAM2, and PROGRAM3, these members are copied as /usr/sbllib/program1.c, /usr/sbllib/program2.c, and /usr/sbllib/program3.c.

OSHELL - Invokes BPXBATCH from TSO/E

Format

OSHELL
Description

OSHELL uses BPXBATCH to run the shell command or shell script:

```
oshell shell_command
```

For example, to display process information, enter:

```
oshell ps -ej
```

For more information about BPXBATCH, see BPXBATCH

When you use OSHELL, do not use an & to run a shell command in the background.

Some examples of using the OSHELL command are:

- List files in a directory
- Create, delete, or rename directories, files, and special files
- Display contents of a file
- Copy files
- Display file attributes
- Search files for text strings
- Compare files or directories
- Run executable files
- Display the attributes and contents of a symbolic link
- Set up character special files
- Set up standard directories for a root file system

Some of these tasks may require superuser authority.

OSTEPLIB - Build a list of files

Format

```
OSTEPLIB pathname
```

Description

Use the OSTEPLIB command to build a list of files that are sanctioned by your installation as valid step libraries for programs that have the set-user-ID or set-group-ID bit set. This permission setting allows a program to have temporary access to files that are not normally accessible to other users. Step libraries have many uses; for example, selected users can test new versions of runtime libraries before the new versions are made generally available.

You must have superuser authority to issue OSTEPLIB.

The sanctioned list is valid if it conforms to the following rules:

- You can include comment lines in the list. Each comment line must start with /* and end with */.
- You must follow standard MVS data set naming conventions in naming the files in the list.
- Each data set name must be fully qualified and cannot be enclosed in quotation marks.
- Each data set name must be on a line by itself, with no comments.
- You can put blanks before and after each data set name. Entirely blank lines in the list are ignored.
- You can use the * character to specify multiple files that begin with the same characters. For example, if you list SYS1.*, you are sanctioning any file that begins with SYS1. as a step library.
Following is an example of a file that contains a correctly formatted list of sanctioned step libraries:

```c
/******************************************************************************
/*                                                               */
/*Name: Sample Sanctioned List for set-user-ID and set-group-ID files */
/*                                                               */
/*Updated by:   May only be updated by OSTEPLIB TSO/E command */
/*                                                               */
/*Description:  Contains a list of data set names that may be used as STEPLIB libraries for SETUID */
/*              programs */
/*              Wild cards may be used to specify multiple data set names that have the same prefix characters. */
/*******************************************************************************/

Sanction all data set names beginning with SYS1.CEE

SYS1.CEE*

Sanction data set containing vers. 2 of the C run time library

ADMIN.CEE.RTLV2

Parameters

**pathname**

Specifies the path name of the file to contain the list of sanctioned step libraries. The path name can be absolute or relative to the root. Avoid using the space character or single quotation mark (apostrophe) within the path name. The path name cannot be enclosed in single quotes.

If you omit the path name operand, the new sanctioned list file is created with the same file name as the old one and replaces it when it has been validated.

**UNMOUNT - Remove a file system from the file hierarchy**

**Format**

```
UNMOUNT FILESYSTEM(file_system_name) DRAIN | FORCE | IMMEDIATE | NORMAL | REMOUNT(RDWR|READ|SAMEMODE) | RESET
```

**Description**

The UNMOUNT command removes a file system from the file system hierarchy. The alias for this command is U Mount.

Be aware of these restrictions when using the UNMOUNT command.

- A file system that has file systems mounted on it cannot be unmounted. Any child file systems must be unmounted first.
- A file system cannot be explicitly remounted in the mode that the file system is already mounted in.

You must have mount authority before you can issue the UNMOUNT command. See the section on mounting file systems in *z/OS UNIX System Services Planning*.

**Parameters**

**FILESYSTEM(file_system_name)**

Specifies the name of the file system to be removed from the file system. The name supplied is changed to all uppercase characters. This operand is required.
**file_system_name**

The fully qualified name of the data set that contains the file system. The file system name supplied is changed to all uppercase characters.

Specify the name of file system exactly as it was specified when the file system was originally mounted. You can enclose it in single quotes, but they are not required.

If FILESYSTEM(“file_system_name”) is specified, the file system name will not be translated to uppercase.

**DRAIN**

Specifies that an unmount drain request is to be made. The system will wait for all use of the file system to be ended normally before the unmount request is processed or until another UNMOUNT command is issued.

UNMOUNT can be specified with IMMEDIATE to override a previous UNMOUNT DRAIN request for a file system. If this is used in the foreground, your TSO/E session waits until the UNMOUNT request has completed. <ATTN> (or <PA1>) does not terminate the command.

UNMOUNT DRAIN is not supported in a sysplex environment. If an UNMOUNT DRAIN is issued in a sysplex, the following behavior is exhibited:

- If there is no activity in the file system, UNMOUNT DRAIN will perform the unmount, but it will behave like an UNMOUNT NORMAL.
- If there is activity in the file system, UNMOUNT DRAIN will return a Return_value of -1 with Return_code EINVAL and Reason_code JrNotSupInSysplex.

**FORCE**

Specifies that the system is to unmount the file system immediately. Any users accessing files in the specified file system receive failing return codes. All data changes to files in the specified file system are saved, if possible. If the data changes to the files cannot be saved, the unmount request continues and data is lost.

An UNMOUNT IMMEDIATE request must be issued before you can request a UNMOUNT FORCE of a file system. Otherwise, UNMOUNT FORCE fails.

**IMMEDIATE**

Specifies that the system is to unmount the file system immediately. Any users accessing files in the specified file system receive failing return codes. All data changes to files in the specified file system are saved. If the data changes to files cannot be saved, the unmount request fails.

**NORMAL**

Specifies that if no user is accessing any of the files in the specified file system, the system processes the unmount request. Otherwise, the system rejects the unmount request. NORMAL is the default.

**REMount(RDWR|READ|SAMEMODE)**

Specifies that the specified file system be remounted and its mount mode changed, if necessary. REMOUNT takes an optional argument of RDWR, READ, UNMOUNT, or SAMEMODE.

- If REMOUNT is specified without any arguments, the mount mode is changed from RDWR to READ, or READ to RDWR.
- If RDWR is specified and the current mode is READ, the file system is remounted in RDWR mode.
- If READ is specified and the current mode is RDWR, the file system is remounted in READ mode.
- If SAMEMODE is specified, the file system is remounted (internally unmounted and remounted) without changing the mount mode. You can use this option to attempt to regain use of a file system that had I/O errors.

REMount is supported in a sysplex.

If a problem occurs with the remount, determine the failure, correct the problem, and try the remount again. The file system might not be available until the problems are corrected.

**RESET**

A reset request stops a previous UNMOUNT DRAIN request.
UNMOUNT RESET is not supported in a sysplex because UNMOUNT DRAIN is not supported in a sysplex environment (see the description for DRAIN).

Usage notes for UNMOUNT

1. When the UNMOUNT command completes, the unmount is complete on the owner system, and on the system where the unmount command was issued from, but not necessarily on the other systems in the sysplex. On those systems, the unmount completes asynchronously.
2. The /samples directory contain sample UNMOUNT commands (called unmountx).
3. If you unmount a TFS file system, all data stored in that file system is discarded. For more information about TFS, see z/OS UNIX System Services Planning.
4. The root file system can be unmounted, but the IMMED operand must be specified. Because unmounting the root stops all file system activity, a subsequent mount of a root file system should be done as soon as possible.
5. While the root file system is unmounted, a dummy file system root named SYSROOT is displayed as the current root file system. During this time, any operation that requires a valid file system will fail. When the new root file system is subsequently mounted, you should terminate any currently dubbed users or issue a chdir using a full path name to the appropriate directory so that the users can access the new root file system.
6. If the file system that you are unmounting is an NFS-supported file system, the UNMOUNT command may receive an EAGAIN return code if the request was made before an internal caching clock has expired. That is, there is a 60-second delay from last use before termination is possible. Try the request again.
7. Currently a move of a file system that has open FIFOs causes all FIFOs to be marked stale. They must be closed and reopened. Rather than do this on a remount, a remount attempt of a file system with open FIFOs will be rejected with EINVAL, JrFIFOInFileSys. FIFOs break on move or remount because FIFOs are always function-shipped to the file system owner, regardless of the mount mode. For remount, although the owner does not change, the vfs_umounts on all systems in the sysplex result in the XPFS control blocks (XFS, Xnodes) being released. These blocks contain owner information. Rejecting remount if open FIFOs is not expected to impact customers, since remount is typically done on a read-only file system to switch it to RDWR, and then back to READ, and FIFOs are not useful in a read-only file system.

Return codes

0
Processing successful.

12
Processing unsuccessful. An error message has been issued.

Examples

1. The following command specifies a normal unmount by default:

   UNMOUNT FILESYSTEM('HFS.WORKDS')

2. Before you request a forced unmount of a file system, you must issue an immediate unmount request:

   UNMOUNT FILESYSTEM('HFS.WORKDS') IMMEDIATE
   UNMOUNT FILESYSTEM('HFS.WORKDS') FORCE

3. To unconditionally change the mount mode of a file system:

   UNMOUNT FILESYSTEM(HFS.OMVS.BIN) REMOUNT

TSO/E commands 929
4. To change the mount mode of a file system to read/write, provided it is currently read-only:

```
UNMOUNT FILESYSTEM(HFS.OMVS.BIN) REMOUNT(RDWR)
```

**ZLSOF - Displays information about open files, sockets, and pipes**

See “zls - Display information about open files, sockets, and pipes” on page 877.
Chapter 4. REXX system commands

This part describes the REXX system commands.

The REXX system commands for working with the file system are:

- `bpxmtext`
- `bpxwmigf`
- `zslof`

You can enter these commands through System REXX as an MVS system command. See the section Communication with System REXX in z/OS MVS System Commands. Some UNIX commands may require mixed case arguments or run longer than the System REXX time limit. Communication with System REXX describes using lower case and other quote rules. It also describes how to override the default time limit and how to use command prefixes. In order to use z/OS UNIX facilities, it is necessary to logon to the console with a user ID that has access to z/OS UNIX. Also, depending on the command, the use ID might also need to have superuser authority or be permitted to the BPX.SUPERUSER resource in the FACILITY class.

**bpxmtext: Display reason code text**

See “bpxmtext - Display reason code text” on page 67.

**bpxwmigf: Migrate HFS file systems to zFS**

See “bpxwmigf - Migrate HFS file systems to zFS” on page 74.

**zlslof: Display information about open files, sockets, and pipes**

See “zlslof - Display information about open files, sockets, and pipes” on page 877.
Appendix A. Summary of z/OS UNIX shell commands

The following list presents z/OS shell commands and utilities grouped by the task a user might want to perform. Similar tasks are organized together. Stub commands (cancel, cu and lpstat) are not listed because their functions are not supported by z/OS UNIX System Services.

The list also shows the command name, the standard or specification it satisfies, and its function. XPG4.2 refers to X/Open CAE Issue 4 Version 2 Specifications. XPG5.0 refers to X/Open CAE Issue 5 Specifications.

### General use

<table>
<thead>
<tr>
<th>Command</th>
<th>Standard/Specification</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>at</td>
<td>POSIX.2 XPG4.2</td>
<td>Run a command at a specified time.</td>
</tr>
<tr>
<td>batch</td>
<td>POSIX.2 XPG4.2</td>
<td>Run commands when the system is not busy.</td>
</tr>
<tr>
<td>bpxmtext</td>
<td>—</td>
<td>Display reason code text.</td>
</tr>
<tr>
<td>ceebltxt</td>
<td>—</td>
<td>Transform message source files into loadable message text files.</td>
</tr>
<tr>
<td>clear</td>
<td>—</td>
<td>Clear the screen of all previous output.</td>
</tr>
<tr>
<td>command</td>
<td>POSIX.2 XPG4.2</td>
<td>Run a simple command.</td>
</tr>
<tr>
<td>configfs</td>
<td>—</td>
<td>Invoke vfs_pfsctl functions for HFS file systems.</td>
</tr>
<tr>
<td>date</td>
<td>POSIX.2 XPG4.2</td>
<td>Display the date and time.</td>
</tr>
<tr>
<td>echo</td>
<td>POSIX.2 XPG4.2</td>
<td>Write arguments to standard output.</td>
</tr>
<tr>
<td>edcmtext</td>
<td>—</td>
<td>Display errnojr reason code text.</td>
</tr>
<tr>
<td>exec</td>
<td>POSIX.2 XPG4.2</td>
<td>Run a command and open, close, or copy the file descriptors.</td>
</tr>
<tr>
<td>man</td>
<td>POSIX.2 XPG4.2</td>
<td>Print sections of the online reference manual.</td>
</tr>
<tr>
<td>nice</td>
<td>POSIX.2 XPG4.2</td>
<td>Run a command at a different priority.</td>
</tr>
<tr>
<td>passwd</td>
<td>—</td>
<td>Change user passwords and password phrases.</td>
</tr>
<tr>
<td>print</td>
<td>—</td>
<td>Return arguments from the shell.</td>
</tr>
<tr>
<td>printf</td>
<td>POSIX.2 XPG4.2</td>
<td>Write formatted output.</td>
</tr>
<tr>
<td>sh</td>
<td>POSIX.2 XPG4.2</td>
<td>Invoke a shell (command interpreter).</td>
</tr>
<tr>
<td>tcsh</td>
<td>—</td>
<td>Invoke a tcsh shell.</td>
</tr>
<tr>
<td>time</td>
<td>POSIX.2 XPG4.2</td>
<td>Display processor and elapsed times for a command.</td>
</tr>
<tr>
<td>uptime</td>
<td>—</td>
<td>Report how long the system has been running.</td>
</tr>
<tr>
<td>wall</td>
<td>—</td>
<td>Broadcast a message to logged-in users.</td>
</tr>
<tr>
<td>whence</td>
<td>—</td>
<td>Tell how the shell interprets a command name.</td>
</tr>
<tr>
<td>whoami</td>
<td>—</td>
<td>Display your effective user name.</td>
</tr>
<tr>
<td>xlc</td>
<td>—</td>
<td>Compiler invocation using a customizable configuration file.</td>
</tr>
<tr>
<td>xlC</td>
<td>—</td>
<td>C++ compiler invocation using a customizable configuration file.</td>
</tr>
<tr>
<td>xlc++</td>
<td>—</td>
<td>C++ compiler invocation using a customizable configuration file.</td>
</tr>
</tbody>
</table>
## Controlling your environment

<table>
<thead>
<tr>
<th>Command</th>
<th>Standard</th>
<th>XPG</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Display or create a command alias.</td>
</tr>
<tr>
<td>asa</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Interpret ASA/Fortran carriage control.</td>
</tr>
<tr>
<td>automount</td>
<td>—</td>
<td>—</td>
<td>Configure the automount facility.</td>
</tr>
<tr>
<td>cal</td>
<td>—</td>
<td>XPG.2</td>
<td>Display a calendar for a month or year.</td>
</tr>
<tr>
<td>calendar</td>
<td>—</td>
<td>XPG.2</td>
<td>Display all current appointments.</td>
</tr>
<tr>
<td>captoinfo</td>
<td>—</td>
<td>—</td>
<td>Prints terminal entries in the termcap file.</td>
</tr>
<tr>
<td>chcp</td>
<td>—</td>
<td>—</td>
<td>Set or query ASCII/EBCDIC code pages for the terminal.</td>
</tr>
<tr>
<td>configstk</td>
<td>—</td>
<td>—</td>
<td>Configure the AF_UNIX stack.</td>
</tr>
<tr>
<td>env</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Display environments, or set an environment for a process.</td>
</tr>
<tr>
<td>export</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Set the export attributes for variables, or show currently exported variables.</td>
</tr>
<tr>
<td>fc</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Process a command history list.</td>
</tr>
<tr>
<td>hash</td>
<td>—</td>
<td>XPG.2</td>
<td>Create a tracked alias.</td>
</tr>
<tr>
<td>history</td>
<td>—</td>
<td>—</td>
<td>Process a command history list.</td>
</tr>
<tr>
<td>id</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Return the user identity.</td>
</tr>
<tr>
<td>infocmp</td>
<td>—</td>
<td>—</td>
<td>Compare and print the terminal description.</td>
</tr>
<tr>
<td>ipcrm</td>
<td>—</td>
<td>—</td>
<td>Remove message queue, semaphore set, or shared memory identifiers.</td>
</tr>
<tr>
<td>ipcs</td>
<td>—</td>
<td>—</td>
<td>Report status of the interprocess communication facility.</td>
</tr>
<tr>
<td>lm</td>
<td>—</td>
<td>—</td>
<td>Start the login monitor for OCS support.</td>
</tr>
<tr>
<td>locale</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Get locale-specific information.</td>
</tr>
<tr>
<td>localedef</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Define the locale environment.</td>
</tr>
<tr>
<td>logger</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Log messages.</td>
</tr>
<tr>
<td>logname</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Return a user's login name.</td>
</tr>
<tr>
<td>newgrp</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Change to a new group.</td>
</tr>
<tr>
<td>ocscfg</td>
<td>—</td>
<td>—</td>
<td>Configure, unconfigure, or query an OCS object.</td>
</tr>
<tr>
<td>printenv</td>
<td>—</td>
<td>—</td>
<td>Display the value of environment variables.</td>
</tr>
<tr>
<td>r</td>
<td>—</td>
<td>—</td>
<td>Process a command history list.</td>
</tr>
<tr>
<td>readonly</td>
<td>POSIX.2</td>
<td>—</td>
<td>Mark a variable as read-only.</td>
</tr>
<tr>
<td>return</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Return from a shell function or . (dot) script.</td>
</tr>
<tr>
<td>script</td>
<td>—</td>
<td>—</td>
<td>Makes a typescript of a terminal session.</td>
</tr>
<tr>
<td>set</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Set or unset command options and positional parameters.</td>
</tr>
<tr>
<td>shift</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Shift positional parameters.</td>
</tr>
<tr>
<td>stty</td>
<td>POSIX.2</td>
<td>XPG.2</td>
<td>Set or display terminal options.</td>
</tr>
<tr>
<td>su</td>
<td>—</td>
<td>—</td>
<td>Change the user ID connected with a session.</td>
</tr>
<tr>
<td>sysvar</td>
<td>—</td>
<td>—</td>
<td>Display static system symbols.</td>
</tr>
<tr>
<td>tic</td>
<td>—</td>
<td>—</td>
<td>Compile term descriptions into terminfo database entries.</td>
</tr>
</tbody>
</table>
touch POSIX.2 XPG4.2 Change the file access and modification times.
tput POSIX.2 XPG4.2 Change characteristics of terminals.
tso — Run a TSO command from the shell.
tsocmd — Run a TSO/E command from the shell (including authorized commands).
tty POSIX.2 — Return the user's terminal name.
uconvdef — Create binary conversion tables.
unalias POSIX.2 XPG4.2 Remove alias definitions.
uname POSIX.2 XPG4.2 Display the name of the current operating system.
unset POSIX.2 XPG4.2 Unset values and attributes of variables and functions.
who POSIX.2 XPG4.2 Display information about current users.

**Daemons**
cron — Run commands at specified dates and times.
inetd — Handle login requests.
rlogind — Validate rlogin requests.
uupd — Invoke uucico for TCP/IP connections from remote UUCP systems.

**Managing directories**
basename POSIX.2 XPG4.2 Return the nondirectory components of a path name.
cd POSIX.2 XPG4.2 Change the working directory.
chggrp POSIX.2 XPG4.2 Change the group owner of a file or directory.
chmod POSIX.2 XPG4.2 Change the mode of a group or directory.
chown POSIX.2 XPG4.2 Change the owner or group of a file or directory.
chroot — Change the root directory for the execution of a command.
dircmp — XPG4.2 Compare directories
dirname POSIX.2 XPG4.2 Return the directory components of a path name
ls POSIX.2 XPG4.2 List file and directory names and attributes
mkdir POSIX.2 XPG4.2 Make a directory
mount — Logically mount a file system
mv POSIX.2 XPG4.2 Rename or move a file or directory
pathchk POSIX.2 XPG4.2 Check a path name
pwd POSIX.2 XPG4.2 Return the working directory name
rm POSIX.2 XPG4.2 Remove a directory entry
rmdir POSIX.2 XPG4.2 Remove a directory
unlink — XPG5.0 Removes a directory entry
### Managing files

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>amplist</td>
<td>Display formatted information from object and executable files</td>
</tr>
<tr>
<td>as</td>
<td>Use the HLASM assembler to produce object files</td>
</tr>
<tr>
<td>cat</td>
<td>Concatenate or display text files</td>
</tr>
<tr>
<td>chaudit</td>
<td>Change audit flags for a file</td>
</tr>
<tr>
<td>chlabel</td>
<td>Set the multilevel security label to files and directories</td>
</tr>
<tr>
<td>cksum</td>
<td>Calculate and display checksums and byte counts</td>
</tr>
<tr>
<td>cmp</td>
<td>Compare two files</td>
</tr>
<tr>
<td>col</td>
<td>Remove reverse line feeds</td>
</tr>
<tr>
<td>comm</td>
<td>Show and select or reject lines common to two files</td>
</tr>
<tr>
<td>compress</td>
<td>Lempel-Ziv file compression</td>
</tr>
<tr>
<td>copytree</td>
<td>Make a copy of a file hierarchy while preserving all file attributions</td>
</tr>
<tr>
<td>cp</td>
<td>Copy a file</td>
</tr>
<tr>
<td>csplit</td>
<td>Split text files</td>
</tr>
<tr>
<td>ctags</td>
<td>Create tag files for ex, more, and vi</td>
</tr>
<tr>
<td>dot or.</td>
<td>Run a shell file in the current environment</td>
</tr>
<tr>
<td>cut</td>
<td>Cut out selected fields of each line of a file</td>
</tr>
<tr>
<td>dd</td>
<td>Convert and copy a file</td>
</tr>
<tr>
<td>df</td>
<td>Display the amount of free space in the file system</td>
</tr>
<tr>
<td>diff</td>
<td>Compare two text files and show the differences</td>
</tr>
<tr>
<td>du</td>
<td>Summarize usage of file space</td>
</tr>
<tr>
<td>ed</td>
<td>Use the ed line-oriented text editor</td>
</tr>
<tr>
<td>egrep</td>
<td>Search a file for a specified pattern</td>
</tr>
<tr>
<td>ex</td>
<td>Use the ex text editor</td>
</tr>
<tr>
<td>exrecover daemon</td>
<td>Retrieve vi and ex files</td>
</tr>
<tr>
<td>extattr</td>
<td>Set, reset, or display extended attributes for files</td>
</tr>
<tr>
<td>expand</td>
<td>Expand tabs to spaces</td>
</tr>
<tr>
<td>fgrep</td>
<td>Search a file for a specified pattern</td>
</tr>
<tr>
<td>file</td>
<td>Determine file type</td>
</tr>
<tr>
<td>filecache</td>
<td>Manage file caches</td>
</tr>
<tr>
<td>find</td>
<td>Find a file meeting specified criteria</td>
</tr>
<tr>
<td>fold</td>
<td>Break lines into shorter lines</td>
</tr>
<tr>
<td>head</td>
<td>Display the first part of a file</td>
</tr>
<tr>
<td>iconv</td>
<td>Convert characters from one code set to another</td>
</tr>
<tr>
<td>join</td>
<td>Join two sorted, textual relational databases</td>
</tr>
<tr>
<td>line</td>
<td>Copy one line of standard input</td>
</tr>
<tr>
<td>link</td>
<td>Create a hard link to a file</td>
</tr>
<tr>
<td>ln</td>
<td>Create a link to a file</td>
</tr>
<tr>
<td>mkfifo</td>
<td>Make a FIFO special file</td>
</tr>
<tr>
<td>Command</td>
<td>Standard</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td><code>mknod</code></td>
<td>—</td>
</tr>
<tr>
<td><code>mount</code></td>
<td>—</td>
</tr>
<tr>
<td><code>more</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>mv</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>nl</code></td>
<td>—</td>
</tr>
<tr>
<td><code>nm</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>obrowse</code></td>
<td>—</td>
</tr>
<tr>
<td><code>od</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>oedit</code></td>
<td>—</td>
</tr>
<tr>
<td><code>pack</code></td>
<td>—</td>
</tr>
<tr>
<td><code>paste</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>patch</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>pcat</code></td>
<td>—</td>
</tr>
<tr>
<td><code>pg</code></td>
<td>—</td>
</tr>
<tr>
<td><code>sed</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>sort</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>spell</code></td>
<td>—</td>
</tr>
<tr>
<td><code>split</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>strings</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>sum</code></td>
<td>—</td>
</tr>
<tr>
<td><code>tabs</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>tail</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>tee</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>tr</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>tsort</code></td>
<td>—</td>
</tr>
<tr>
<td><code>umask</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>uncompress</code></td>
<td>—</td>
</tr>
<tr>
<td><code>unexpand</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>uniq</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>unmount</code></td>
<td>—</td>
</tr>
<tr>
<td><code>unpack</code></td>
<td>—</td>
</tr>
<tr>
<td><code>uudecode</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>uuencode</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>vi</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>wc</code></td>
<td>POSIX.2</td>
</tr>
<tr>
<td><code>zcat</code></td>
<td>—</td>
</tr>
</tbody>
</table>

### Printing files

<table>
<thead>
<tr>
<th>Command</th>
<th>Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cancel</code></td>
<td>—</td>
<td>— Cancel print queue requests (stub command).</td>
</tr>
<tr>
<td><code>infocmp</code></td>
<td>—</td>
<td>— Compare and print the terminal description.</td>
</tr>
</tbody>
</table>
### Computing and managing logic

<table>
<thead>
<tr>
<th>Command</th>
<th>Standard</th>
<th>XPG Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bc</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Use the arbitrary-precision arithmetic calculation language</td>
</tr>
<tr>
<td><code>break</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Exit from a for, while, or until loop in a shell script</td>
</tr>
<tr>
<td><code>colon</code> or <code>:</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Do nothing, successfully</td>
</tr>
<tr>
<td><code>continue</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Skip to the next iteration of a loop in a shell script</td>
</tr>
<tr>
<td><code>do</code> or <code>.</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Run a shell file in the current environment</td>
</tr>
<tr>
<td><code>eval</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Construct a command by concatenating arguments</td>
</tr>
<tr>
<td><code>exec</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Run a command and open, close, or copy the file descriptors</td>
</tr>
<tr>
<td><code>exit</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Return to the parent process from which the shell was called or to TSO/E</td>
</tr>
<tr>
<td><code>expr</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Evaluate arguments as an expression</td>
</tr>
<tr>
<td><code>false</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Return a nonzero exit code</td>
</tr>
<tr>
<td><code>grep</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Search a file for a specified pattern</td>
</tr>
<tr>
<td><code>if</code> or <code>[</code></td>
<td>—</td>
<td>XPG4.2</td>
<td>Test for a condition</td>
</tr>
<tr>
<td><code>let</code></td>
<td>—</td>
<td>—</td>
<td>Evaluate an arithmetic expression</td>
</tr>
<tr>
<td><code>test</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Test for a condition</td>
</tr>
<tr>
<td><code>trap</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Intercept abnormal conditions and interrupts</td>
</tr>
<tr>
<td><code>true</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Return a value of 0</td>
</tr>
</tbody>
</table>

### Controlling processes

<table>
<thead>
<tr>
<th>Command</th>
<th>Standard</th>
<th>XPG Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bg</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Move a job to the background</td>
</tr>
<tr>
<td><code>bpxtrace</code></td>
<td>—</td>
<td>—</td>
<td>Activate or deactivate traces for processes</td>
</tr>
<tr>
<td><code>cron</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Schedule regular background jobs</td>
</tr>
<tr>
<td><code>fg</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Bring a job into the foreground</td>
</tr>
<tr>
<td><code>jobs</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Return the status of jobs in the current session</td>
</tr>
<tr>
<td><code>kill</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>End a process or job, or send it a signal</td>
</tr>
<tr>
<td><code>nohup</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Start a process that is immune to hangups</td>
</tr>
<tr>
<td><code>ps</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Return the status of a process</td>
</tr>
<tr>
<td><code>renice</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Change priorities of a running process</td>
</tr>
<tr>
<td><code>sleep</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Suspend execution of a process for an interval of time</td>
</tr>
<tr>
<td><code>stop</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Suspend a process or job</td>
</tr>
<tr>
<td><code>submit</code></td>
<td>—</td>
<td>—</td>
<td>Submit a batch job for background processing</td>
</tr>
<tr>
<td><code>suspend</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Send a SIGSTOP to the current shell</td>
</tr>
<tr>
<td><code>time</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Display processor and elapsed times for a command</td>
</tr>
</tbody>
</table>
times — XPG4.2 Get process and child process times
wait POSIX.2 XPG4.2 Wait for a child process to end
ulimit — XPG4.2 Set process limits

**Writing shell scripts**

autoload — — Indicate function name not defined.
dspmsg — — Display selected messages from message catalogs.
functions — — Display or assign attributes to functions.
getconf POSIX.2 XPG4.2 Get configuration values.
getopts POSIX.2 XPG4.2 Parse utility options.
integer — — Mark each variable with an integer value.
read POSIX.2 XPG4.2 Read a line from standard input.
type — XPG4.2 Tell how the shell interprets a name.
typeset — — Assign attributes and values to variables.
xargs POSIX.2 XPG4.2 Construct an argument list and run a command.

**Developing or porting application programs**

ar POSIX.2 XPG4.2 Create or maintain library archives
awk POSIX.2 XPG4.2 Process programs written in the awk language
c89 POSIX.2 XPG4.2 Compile, link-edit, and assemble Standard C source code and create an executable file on z/OS
c++/cxx — — Compile, link-edit, and assemble C++ and Standard C source code and create an executable file on z/OS
cc — XPG4.2 Compile, link-edit, and assemble Common Usage C source code and create an executable file on z/OS
dbx — — Use the debugger
dbglld — — Create a module map for debugging
dspcat — — Display all or part of a message catalog
gencat — XPG4.2 Create or edit message catalogs
lex POSIX.2 XPG4.2 Generate a program for lexical tasks
make POSIX.2 XPG4.2 Maintain program-generated and interdependent files
mkcatdefs — — Preprocess a message source file
runcat — — Pipe output from mkcatdefs to gencat
strip POSIX.2 XPG4.2 Remove unnecessary information from an executable file
yacc POSIX.2 XPG4.2 Use the yacc compiler

**Communicating with the system or other users**

mail — XPG4.2 Read and send mail messages
mailx POSIX.2 XPG4.2 Send or receive electronic mail
### Working with archives

<table>
<thead>
<tr>
<th>Command</th>
<th>Standard</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ar</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Create or maintain library archives</td>
</tr>
<tr>
<td><code>cpio</code></td>
<td>—</td>
<td>XPG4.2</td>
<td>Copy in/out file archives</td>
</tr>
<tr>
<td><code>pax</code></td>
<td>POSIX.2</td>
<td>XPG4.2</td>
<td>Interchange portable archives</td>
</tr>
<tr>
<td><code>tar</code></td>
<td>—</td>
<td>XPG4.2</td>
<td>Manipulate the <code>tar</code> archive files to copy or back up a file</td>
</tr>
</tbody>
</table>

### Working with UUCP

<table>
<thead>
<tr>
<th>Command</th>
<th>Standard</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uucc</code></td>
<td>—</td>
<td>—</td>
<td>Compile UUCP configuration files</td>
</tr>
<tr>
<td><code>uucico</code> daemon</td>
<td>—</td>
<td>—</td>
<td>Process UUCP file transfer requests</td>
</tr>
<tr>
<td><code>uucp</code></td>
<td>—</td>
<td>XPG4.2</td>
<td>Copy files between remote UUCP systems</td>
</tr>
<tr>
<td><code>uucpd</code></td>
<td>—</td>
<td>—</td>
<td>Invoke uucico for TCP/IP connections from remote UUCP systems</td>
</tr>
<tr>
<td><code>uulog</code></td>
<td>—</td>
<td>XPG4.2</td>
<td>Display log information about UUCP events</td>
</tr>
<tr>
<td><code>uuname</code></td>
<td>—</td>
<td>XPG4.2</td>
<td>Display list of remote UUCP systems</td>
</tr>
<tr>
<td><code>uupick</code></td>
<td>—</td>
<td>XPG4.2</td>
<td>Manage files sent by uuto and uucp</td>
</tr>
<tr>
<td><code>uustat</code></td>
<td>—</td>
<td>XPG42</td>
<td>Display status of pending UUCP transfers</td>
</tr>
<tr>
<td><code>uuto</code></td>
<td>—</td>
<td>XPG42</td>
<td>Copy files to users on remote UUCP systems</td>
</tr>
<tr>
<td><code>uux</code></td>
<td>—</td>
<td>XPG42</td>
<td>Request command execution on remote UUCP systems</td>
</tr>
<tr>
<td><code>uuxqt</code> daemon</td>
<td>—</td>
<td>—</td>
<td>Carry out command requests from remote UUCP systems</td>
</tr>
</tbody>
</table>
# Appendix B. Summary of tcsh shell commands

The following list presents the built-in tcsh shell commands, grouped by the task a user might want to perform, and their functions. Similar tasks are organized together.

## General use

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc</td>
<td>Show the amount of dynamic memory acquired.</td>
</tr>
<tr>
<td>builtins</td>
<td>Print the names of all built-in commands.</td>
</tr>
<tr>
<td>bye</td>
<td>Terminate the login shell.</td>
</tr>
<tr>
<td>echo</td>
<td>Write arguments to standard output.</td>
</tr>
<tr>
<td>echotc</td>
<td>Exercise the terminal capabilities in args.</td>
</tr>
<tr>
<td>exec</td>
<td>Run a command and open, close, or copy the file descriptors.</td>
</tr>
<tr>
<td>glob</td>
<td>Write each word to standard output.</td>
</tr>
<tr>
<td>hashstat</td>
<td>Print a statistic line on hash table effectiveness.</td>
</tr>
<tr>
<td>login</td>
<td>Terminate a login shell.</td>
</tr>
<tr>
<td>logout</td>
<td>Terminate a login shell.</td>
</tr>
<tr>
<td>nice</td>
<td>Run a command at a different priority.</td>
</tr>
<tr>
<td>notify</td>
<td>Notify user of job status changes.</td>
</tr>
<tr>
<td>repeat</td>
<td>Execute command count times.</td>
</tr>
<tr>
<td>source</td>
<td>Read and execute commands from name.</td>
</tr>
<tr>
<td>time</td>
<td>Display processor and elapsed times for a command.</td>
</tr>
<tr>
<td>where</td>
<td>Report all instances of command.</td>
</tr>
<tr>
<td>which</td>
<td>Display next executed command.</td>
</tr>
</tbody>
</table>

## Controlling your environment

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@ (at)</td>
<td>Print the value of tcsh shell variables, or assign a value.</td>
</tr>
<tr>
<td>alias</td>
<td>Display or create a command alias.</td>
</tr>
<tr>
<td>bindkey</td>
<td>List all bound keys, or change key bindings.</td>
</tr>
<tr>
<td>complete</td>
<td>List completions.</td>
</tr>
<tr>
<td>history</td>
<td>Display a command history list.</td>
</tr>
<tr>
<td>hup</td>
<td>Run command so it exits on a hang-up signal.</td>
</tr>
<tr>
<td>newgrp</td>
<td>Change to a new group.</td>
</tr>
<tr>
<td>onintr</td>
<td>Control the action of the tcsh shell on interrupts.</td>
</tr>
<tr>
<td>printenv</td>
<td>Display the values of environment variables.</td>
</tr>
<tr>
<td>rehash</td>
<td>Recompute internal hash table.</td>
</tr>
<tr>
<td>sched</td>
<td>Print scheduled event list.</td>
</tr>
<tr>
<td>set</td>
<td>Set or unset command options and positional parameters.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>setenv</code></td>
<td>Set environment variable name to value</td>
</tr>
<tr>
<td><code>settc</code></td>
<td>Tell tcsh shell the terminal capability cap value</td>
</tr>
<tr>
<td><code>setty</code></td>
<td>Control tty mode changes</td>
</tr>
<tr>
<td><code>shift</code></td>
<td>Shift positional parameters</td>
</tr>
<tr>
<td><code>telltc</code></td>
<td>List terminal capability values</td>
</tr>
<tr>
<td><code>unalias</code></td>
<td>Remove alias definitions</td>
</tr>
<tr>
<td><code>uncomplete</code></td>
<td>Remove completions whose names match pattern</td>
</tr>
<tr>
<td><code>unhash</code></td>
<td>Disable use of internal hash table</td>
</tr>
<tr>
<td><code>unlimit</code></td>
<td>Remove resource limitations</td>
</tr>
<tr>
<td><code>unset</code></td>
<td>Unset values and attributes of variables and functions</td>
</tr>
<tr>
<td><code>unsetenv</code></td>
<td>Remove environment variables that match pattern</td>
</tr>
<tr>
<td><code>watchlog</code></td>
<td>Report on users who are logged in.</td>
</tr>
</tbody>
</table>

### Managing directories

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cd</code></td>
<td>Change the working directory</td>
</tr>
<tr>
<td><code>chdir</code></td>
<td>Change the working directory</td>
</tr>
<tr>
<td><code>dirs</code></td>
<td>Print the directory stack</td>
</tr>
<tr>
<td><code>popd</code></td>
<td>Pop the directory stack</td>
</tr>
<tr>
<td><code>pushd</code></td>
<td>Make exchanges within directory stack</td>
</tr>
</tbody>
</table>

### Computing and managing logic

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>break</code></td>
<td>Exit from a loop in a shell script</td>
</tr>
<tr>
<td><code>breaksw</code></td>
<td>Cause a break from a switch</td>
</tr>
<tr>
<td><code>continue</code></td>
<td>Skip to the next iteration of a loop in a shell script</td>
</tr>
<tr>
<td><code>default</code></td>
<td>Label default case in a switch statement</td>
</tr>
<tr>
<td><code>eval</code></td>
<td>Construct a command by concatenating arguments</td>
</tr>
<tr>
<td><code>exec</code></td>
<td>Run a command and open, close, or copy the file descriptors</td>
</tr>
<tr>
<td><code>exit</code></td>
<td>Return to the shell's parent process or to TSO/E</td>
</tr>
<tr>
<td><code>filetest</code></td>
<td>Apply a file inquiry operator to a file</td>
</tr>
</tbody>
</table>

### Managing files

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ls-F</code></td>
<td>List files</td>
</tr>
</tbody>
</table>

### Controlling processes

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bg</code></td>
<td>Move a job to the background.</td>
</tr>
<tr>
<td><code>fg</code></td>
<td>Bring a job into the foreground.</td>
</tr>
<tr>
<td><code>jobs</code></td>
<td>Return the status of jobs in the current session.</td>
</tr>
<tr>
<td><code>kill</code></td>
<td>End a process or job, or send it a signal.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>limit</td>
<td>Limit consumption of processes.</td>
</tr>
<tr>
<td>nohup</td>
<td>Start a process that is immune to hangups.</td>
</tr>
<tr>
<td>stop</td>
<td>Suspend a process or job</td>
</tr>
<tr>
<td>suspend</td>
<td>Send a SIGSTOP to the current shell</td>
</tr>
<tr>
<td>time</td>
<td>Display processor and elapsed times for a command</td>
</tr>
<tr>
<td>wait</td>
<td>Wait for a child process to end</td>
</tr>
</tbody>
</table>
Appendix C. Regular expressions (regexp)

Summary

The commands that use basic and extended regular expressions are as follows:

**Basic**
- ed, expr, grep, sed

**Extended**
- awk, grep with -E option, sed with the -E option.

Table 40 on page 945 summarizes the features that apply to the applicable shell commands.

<table>
<thead>
<tr>
<th>Notation</th>
<th>awk</th>
<th>ed</th>
<th>grep -E</th>
<th>expr</th>
<th>sed</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>^</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[...]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[:]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>re*</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>re+</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>re?</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>re/re</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>d</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(...)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(...)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt;</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>&gt;</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>{,}</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Examples

The following patterns are given as illustrations, along with descriptions of what they match:

**abc**
- Matches any line of text containing the three letters abc in that order.

**a.c**
- Matches any string beginning with the letter a, followed by any character, followed by the letter c.

**^.$**
- Matches any line containing exactly one character (the newline is not counted).
\(a(b^*|c^*)d\)

Matches any string beginning with a letter \(a\), followed by either zero or more of the letter \(b\), or zero or more of the letter \(c\), followed by the letter \(d\).

\(.* \[a-z]+ .*\)

Matches any line containing a word, consisting of lowercase alphabetic characters, delimited by at least one space on each side.

\((morty).*\1\)

\(morty.*morty\)

These expressions both match lines containing at least two occurrences of the string \(morty\).

\([[:space:]][[:alnum:]]\)

Matches any character that is either a white space character or alphanumerical.

**Regular expressions (regexp)**

**Summary**

The commands that use basic and extended regular expressions are as follows:

**Basic**

- ed, expr, grep, sed

**Extended**

- awk, grep with \(-E\) option, sed with the \(-E\) option.

Table 41 on page 946 summarizes the features that apply to the applicable shell commands.

<table>
<thead>
<tr>
<th>Notation</th>
<th>awk</th>
<th>ed</th>
<th>grep -E</th>
<th>expr</th>
<th>sed</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>^</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[...]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[:]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>re*</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>re+</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>re?</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>re/re</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>\d</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(...)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(...)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt;</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>&gt;</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>{ }</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Examples**

Examples of using regular expressions.
The following patterns are given as illustrations, along with descriptions of what they match:

**abc**
Matches any line of text containing the three letters abc in that order.

**a.c**
Matches any string beginning with the letter a, followed by any character, followed by the letter c.

**^.$**
Matches any line containing exactly one character (the newline is not counted).

**a(b*|c*)d**
Matches any string beginning with a letter a, followed by either zero or more of the letter b, or zero or more of the letter c, followed by the letter d.

**.* [a-z]+ .***
Matches any line containing a word, consisting of lowercase alphabetic characters, delimited by at least one space on each side.

**(morty).*\1**

**morty.*morty**
These expressions both match lines containing at least two occurrences of the string morty.

**[[:space:][:alnum:]]**
Matches any character that is either a white space character or alphanumeric.
Appendix D. Running shell scripts or executable files under MVS environments

This topic describes the IBM-supplied BPXBATCH program. It also discusses using OSHELL to run shell commands and scripts from MVS.

BPXBATCH

Related information

BPXBATCH makes it easy for you to run shell scripts and executable files that reside in z/OS UNIX files through the MVS job control language (JCL). If you do most of your work from TSO/E, using BPXBATCH saves you the trouble of going into the shell to run your scripts and executable files. REXX execs can also use BPXBATCH to run shell scripts and executable files. Be sure to read the following sections:

• “General rules for setting return codes” on page 956.
• “Circumstances that cause the internal and real return code to be multiplied by 256 before being issued by BPXBATCH” on page 957.
• “Examples of interpreting BPXBATCH return codes in different environments” on page 957.

In addition to using BPXBATCH, a user who wants to perform a local spawn without being concerned about environment setup (that is, without having to set specific environment variables which could be overwritten if they are also set in the user’s profile) can use BPXBATSL. BPXBATSL provides users with an alternate entry point into BPXBATCH, and forces a program to run using a local spawn instead of fork/exec as BPXBATCH does. This ultimately allows a program to run faster.

BPXBATSL is also useful when the user wants to perform a local spawn of their program but also needs subsequent child processes to be fork/exec’ed. Formerly, this could not be done since BPXBATCH and the requested program shared the environment variables. Failure to meet the following conditions will result in a failure when BPXBATSL is invoked. For more details about these restrictions, see the descriptions of the spawn() function and BPX1SPN callable service in z/OS UNIX System Services Programming: Assembler Callable Services Reference:

• The invoker must have an UID of 0 to issue a SH request.
• The child process is not setuid or setgid to a value different from the parent.
• The spawned file name is not an external link or a sticky bit file.
• The parent has enough resources to allow the child process to reside in the same address space.

BPXBATSL is an alias of BPXBATCH.

BPXBATA2 and BPXBATA8 are provided as APF-authorized alternatives to BPXBATSL. BPXBATA2 and BPXBATA8 provide the capability for a target APF-authorized z/OS UNIX program to run in the same address space as the originating job, allowing it to share the same allocations, job log, and so on. BPXBATA2 is specifically intended to provide the capability for APF-authorized z/OS UNIX program to be started in a PSW Key 2. To ensure that the target program receives control PSW Key 2, a PPT entry for BPXBATA2 must be set up that specifies that BPXBATA2 starts up PSW Key 2.

The same restrictions that apply to BPXBATSL apply to BPXBATA2 and BPXBATA8, in addition to, the following:

• The PGM keyword is the only invocation type that is supported. The SH keyword is not supported.
• The interfaces can only be used from started task address spaces.
• The z/OS UNIX program that is the target of the BPXBATA2 and BPXBATA8 job must be marked as an APF-authorized executable file.
Any other usage of the BPXBATA8 and BPXBATA2 interfaces than what is described is not supported and will cause the invoking job to fail.

**Format**

For JCL:
EXEC PGM=BPXBATCH,PARM='SH|PGM program_name'

For TSO/E:
BPXBATCH SH|PGM program_name

**Description**

The BPXBATCH program allows you to submit MVS batch jobs that run shell commands or scripts, or z/OS XL C/C++ executable files. You can invoke BPXBATCH from a JCL job or from TSO/E (as a command, through a CALL command, or from a CLIST or REXX EXEC).

With BPXBATCH, you can allocate the MVS standard file stdin only as z/OS UNIX files for passing input. You can allocate the MVS standard files stdout, stderr or stdenv as MVS data sets or z/OS UNIX text files. The stdenv file for containing environment variables or the stderr and stdout files for saving job output can be allocated as SYSOUT, PDSE, PDS, or sequential data sets. If you do not allocate them, stdin, stdout, stderr, and stdenv default to /dev/null. Allocate the standard files using the data definition PATH keyword options, or standard data definition options for MVS data sets, for stdenv, stdout and stderr.

For MVS data sets, use the standard data definition options for MVS data sets.

For JCL jobs, specify PATH keyword options on DD statements:

```plaintext
//jobname JOB ...
//stepname EXEC PGM=BPXBATCH,PARM='SH|PGM program_name'
//STDIN   DD  PATH='/stdin-file-pathname',PATHOPTS=(ORDONLY)
//STDOUT  DD  PATH='/stdout-file-pathname',PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
//STDERR  DD  PATH='/stderr-file-pathname',PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
```

Your application in the executable file can also allocate stdin, stdout, stderr, and stdenv dynamically by using SVC 99.

For TSO/E, you specify PATH keyword options on the ALLOCATE command:

```plaintext
ALLOCATE FILE(STDIN) PATH('/stdin-file-pathname') PATHOPTS(ORDONLY)
ALLOCATE FILE(STDOUT) PATH('/stdout-file-pathname')
ALLOCATE FILE(STDERR) PATH('/stderr-file-pathname')

BPXBATCH SH|PGM program_name
```

stdin and stdenv must always be allocated as read. stdout and stderr must always be allocated as write.

As previously stated, a user who wants to perform a local spawn without being concerned about environment setup (that is, without having to set specific environment variables which could be overwritten if they are also set in the user’s profile) can use BPXBATSL. BPXBATSL provides users with an alternate entry point into BPXBATCH, and forces a program to run using a local spawn instead of fork/exec as BPXBATCH does. This ultimately allows a program to run faster.
The following example contains DD statements that are accessible to a program that was given control from BPXBATCH:

```plaintext
//jobname JOB ...
//stepname EXEC PGM=BPXBATCHSL,PARM='PGM program_name'
/* The following 2 DDs are still available in the program which gets
* control from BPXBATCH:
//DD1       DD DSN=MVSDSN.FOR.APPL1,DISP=SHR
//DD2       DD DSN=MVSDSN.FOR.APPL2,DISP=SHR
/* The following DDs are processed by BPXBATCH to create file descriptors
*/ for stdin, stdout, stderr
//STDIN   DD  PATH='/stdin-file-pathname',PATHOPTS=(ORDONLY)
//STDOUT  DD  PATH='/stdout-file-pathname',PATHOPTS=(OWRONLY,OCREAT,OTRUNC
//            PATHMODE=SIRWXU
//STDERR  DD  PATH='/stderr-file-pathname',PATHOPTS=(OWRONLY,OCREAT,OTRUNC
//            PATHMODE=SIRWXU
```

### Parameters

BPXBATCH accepts one parameter string as input, the combination of SH|PGM and program_name. At least one blank character must separate the parts of the parameter string. The total length of the parameter string will support up to 32754 characters.

BPXBATCH was first created for use with JCL, which had a maximum parameter length of 100. Now, BPXBATCH can also be called form CLIST, REXX, and TSO. These additional environments do not have the 100 character parameter limit. From a TSO command environment, the parameter string itself will now support up to 32754 characters.

Parameters to BPXBATCH can also be supplied via the stdparm DD up to a limit of 65,536 characters. When the stdparm DD is allocated BPXBATCH will use the data found in the z/OS UNIX file or MVS data set associated with this DD rather than what is found on the parameter string or in the stdin DD. An informational message BPXM079I is displayed indicating that this is occurring, as a warning to the user. The stdparm DD will allow either a z/OS UNIX file, or a MVS SYSIN, PDS or PDSE member or a sequential data set.

If neither SH nor PGM is specified as part of the parameter string, BPXBATCH assumes that the shell is to be started to run the shell script allocated by stdin.

**SH|PGM**

Specifies whether BPXBATCH is to run a shell script or command or a z/OS XL C/C++ executable file located in a z/OS UNIX file.

**SH**

Specifies that the shell designated in your TSO/E user ID’s security product profile is to be started and is to run shell commands or scripts provided from stdin or the specified program_name.

If SH is specified with no program_name information, BPXBATCH attempts to run anything read in from stdin.

**PGM**

Specifies that the program identified by the program_name parameter is invoked directly from BPXBATCH. This is done either via a spawn or a fork and exec. BPXBATCH creates a process for the program to run in and then calls the program. If you specify PGM, you must also specify program_name.

All environment variables read from the stdenv file are set when the program is run if stdenv was allocated. If the HOME and LOGNAME variables are not specified in the stdenv file, or stdenv was not allocated, then HOME and LOGNAME, if possible, are set when the program is run.

Refer to “Usage notes” on page 952 for more information about environment variable processing.

**Restriction:** When using PGM, the program_name cannot contain any shell specific functions because they will not be resolved. If shell specific functions must be specified, then SH should be used to avoid possible errors or unpredictable results.
**program_name**

Specifies the shell command name or the z/OS UNIX path name for the shell script or z/OS XL C/C++ executable file to be run. In addition, program_name can contain option information.

The program_name is interpreted as case-sensitive.

When PGM and program_name are specified and the specified program name does not begin with a slash character (/), BPXBATCH prefixes the user's initial working directory information to the program path name.

**Usage notes**

1. BPXBATCH is an alias for the program BPXMBATC, which resides in the SYS1.LINKLIB data set.
2. BPXBATCH must be invoked from a user address space running with a program status word (PSW) key of 8.
3. BPXBATCH does not translate characters on the supplied parameter information. You should supply parameter information, including z/OS UNIX path names, using only the POSIX portable character set. For information about the POSIX portable character set, see z/OS UNIX System Services Programming Tools.
4. If your BPXBATCH job returns ABEND 4093 reason code 0000001c, you need to expand the region size. For example:

   ```shell
   //SHELLCMD  EXEC PGM=BPXBATCH,REGION=8M,PARM='SH shell_cmd'
   ```

5. BPXBATCH does not support any ddnames other than stdin, stdout, stderr, stdenv or stdparm. Attempting to allocate or reference any other ddnames will result in enqueue failures or unpredictable results. To use an MVS data set in your batch UNIX application, use "dynamic allocation", such as SVC99 or the TSO ALLOC command. Also, you must remove all "static allocations" (ddnames referring to the MVS data set in question) from all steps in the batch job.
6. If you define an MVS data set for stdout or stderr, consider the following:
   - It must be a sequential data set, a partitioned data set (PDS) member, a partitioned data set extended (PDSE) member, or SYSOUT.
   - The data set must have a nonzero logical record length (LRECL) and a defined record format (RECFM); otherwise, BPXBATCH will fail with error message BPXM012I indicating an open failure for the affected ddname.
   - If the LRECL of the target stdout or stderr data set is not large enough to hold a line of output, the data will be truncated and message BPXM080I will be put out indicating this has occurred. This can happen for both fixed and variable blocked data sets. For variable block data sets, the first four bytes of each record, record segment, or block make up a descriptor word containing control information. You must allow for these additional 4 bytes in the specified LRECL if you intend to avoid truncation of the output to the stdout and stderr DDs.
   - If you use two members of the same partitioned data set for the stdout or stderr ddnames, then you must use a PDSE (not a PDS). Using a PDS instead of a PDSE can result in a 213 abend (and, if running in a batch job, an abnormal end for the job step) or the output not appearing in the members as expected.
   - When you specify an MVS data set for either the stdout or stderr ddnames, a child process will be created to run the target z/OS UNIX program. In some cases, the child process will run in a separate address space from the BPXBATCH job. In such cases, the job log messages for the child will not appear in the job log of the BPXBATCH job. To capture the child's job log messages, set the _BPXK_JOBLOG=STDERR environment variable. This will cause the child's job log messages to be written to the stderr data set specified in the BPXBATCH job.
   - In early releases of z/OS, if a MVS data set were specified on stdout or stderr, BPXBATCH ignored the data set and defaulted to /dev/null. To remain compatible with this behavior, the current support does the same defaulting if the MVS data set type is not supported (for example,
DD Dummy, Terminal, or SYSIN), or if the MVS data set cannot be opened by BPXBATCH. Also, message BPXM081I is displayed that indicates when this default behavior is being taken by BPXBATCH.

- If stdout or stderr are allocated as a PDS or PDSE member and overwriting of the output is expected from multiple runs of the same job or command, the data set should not be allocated with a disposition of NEW but rather as SHR or OLD. If the data set is allocated as NEW, the member will be created on the 1st run, but subsequent runs will cause i/o errors when attempting to write to the member.

- If stdout or stderr are allocated as a sequential data set and appending of the output is expected from multiple runs of the same job or command, the data set should be allocated with a disposition of MOD.

- In general, any I/O errors that occur with an MVS data set defined to the stdout or stderr (or stdparm or stdenv) ddnames will result in an abend (x13 or x37, for instance) and, if running in a batch job, an abnormal end for the job step, except for an abend B37, which will be ignored. For example: If the user does not have security access to the data set defined to stdout, then when BPXBATCH attempts to open the data set, a 913 abend will occur and message IEC150I will provide details about the error.

- To avoid the possibility of a timeout abend when directing stdout and stderr to an MVS data set or to SYSOUT, specify TIME=1440 (NOLIMIT) on the EXEC statement.

7. BPXBATCH now supports a parameter string up to 32754 characters when called from a TSO command environment. Also from both a batch and TSO environment, up to 65,536 characters can now be supplied via the a new input DD named stdparm. When the stdparm DD is allocated BPXBATCH will use the data found in the z/OS UNIX file or MVS data set associated with this DD rather that what is found on the parameter string or in the stdin DD. As a warning to the user, an informational message BPXM079I will be displayed indicating that this is occurring.

The stdparm DD will allow either a z/OS UNIX file, or an MVS SYSIN PDS, PDSE or sequential data set. The following are characteristics of the parameter data that can be supplied in the stdparm DD, if a z/OS UNIX file is specified:

- It must be a text file defined with read access only
- Specify one argument per line
- The file cannot have sequence numbers in it.

**Tip:** If you use the ISPF editor to create the file, set the sequence numbers off by typing number off on the command line before you begin typing data. If sequence numbers already exist, type UNNUM to remove them and then type number off.

If a MVS data set is specified:

- Specify one argument per line. If the parameter string for an argument spans more than one line of a data set or file, this string will be divided into two or more arguments that are passed to the corresponding shell script or program.
- The maximum length of a single argument supplied to the program is 32,760, which is the same as the maximum LRECL for an unspanned non-VSAM data set.
- The record format of the data set can fixed or variable (unspanned).
- The data set cannot have sequence numbers in it. If you use the ISPF editor to edit the data set, set the sequence numbers off by typing number off on the command line before you begin typing in the data. If sequence numbers already exist, type UNNUM to remove them and set number mode off.
- Trailing blanks are truncated for SYSIN and variable block data sets, but not for fixed block data sets. For a fixed block data set, trailing blanks will be included in the parameter text for a given argument up to the end of the record.

8. BPXBATCH does not close file descriptors other than - -2. Other file descriptors that are open and not defined as “marked to be closed” remain open when you call BPXBATCH and BPXBATCH runs the specified script or executable file.
9. BPXBATCH uses write-to-operator (WTO) routing code 11 to write error messages to either the JCL job log or your TSO/E terminal. Your TSO/E user profile must specify WTPMSG so that messages can be displayed at the terminal.

10. BPXBATCH (with the SH parameter) must not be used to run an executable file, shell command, or shell script in the background (by specifying the shell & symbol) unless the shell nohup command is also used. If the shell ampersand (&) symbol is used without nohup, the results are unpredictable.

11. BPXBATCH, when used with the PGM parameter, sets up environment variables for the program to be run. If the stdenv file is not allocated, the HOME and LOGNAME environment variables are set. If stdenv is allocated, the environment variables read from the file it represents are set, with HOME or LOGNAME or both environment variables added if they are not specified in the stdenv file. The following types of files can be allocated to stdenv:
   - z/OS UNIX text file
   - Sequential format MVS data set (including SYSIN data set)
   - Member of a partitioned data set (PDS)
   - Member of a partitioned data set extended (PDSE)

Other forms of MVS data sets, such as DUMMY, TERMINAL, or SYSOUT are not supported for stdenv.

The stdenv file consists of one or more records, where record is defined as a string terminated with a <newline> character (X'15') in a z/OS UNIX file, or a fixed or variable (nonspanned) format record in an MVS data set. Other MVS record formats are not supported for stdenv. The following rules apply to the specification of environment variables in stdenv files:

- Only one environment variable can be specified per record.
- Each environment variable is specified as variable=value.
- Environment variable names must begin in column 1, unless names beginning with blanks are used.
- Environment variable records should not be terminated with null characters (X'00'). BPXBATCH automatically appends a null character to the end of each environment variable, and the lengths of environment variables as seen by the program include the null characters.
- Trailing blanks (X'40') are truncated for MVS SYSIN data sets, but are not truncated for any other type of file.
- Be careful that sequence numbers are not present in MVS data sets, because they will be treated as part of the environment variables. ISPF edit users should always set number mode off when creating environment variables, including JCL data sets with environment variables specified as SYSIN.

Some environment variables are release-dependent. If BPXBATCH is executed on a system that does not support the environment variable, you will not get an error message and the variable will be ignored. Use the uname shell command to determine the release number of the operating system that BPXBATCH is running on.

Environment variables (including PATH) are established at the start of the executable program, not for BPXBATCH itself. Thus, PATH is not searched to locate the program, but instead is used if the program invokes other executable programs. In the following example, someprogram might be found only in the initial working directory defined by the user's profile, not by the PATH environment variable:

```
//jobname JOB ...
//stepname EXEC PGM=BPXBATCH,PARM='PGM someprogram parm1 parm2'
//STDOUT DD PATH="/tmp/pgmout",PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//               PATHMODE=SIRWXU
//STDENV DD *
//PATH="/bin:/u/usr/joeuser STEPLIB=SYS1.JOE.STEPLIB
/*
```

12. BPXBATCH uses two more environment variables for execution that are specified by stdenv:
The _BPX_BATCH_UMASK environment variable allows the user the flexibility of modifying the permission bits on newly created files instead of using the default mask (when PGM is specified).

**Note:** This variable is overridden by umask (usually set from within /etc/profile) if BPXBATCH is invoked with the 'SH' option (SH is the default). SH causes BPXBATCH to execute a login shell which runs the /etc/profile script (and runs the user's .profile) and which may set the umask before execution of the intended program.

_BPX_BATCH_SPAWN causes BPXBATCH to use SPAWN instead of fork/exec and allows data definitions to be carried over into the spawned process. When _BPX_BATCH_SPAWN is set to YES, spawn will be used. If it is set to NO, which is equivalent to the default behavior, fork/exec will be used to execute the program.

If _BPX_BATCH_SPAWN is set to YES, then you must consider two other environment variables that affect spawn (BPX1SPN):

- _BPX_SHAREAS = YES|NO|REUSE|MUST
  - When YES or REUSE, the child process created by spawn will run in the same address space. Failure to meet these conditions will result in a spawn failure when MUST is used. For more detail about these restrictions see the descriptions of the spawn() function and BPX1SPN callable service in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*:
    - The invoker must have an UID of 0 to issue a SH request
    - The child process is not setuid or setgid to a value different from the parent
    - The spawned file name is not an external link or a sticky bit file
    - The parent has enough resources to allow the child process to reside in the same address space
    - The NOSHAREAS extended attribute is not set
  - When no, the child and parent run in separate address spaces.

- _BPX_SPAWN_SCRIPT=YES
  - Spawn will recognize a header in the first line of a z/OS UNIX file that indicates the file to be executed and its first set of arguments. This header will only be recognized when a z/OS UNIX file is not found in an executable format. The format of the header is as follows:

```
#! Path String
```

where #! is the file magic number. The magic number indicates that the first line of a file is a special header that contains the name of the program to be executed and any argument data to be supplied to it.

When _BPX_SPAWN_SCRIPT=yes, spawn will first recognize the file magic number and will process the file accordingly. If the file magic number is not found in the file's first line, spawn will treat the specified file as a shell script and will invoke the shell to run the shell script.

For more information about spawn, see BPX1SPN in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.

13. When using BPXBATCH with the SH parameter, environment variables specified in the stdenv DD are overridden by those specified in /etc/profile and .profile (which overrides /etc/profile). This is because SH causes BPXBATCH to execute a login shell which runs the /etc/profile script and runs the user's .profile.

**Files**
- SYS1.LINKLIB(BPXMBATC) is the BPXBATCH program location.
- The default for stdin is /dev/null.
• The default for stdout is /dev/null.
• The default for stderr is the value of stdout. If all defaults are accepted, stderr is /dev/null.
• The default for stdenv is /dev/null.

**BPXBATCH return codes**

0
Processing successful.

254
Processing unsuccessful. BPXBATCH requires OMVS to be started.

255
Processing unsuccessful. An error message was issued.

4095
Processing unsuccessful. An error message was issued.

32000
BPXBATCH invoked the BPX1FRK (fork) callable service. This is usually invoked only by a TSO/E user. One of the following conditions might have resulted:

• BPXBATCH failed to open specified files after the program fork. Files are normally opened and closed prior to a fork. Try running BPXBATCH again.
• The program_name or the shell exited with an exit status of 125.

32512
One of the following conditions might have occurred:

• The PGM keyword was specified for BPXBATCH and no program_name could be found. Message BPXM008I was written to the job log or stderr.
• The SH keyword was specified for BPXBATCH and either /bin/login or the shell did not exist.
• The SH keyword was specified with a program_name value for BPXBATCH and no program_name could be found. The shell exited with an exit status of 127. stdout contains a shell message indicating the program was not found.
• The program_name or the shell exited with an exit status of 127.

**other multiples of 256**
A return code greater than 255, unless explicitly documented as a return code from BPXBATCH (32000 or 32512), is actually an exit status being returned from the program that was invoked by BPXBATCH. The exit status can be determined by dividing the value of BPXYWAST by 256.

**BPXYWAST**
BPXBATCH invoked the BPX1FRK (fork) callable service. This is usually invoked only by a TSO/E user. Processing was successful with wait() status containing a nonzero value. The wait status was mapped by BPXYWAST and returned by BPX1WAT (wait).

No error messages were issued by BPXBATCH.

**General rules for setting return codes**
When running a shell script, your job log should reflect the failure in the script if you code set -o errexit in the script itself. The command set -o errexit tells the noninteractive shell to execute the ERR trap and then exit. As soon as the script encounters an error, processing of the shell script ends, and you might see a console message such as the following one:

```
$HASP165 jobname ENDED AT PLPSC MAXCC=23
```

In the job log you might find:
Without set -o errexit coded in your script, the script is going to keep executing even if it encounters an error, and the last shell command issued in your script is what is reflected in the job log. If this last shell command is successful, then you get MAXCC=00.

Return Codes < 128 indicates a failure of the command. You need to look at the application script that is running to determine what the return code means. The script is either setting the return code explicitly or the last command that is executed in the script has an exit status equaling the return code.

If Exit Status >= 128, the process was terminated by a signal. Subtract 128 from the RC to get the signal number. A list of signal numbers and their function is documented in the table of z/OS UNIX signals in z/OS UNIX System Services Programming: Assembler Callable Services Reference.

For example, if the return code is 157, the signal number is 157-128, which is Signal 29. Signal 29 is SIGXCPU - CPU time limit exceeded.

The shell command `kill` can be used to find the signal number. If `nnn` is the return code, shell command `kill -l nnn` returns the signal name. For example, `kill -l 157` returns XCPU

Circumstances that cause the internal and real return code to be multiplied by 256 before being issued by BPXBATCH

1. BPXBATCH is run from within a TSO environment.
2. Environment variable `_BPX_BATCH_SPAWN=YES` is set. (This means using spawn instead of Fork/Exec.)
3. STDOUT and/or STDERR are pointing to an MVS data set or to SYSOUT.

Examples of interpreting BPXBATCH return codes in different environments

1. The OSHELL REXX exec invokes BPXBATCH to run noninteractive shell commands from the TSO/E READY prompt.
   
   Issuing `tso oshell exit 255` results in `RC = 65280`. The value 65280 is set by multiplying the return code 255 by 256 (255 * 256 = 65280).

2. A job or job step return code can only be a remainder of a value divided by 4096. In the previous example, where the return code was 65280, the return code would be converted as follows: 65280/4096 = 3840.

3. In a situation with a BPXBATCH job (not running under TSO) and STDOUT and STDERR pointing to the default `/dev/null` or to a UNIX file, it is the environment variable `_BPX_BATCH_SPAWN` that decides how the return code is set and what you can find in the job log.

Using the simple UNIX command `exit 3` as an example, the return codes would be as follows:

```
> BPXBATCH jobstep information for _BPX_BATCH_SPAWN=YES:
JOBNAME  STEPNAMES  PROCSTEP  RC
jobname  OMVS             768
> BPXBATCH jobstep information for _BPX_BATCH_SPAWN=NO:
JOBNAME  STEPNAMES  PROCSTEP  RC
jobname  OMVS             00
jobname  *OMVSEX           00
jobname  *OMVSEX           03
```

The following examples are all taken from an environment having BPXBATCH using stdout and stderr pointing directly to SYSOUT:

```
_//OMVS     EXEC PGM=BPXBATCH,PARM='SH sleep 300'
_//STDOUT   DD SYSOUT=*,LRECL=137,RECFM=VB
_//STDERR   DD SYSOUT=*,LRECL=137,RECFM=VB
```
After starting the job, the sleep process can be stopped by sending a signal to the job.

```bash
> kill -s signame process
```

`signame` is the signal name. For example USR1 for SIGUSR1 process is the process ID.

Table 42 on page 958 lists signals sent to the process via the **kill** command and the return codes in the job output:

<table>
<thead>
<tr>
<th>Signal number</th>
<th>Signal name</th>
<th>Calculation</th>
<th>Job return code</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HUP</td>
<td>((128+01)*256)/4096</td>
<td>256</td>
<td>Same as with RC=1</td>
</tr>
<tr>
<td>3</td>
<td>ABRT</td>
<td>((128+03)*256)/4096</td>
<td>768</td>
<td>None</td>
</tr>
<tr>
<td>9</td>
<td>KILL</td>
<td>((128+09)*256)/4096</td>
<td>2304</td>
<td>(**)</td>
</tr>
<tr>
<td>14</td>
<td>ALRM</td>
<td>((128+14)*256)/4096</td>
<td>3584</td>
<td>None</td>
</tr>
<tr>
<td>15</td>
<td>TERM</td>
<td>((128+15)*256)/4096</td>
<td>3840</td>
<td>None</td>
</tr>
<tr>
<td>16</td>
<td>USR1</td>
<td>((128+16)*256)/4096</td>
<td>0</td>
<td>RC=0 but not OK</td>
</tr>
<tr>
<td>17</td>
<td>USR2</td>
<td>((128+17)*256)/4096</td>
<td>256</td>
<td>Same as signal 1</td>
</tr>
<tr>
<td>24</td>
<td>Quit</td>
<td>((128+24)*256)/4096</td>
<td>2048</td>
<td>None</td>
</tr>
<tr>
<td>25</td>
<td>TSTP</td>
<td>((128+25)*256)/4096</td>
<td>2304</td>
<td>Same as signal 9</td>
</tr>
</tbody>
</table>

It is difficult to decipher the signal sent from the return code for several reasons:

- Signal 16 and 32 result in a RC=0, which generally indicates successful job completion.
- Signal x and signal x+16 both end up having the same return code; one signal cannot be distinguished from the other. For example, signal 9 and signal 25 (9+16) both have RC = 2304.

However, with STDERR, you will typically see a CEE message showing the actual signal causing the process to end:

```text
CEE5208S The signal SIGname was received
```

**Using OSHELL to run shell commands and scripts from MVS**

You can use the OSHELL REXX exec to run a shell command or shell script from the TSO/E READY prompt and display the output to your terminal. This exec uses BPXBATCH to run the shell command or shell script:

```bash
oshell shell_command
```

For example, to display process information, enter:

```bash
oshell ps -ej
```

With this exec, do not use an & to run a shell command in the background.
Appendix E. BPXCOPY: Copying a sequential or partitioned data set or PDSE member into files

Use BPXCOPY to copy sequential or partitioned data sets or PDSE members into files.

Format

JCL:

```
EXEC PGM=BPXCOPY,PARM='ELEMENT HEADID LINK TYPE PATHMODE SYMLINK SYMPATH APF | NOAPF PROGCTL | NOPROGCTL SHAREAS | NOSHAREAS UID GID SHARELIB | NOSHARELIB
```

Description

BPXCOPY copies a file, a sequential data set, or partitioned data set or PDSE member into a file. You can invoke BPXCOPY in several ways:

- From JCL using EXEC PGM=BPXCOPY. BPXCOPY does not need the Terminal Monitor Program (TMP) to be started when it is invoked from JCL.
- From LINK, XCTL, ATTACH, a TSO/E CALL command with the asis option, or by a CALL after a LOAD.

BPXCOPY provides similar function to the OPUT command, but differs from OPUT in these ways:

- There is no code page conversion available.
- The specified file name cannot be longer than 8 characters.
- The path name of the directory that is specified cannot be longer than 255 characters.
- You can define hard links to the file.
- You can define symbolic links to the file.
- You can set the permission access bits of the file.
- You can set the extended attributes of the file.
- You can set the owning UID and GID of the file.
- Do not specify PATHOPTS when using the TSO/E ALLOCATE command or a JCL DD statement. It will be ignored.

A DD statement allocates a data set or file and sets up a ddname. For BPXCOPY:

- The input ddname can specify an MVS data set (either a sequential data set or a member of a partitioned data set or PDSE) or the input ddname can be the full path name of the file. When you invoke BPXCOPY from JCL, you must use SYSUT1 as the input ddname. If BPXCOPY is invoked from LINK, XCTL, or ATTACH, a TSO/E CALL command with the asis option, or by a call after a LOAD, you can specify an alternative ddname.
- The output ddname is associated with the path name of the directory in which the file resides. The absolute path name for the file is this path name combined with the name specified with the ELEMENT parameter. When you invoke BPXCOPY from JCL, you must use SYSUT2 as the output ddname. If BPXCOPY is invoked from LINK, XCTL, or ATTACH, a TSO/E CALL command with the asis option, or by a CALL after a LOAD, you can specify an alternative ddname.
- The message output ddname is associated with an MVS data set. The default ddname is SYSTSPRT, which typically directs messages to SYSOUT. When you invoke BPXCOPY from JCL, you must use SYSTSPRT as the message output ddname. SYSTSPRT’s default LRECL is 137, with a BLKSIZE of 3155. If BPXCOPY is invoked from LINK, XCTL, or ATTACH, a TSO/E CALL command with the asis option, or by a CALL after a LOAD, you can specify an alternative ddname.
- BPXCOPY invokes IKJTSOEV, which will always have an allocation for ddnames SYSTSIN and SYSTSPRT. SYSTSPRT must be allocated correctly as described in the preceding bullet and ensured that it is closed.

© Copyright IBM Corp. 1996, 2019
after entry to BPXCOPY. See *z/OS TSO/E Programming Services* for more information about the IKJTSOEV service.

**Parameters**

You can specify the following keyword parameters with BPXCOPY. The parameters can be separated by any delimiter (space, comma, tab, or comment (/*)).

**ELEMENT(element_name)**

element_name is a simple 1-to-8-character file name of the output file. The element_name specified is converted to uppercase characters.

The directory path name for the output file is specified with the PATH keyword on a JCL DD statement.

The path name of the output file consists the directory path name that is appended with the element_name.

This parameter is required.

**HEADID('character_string')**

An 8-byte character string, enclosed in single quotation marks, that will appear on the header of each page of output created.

This optional parameter is provided for SMP/E usage, not for a typical user.

**LINK('linkname','linkname',...)**

The names of hard links to the file. Each linkname is concatenated with the output directory path name. On the JCL DD statement for the directory, the maximum length for a path name (before concatenation) is 255 characters. Path names with a length of up to 1023 characters can be specified only if BPXCOPY is invoked from LINK, XCTL, or ATTACH, a TSO/E CALL command, or by a CALL after a LOAD.

If you specify this parameter, you create one or more hard links to the file when the data is copied into a file. The linkname must be enclosed in single quotation marks. You can specify up to 64 linknames, and each must be enclosed in single quotes. Specifying LINK is optional.

**SYMLINK('linkname','linkname',...)**

The names of symbolic links to the file. Each linkname is concatenated with the output directory path name. On the JCL DD statement for the directory, the maximum length for a path name (before concatenation) is 255 characters. Path names with a length of up to 1023 characters (after concatenation) can be specified if BPXCOPY is involved from LINK, XCTL, or ATTACH, a TSO/E CALL command, or by a CALL after a LOAD.

If you specify this parameter, you create one or more symbolic links to the file. The linkname must be enclosed in single quotation marks. You can specify up to 64 linknames, and each must be enclosed in single quotation marks. Specifying SYMLINK is optional. If you specify SYMLINK, you must also specify SYMPATH.

**SYMPATH('path name','pathname',...)**

The path names of the file for which the symbolic link is created. Each path name may be an absolute path name (beginning with a slash) or a relative path name (not beginning with a slash). When an absolute path name is used, the symbolic link will be resolved starting at the root directory. When a relative path name is used, the symbolic link will be resolved starting at the parent directory of the symbolic link.

For JCL, the maximum length for a path name is limited by the 100 character limit on the entire PARM string (including other parameters) on the EXEC statement. Path names with a length of up to 1023 characters can be specified if BPXCOPY is invoked from LINK, XCTL, or ATTACH, a TSO/E CALL command, or by a CALL after a LOAD.

Specifying SYMPATH is optional, but if you specify SYMPATH, you must also specify SYMLINK. Each SYMLINK linkname must be matched with a corresponding SYMPATH path name. The first linkname define a symbolic link to the first path name, the second linkname defines a symbolic link to the...
second path name, and so on. If there are fewer path names than linknames, the last path name is used for the remaining linknames.

**PATHMODE (mode_bits)**

Changes the access permissions, or modes, of the specified file or directory. Modes determine who can read, write, or search a directory. The bits are used to set execution and permission access of the output file. On BPXCOPY, you can specify PATHMODE as an absolute mode; it must consist of four octal numbers that are separated by commas or blanks.

Absolute modes are four octal numbers specifying the complete list of attributes for the files. Specify attributes by ORing together the bits for each octal number.

<table>
<thead>
<tr>
<th>Octal Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4,0,0,0</td>
<td>Set-user-ID bit</td>
</tr>
<tr>
<td>2,0,0,0</td>
<td>Set-group-ID bit</td>
</tr>
<tr>
<td>1,0,0,0</td>
<td>Sticky bit</td>
</tr>
<tr>
<td>0,4,0,0</td>
<td>Individual read</td>
</tr>
<tr>
<td>0,2,0,0</td>
<td>Individual write</td>
</tr>
<tr>
<td>0,1,0,0</td>
<td>Individual execute (or list directory)</td>
</tr>
<tr>
<td>0,8,4,0</td>
<td>Group read</td>
</tr>
<tr>
<td>0,8,2,0</td>
<td>Group write</td>
</tr>
<tr>
<td>0,8,1,0</td>
<td>Group execute</td>
</tr>
<tr>
<td>0,0,4,0</td>
<td>Other read</td>
</tr>
<tr>
<td>0,0,2,0</td>
<td>Other write</td>
</tr>
<tr>
<td>0,0,0,1</td>
<td>Other execute</td>
</tr>
</tbody>
</table>

Specifying PATHMODE is optional.

For more information about permission bits, see the chmod command.

**TYPE (TEXT|BINARY)**

The format for the file. The default is BINARY for U-format data sets and TEXT for all others. (U-format means undefined-length records.) Specifying TYPE is optional.

**APF|NOAPF**

Specifies whether the APF extended attribute is set or unset. When this attribute is set (APF) on an executable program file (load module), it behaves as if loaded from an APF-authorized library. For example, if this program is exec()ed at the job step level and the program is linked with the AC = 1 attribute, the program is executed as APF-authorized.

To be able to set APF, you must have at least READ access to the BPX.FILEATTR.APFP resource in the FACILITY class.

Specifying APF or NOAPF is optional. If not specified, the attribute is defined as NOAPF.

**PROGCTL|NOPROGCTL**

Specifies whether the PROGCTL extended attribute is set or unset. When this is set (PROGCTL) on an executable program file (load module), the program to behave as if an RDEFINE had been done for the load module to the PROGRAM class. When this program is brought into storage, it does not cause the environment to be marked dirty.

To be able to set PROGCTL, you must have at least READ access to the BPX.FILEATTR.PROGCTL resource in the FACILITY class.

Specifying PROGCTL or NOPROGCTL is optional. If not specified, the attribute is defined as NOPROGCTL.

**SHAREAS | NOSHAREAS**

Specifies whether the SHAREAS extended attribute is set or unset. When this attribute is set (SHAREAS) on an executable program file (load module), the _BPX_SHAREAS environment variable is honored when the file is spawn()ed. When this attribute is not set (NOSHAREAS), the _BPX_SHAREAS environment variable is ignored when the file is spawn()ed.

Specifying SHAREAS or NOSHAREAS is optional. If not specified, the attribute will be defined as SHAREAS.

**SHARELIB | NOSHARELIB**

Specifies whether the st_ShareLib extended attribute is set or unset in the target file.
Note: In order to use BPXCOPY with this keyword parameter, you must have at least READ access to the BPX.FILEATTR.SHARELIB resource in the FACILITY class.

UID(owner)
 Specifies the owner of the file. Owner can be a user name or a numeric user ID (UID). However, if a numeric owner exists as a user name in the user data base, the UID number associated with that user name is used.

Specifying the UID is optional. If it is not specified, the UID of the user running BPXCOPY is used.

These requirements must be met when specifying the UID:
• To be able to set the UID of the file, the user must have UID 0 or have at least READ access to the BPX.SUPERUSER resource in the FACILITY class.
• The UID must be known to the system.
• If a mixed case user name is specified, it must be enclosed in single quotation marks.

GID(group)
 Specifies the group owner of the file. group can be a group name or a numeric group ID (GID). However, if a numeric group exists as a group name in the group data base, the GID number associated with that group name is used.

Specifying the GID is optional. If it is not specified, the GID of the directory path name is used.

These requirements must be met when specifying the GID:
• To be able to set the GID of the file, the user must have UID 0 or have at least READ access to the BPX.SUPERUSER resource in the FACILITY class.
• If a mixed case user name is specified, it must be enclosed in single quotation marks.

Return codes

0
 Processing successful

12
 Processing unsuccessful. An error message has been issued.

Examples

1. JCL and BPXCOPY are used to copy a PDSE member into a directory. These facts are known:
   • The name of the PDSE member is REGEREX.
   • The directory name is /u/turbo/llib.
   • Output messages are to be directed to SYSOUT.
   • Type of data: binary.

   //TEST JOB MSGLEVEL=(1,1)
   //STEP EXEC PGM=BPXCOPY,
   //  PARM='ELEMENT(REGEREX) LINK("../erex") TYPE(BINARY)'
   //SYSUT1 DD DSN=TURBO.LOADLIB(REGEREX),DISP=SHR
   //SYSUT2 DD PATH="/u/turbo/llib"
   //SYSTSPRT DD SYSOUT=*

   The LINK name is concatenated with the directory name from SYSUT2, yielding /u/turbo/llib/..../erex. The file system treats this as /u/turbo/erex, making this an alias for /u/turbo/llib/REGEREX.

2. JCL and BPXCOPY are used to copy a PDS member into a directory. These facts are known:
   • The name of the PDS member is TABLE1.
   • The directory name is /u/carbon/data.
• Output messages are to be directed to SYSOUT.
• Type of data: text.

```
//TEST JOB MSGLEVEL=(1,1)
//STEP EXEC PGM=BPXCOPY,
//     PARM='ELEMENT(TABLE1) TYPE(TEXT) PATHMODE(0,7,6,4)'
//SYSUT1 DD DSN=CARBON.DATA(TABLE1),DISP=SHR
//SYSUT2 DD PATH="/u/carbon/data"
//SYSTSPRT DD SYSOUT=*  
```

The file /u/carbon/data/TABLE1 is created, with read, write, and execute authority for the user; read and write authority for the group; and read authority for other users.

3. A member of an MVS partitioned data set is copied to a file from a program using the LINK macro. These facts are known:

- The ddname of the source: INDD. INDD can be any sequential data set and is defined by an ALLOCATE command that is issued outside the program.
- The ddname of the directory to copy into: OUTDD. OUTDD can be any directory name and is defined by an ALLOCATE command that is issued outside the program.
- Three link names (DATA, link1, and link2) for the target file.
- Output messages are directed to SYSOUT.
- Type of data: text.

```
* COPYEX CSECT
STM 14,12,12(13) Entry linkage
LR  12,15
USING COPYEX,12
LA  10,SAVEAREA
ST  10,8(13)
ST  13,SAVEAREA+4
LR  13,10
* LINK EP=BPXCOPY,PARAM=(OPT_LIST,DD_LIST),VL
* L 13,SAVEAREA+4 Exit linkage
L  14,12(13)
LM  0,12,20(13)
BR  14
*
SAVEAREA DS 18F
*
OPT_LIST DC H'80' Length of option string
DC CL80 'ELEMENT(DATA) HEAD(''0001''') TYPE(TEXT) X
   LINK('link1','link2')'
*
DD_LIST DC H'72' Length of DDNAME list
DC XL56 '0'
DC CL8 'INDD' Logical SYSUT1 input
DC CL8 'OUTDD' Logical SYSUT2 output directory
*
END COPYEX
```

4. JCL and BPXCOPY are used to copy a file to another file.

```
//TEST JOB MSGLEVEL=(1,1)
//STEP EXEC PGM=BPXCOPY,
//     PARM='ELEMENT(PROGINFO) TYPE(TEXT) PATHMODE(0,7,4,4)'
//SYSUT1 DD PATH="/u/dept/data/proginfo"
//SYSUT2 DD PATH="/u/program"
//SYSTSPRT DD SYSOUT=*  
```

There is no inheritance of file attributes, path mode, links, or symbolic links. This information is determined from the input parameter to BPXCOPY, not from the source file.
Appendix F. Localization

Globalization enables you to work in a cultural context that is comfortable for you through locales, character sets, and a number of special environment variables. The process of adapting an internationalized application or program, particular to a language or cultural milieu, is termed localization.

A locale is the subset of your environment that deals with language and cultural conventions. When specifying a locale, the convention is to use the descriptive locale name. See the section on locale naming conventions in z/OS XL C/C++ Programming Guide. It is made up of a number of categories, each of which is associated with an environment variable and controls a specific aspect of the environment. The following list shows the categories and their spheres of influence:

**LC_COLLATE**
- Collating (sorting) order.

**LC_CTYPE**
- Character classification and case conversion.

**LC_MESSAGES**
- Formats of informative and diagnostic messages and interactive responses.

**LC_MONETARY**
- Monetary formatting.

**LC_NUMERIC**
- Numeric, nonmonetary formatting.

**LC_TIME**
- Date and time formats.

**LC_SYNTAX**
- EBCDIC-variant character encodings used by some C functions and utilities.

To give a locale control over a category, set the corresponding variable to the name of the locale. In addition to the environment variables associated with the categories, there are two other variables which are used in conjunction with localization, LANG and LC_ALL. All of these variables affect the performance of the shell commands. The general effects apply to most commands, but certain commands such as `sort`, with its dependence on LC_COLLATE, require special attention to be paid to one or more of the variables. This section discusses such cases in the Localization topic of the command. The effects of each environment variable is as follows:

**LANG**
- Determines the international language value. Utilities and applications can use the information from the given locale to provide error messages and instructions in that locale's language. If LC_ALL variable is not defined, any undefined variable is treated as though it contained the value of LANG.

**LC_ALL**
- Overrides the value of LANG and the values of any of the other variables starting with LC_.

**LC_COLLATE**
- Identifies the locale that controls the collating (sorting) order of characters and determines the behavior of ranges, equivalence classes, and multicharacter collating elements.

**LC_CTYPE**
- Identifies the locale that defines character classes (for example, alpha, digit, blank) and their behavior (for example, the mapping of lowercase letters to uppercase letters). This locale also determines the interpretation of sequences of bytes as characters (such as single-byte versus double-byte characters).

**LC_MESSAGES**
- Identifies the locale that controls the processing of affirmative and negative responses. This locale also defines the language and cultural conventions used when writing messages.
**Localization**

**LC_MONETARY**
Determines the locale that controls monetary-related numeric formatting (for example, currency symbol, decimal point character, and thousands separator).

**LC_NUMERIC**
Determines the locale that controls numeric formatting (for example, decimal point character and thousands separator).

**LC_TIME**
Identifies the locale that determines the format of time and date strings.

**LC_SYNTAX**
Identifies the locale that defines the encodings for the variant characters in the portable character set.

The NLSPATH localization variable specifies where the message catalogs are to be found.

For example,

```bash
NLSPATH="/system/nlslib/%N.cat"
```

specifies that the z/OS shell is to look for all message catalogs in the directory `/system/nlslib`, where the catalog name is to be constructed from the `name` parameter passed to the z/OS shell with the suffix `.cat`.

Substitution fields consist of a `%` symbol, followed by a single-letter keyword. These keywords are currently defined:

- `%N`
The value of the `name` parameter

- `%L`
The value of the LC_MESSAGES category, or LANG, depending on how the catopen() function that opens this catalog is coded. For more information, see catopen() in z/OS XL C/C++ Runtime Library Reference.

- `%l`
The language element from the LC_MESSAGES category

- `%t`
The territory element from the LC_MESSAGES category

- `%c`
The codeset element from the LC_MESSAGES category

Templates that are defined in NLSPATH are separated by colons (`:`). A leading colon or two adjacent colons (`::`) are equivalent to specifying `%N`. For example:

```bash
NLSPATH="::%N.cat:/nlslib/%L/%N.cat"
```

specifies that the z/OS shell should look for the requested message catalog in `name`, `name.cat`, and `/nlslib/category/name.cat`, where `category` is the value of the LC_MESSAGES or LANG category of the current locale.

Do not set the NLSPATH variable unless you need to override the default system path. Otherwise the commands might behave unpredictably.
Appendix G. Stub commands

z/OS UNIX has several stub commands. *Stub commands* are those commands that are recognized by z/OS UNIX but whose functions are not supported. They are:

- `cancel`
- `cu`
- `lpstat`
Appendix H. File formats

This information gives more detailed information about the formats of the files used by certain shell commands.

**cpio - Format of cpio archives**

**Description**

You can use the cpio command to back up or restore files. The cpio command reads and writes either a compact binary format header or an ASCII format header. The tar command reads and writes headers in either the original TAR format from UNIX systems or the USTAR format defined by the POSIX 1003.2 standard.

The pax command reads and writes headers in any of the cpio formats.

A cpio archive consists of one or more concatenated member files. Each member file contains a header optionally followed by file contents as indicated in the header. The end of the archive is indicated by another header describing an (empty) file named TRAILER!!.

There are two types of cpio archives, differing only in the style of the header:

- ASCII archives have totally printable header information; thus, if the files being archived are also ASCII files, the whole archive is ASCII.
- By default, cpio writes archives with binary headers. However, binary archive files cannot usually be ported to other operating systems, so you should not use these.

The information in an ASCII archive header is stored in fixed-width, octal (base 8) numbers padded with zeros on the left. Table 43 on page 969 gives the order and field width for the information in the ASCII header:

<table>
<thead>
<tr>
<th>Field width</th>
<th>Field name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>magic</td>
<td>Magic number 070707</td>
</tr>
<tr>
<td>6</td>
<td>dev</td>
<td>Device where file resides</td>
</tr>
<tr>
<td>6</td>
<td>ino</td>
<td>I-number of file</td>
</tr>
<tr>
<td>6</td>
<td>mode</td>
<td>File mode</td>
</tr>
<tr>
<td>6</td>
<td>uid</td>
<td>Owner user ID</td>
</tr>
<tr>
<td>6</td>
<td>gid</td>
<td>Owner group ID</td>
</tr>
<tr>
<td>6</td>
<td>nlink</td>
<td>Number of links to file</td>
</tr>
<tr>
<td>6</td>
<td>rdev</td>
<td>Device major/minor for special file</td>
</tr>
<tr>
<td>11</td>
<td>mtime</td>
<td>Modify time of file</td>
</tr>
<tr>
<td>6</td>
<td>namesize</td>
<td>Length of filename</td>
</tr>
<tr>
<td>11</td>
<td>filesize</td>
<td>Length of file</td>
</tr>
</tbody>
</table>

After the header information, namesize bytes of path name are stored. namesize includes the null byte of the end of the path name. After this, filesize bytes of the file contents are recorded.

Binary headers contain the same information in 2-byte (short) and 4-byte (long) integers as follows:
Bytes
Field names
2
  magic
2
dev
2
ino
2
mode
2
uid
2
gid
2
nlink
2
rdev
2
mtime
2
namesize
2
filesize

After the header information comes the filename, with namesize rounded up to the nearest 2-byte boundary. Then the file contents appear as in the ASCII archive. The byte ordering of the 2- and 4-byte integers in the binary format is machine-dependent and thus portability of this format is not easily guaranteed.

Related information
The compress, cpio, pax, and tar commands

---

**magic - Format of the /etc/magic file**

**Description**

The `file` command uses the `/etc/magic` file in its attempt to identify the type of a binary file. Essentially, `/etc/magic` contains templates that show what different types of files look like.

The magic file contains lines that describe magic numbers, which identify particular types of files. Lines beginning with a `>` or `&` character represent continuation lines to a preceding main entry:

- `> 

  If the `file` command finds a match on the main entry line, these additional patterns are checked. Any pattern that matches is used. This might generate additional output; a single blank separates each matching line’s output if any output exists for that line.

  If the `file` command finds a match on the main entry line, and a following continuation line begins with this character, that continuation line’s pattern must also match, or neither line is used. Output text that is associated with any line beginning with the `&` character is ignored.

Each line consists of four fields, which are separated by one or more tabs:
The first field is a byte offset in the file, consisting of an optional offset operator and a value. In continuation lines, the offset immediately follows a continuation character.

If no offset operator is specified, then the offset value indicates an offset from the beginning of the file.

The * offset operator specifies that the value located at the memory location following the operator be used as the offset. Thus, *0x3C indicates that the value contained in 0x3C should be used as the offset.

The + offset operator specifies an incremental offset, based on the value of the last offset. Thus, +15 indicates that the offset value is 15 bytes from the last specified offset.

If the byte offset has passed the file length limit, the test will not match.

The second field is the type of the value.

The valid specifiers are listed as follows:

<table>
<thead>
<tr>
<th>Specifier</th>
<th>_UNIX03=NO</th>
<th>_UNIX03=YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>dC</td>
<td>uC</td>
</tr>
<tr>
<td>short</td>
<td>dS</td>
<td>uS</td>
</tr>
<tr>
<td>long</td>
<td>dL</td>
<td>uL</td>
</tr>
</tbody>
</table>

The C, S, I, or L specifiers are correspond to the number of bytes in the C-language types char, short, int, long. u and d can also be followed by an optional size specifiers listed as follows:

The long format of type specifiers is supported. The following table lists the valid specifiers and their interpretation.
Table 44: Valid specifiers for the long format (magic file) and their interpretation (continued)

<table>
<thead>
<tr>
<th>Specifier</th>
<th>_UNIX03=YES</th>
<th>_UNIX03 is not YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>$</td>
<td>$</td>
</tr>
</tbody>
</table>

(c)

The next field is a value, preceded by an optional operator.

If the specifier from the type field is s or string, then interpret the value as a string. Otherwise, interpret it as a number. If the value is a string, then the test will succeed only when a string value exactly matches the bytes from the file. The string value field can contain at most 127 characters per magic line.

If the value is a string, it can contain the following sequences:

- \character
  The backslash-escape sequences as specified in the Base Definitions volume of IEEE Std 1003.1-2001, Table 5-1, Escape Sequences and Associated Actions (\, \a, \b, \f, \n, \r, \t, \v). In addition, the escape sequence \ (the <backslash> character followed by a <space> character) will be recognized to represent a <space> character.

- \octal
  Octal sequences that can be used to represent characters with specific coded values. An octal sequence consists of a backslash followed by the longest sequence of one, two, or three octal-digit characters (01234567).

By default, any value that is not a string will be interpreted as a signed decimal number. Any such value, with a leading 0x or 0X, will be interpreted as an unsigned hexadecimal number; otherwise, with a leading zero, the value will be interpreted as an unsigned octal number. To maintain compatibility with other systems, numeric values are not subject to bounds checking. Use numeric values that match the specified type.

Operators only apply to nonstring types: byte, short, and long. The default operator is = (exact match). The operators are:

- = Equal.
- ! Not equal.
- > Greater than.
- < Less than.
- & All bits in pattern must match.
- ^ At least one bit in pattern must not match.
- x or ? Any value matches (must be the only character in the field). ? is an extension to traditional implementations of magic.

(d)

The rest of the line is the message string to be printed if the particular file matches the template. Note that the contents of this field is ignored if the line begins with the & continuation character. The fourth field may contain a printf ()-type format indicator to output the magic number (see printf for more details on format indicators). If the field contains a printf ()-type format indicator, the value read from the file will be the argument to printf.
Usage notes

1. Characters from a code page other than IBM-1047 should not be added to the /etc/magic file (the default magic file).

2. Characters from a code page other than IBM-1047 can be used in alternate magic files that are specified by the -m or -M option on the `file` command. These characters should only be used in the third field of the magic file template when the field type is string. They will only match files containing these characters when the `file` command is invoked in the non-IBM-1047 locale.

Examples

Here are some sample entries:

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>short</td>
<td>0x5AD4</td>
<td>DOS executable</td>
</tr>
<tr>
<td>0x18</td>
<td>Short</td>
<td>0x40</td>
<td></td>
</tr>
<tr>
<td>&gt;0x3c</td>
<td>Short</td>
<td>0x6584C</td>
<td>OS/2 linear executable</td>
</tr>
<tr>
<td>&gt;0x3C</td>
<td>Short</td>
<td>0x454e</td>
<td></td>
</tr>
<tr>
<td>&gt;+54byte</td>
<td>1</td>
<td></td>
<td>OS/2 format</td>
</tr>
<tr>
<td>&gt;+54byte</td>
<td>2</td>
<td></td>
<td>Windows format</td>
</tr>
<tr>
<td>0</td>
<td>short</td>
<td>0xFDFO</td>
<td>DOS library</td>
</tr>
<tr>
<td>0</td>
<td>string</td>
<td>AH</td>
<td>Halo bitmapped font file</td>
</tr>
<tr>
<td>0</td>
<td>short</td>
<td>0x601A</td>
<td>Atara ST contiguous executable</td>
</tr>
<tr>
<td>&gt;14 long</td>
<td>&gt;0</td>
<td></td>
<td>- not stripped</td>
</tr>
<tr>
<td>0</td>
<td>byte</td>
<td>0X1F</td>
<td></td>
</tr>
<tr>
<td>&gt;1 byte</td>
<td>0x1E</td>
<td></td>
<td>Packed file</td>
</tr>
<tr>
<td>&gt;1 byte</td>
<td>0x9D</td>
<td></td>
<td>Compressed file</td>
</tr>
</tbody>
</table>

Related information

The `file` command.

pax - Format of pax archives and special header summary files

USTAR archive format

Description

pax uses the USTAR archive format described in the `tar` file format description. For more information about the `tar` file format, see “`tar` - Format of tar archives” on page 978.

An example of the special header summary file:

```
#00
#IBMOS390_USTAR_VERS=1
#
# Archive Name: /tmp/spec.pax
#
# This file was created by the IBM z/OS pax or tar utility.
# During the process of creating the archive from which this
# file was extracted, one or more of the source files to be
# stored in the archive was determined to have names or
```
# attributes that are not supported by the standard USTAR format (as described by POSIX.2 IEEE std 1003.2-1992).
# To preserve these files or these characteristics,
# one or more special header files (having the same name as
# this file) were inserted into the archive. Those files
# are recognized by z/OS pax and tar utilities and are
# used during extraction to restore the files to their
# original state.
## The purpose of this file is to summarize the information
# described by all z/OS special header files stored in
# the archive so that users with versions of pax or tar
# that do not support these special header files can
# manually restore some or all of the files and file
# attributes described by them. Note that some file
# attributes are specific to z/OS and cannot be restored
# on other platforms.
## path name: level0/longsymlink
# Reason: 1. FSUMF076 target of symbolic link
("level1/level2/level3/level4/level5/level6/level7/level8/
level9/level10/level11/level12/level13/level14/linkbase")
exceeds 100 chars.
# Unix restore commands:
  ln -s level1/level2/level3/level4/level5/level6/level7/
  level8/level9/level10/level11/level12/level13/level14/
  linkbase level0/longsymlink
## path name: level0/level1/level2/level3/level4/level5/
level6/level7/level8/level9/level10/level11/level12/
level13/level14/longhardlink
# Reason: 1. FSUMF076 target of hard link
("level0/level1/
level2/level3/level4/level5/level6/level7/level8/level9/
level10/level11/level12/level13/level14/linkbase")
exceeds 100 chars.
# Unix restore commands: 1n level0/level1/level12/level13/
level14/level15/level16/level17/level18/level19/level10/
level11/level12/level13/level14/level15/level16/level17/
level18/level19/level10/level11/level12/level13/level14/
level15/level16/level17/level18/level19/level10/
level11/level12/level13/level14/longhardlink

Portability

POSIX.2, X/Open Portability Guide.

Related information

The cpio, pax, and tar commands.

pax interchange format

Description

A pax archive tape or file that is produced in the -x pax format shall contain a series of blocks. The physical layout of the archive shall be identical to the USTAR format described in “tar - Format of tar archives” on page 978. Each file archived shall be represented by the following sequence:

- An optional header block with extended header records. This header block is of the form that is described in “pax header block” on page 975, with a typeflag value of x or g. The extended header...
records, described in “pax extended header” on page 976, shall be included as the data for this header block.

• A header block that describes the file. Any fields in the preceding optional extended header shall override the associated fields in this header block for this file.

• Zero or more blocks that contain the contents of the file.

At the end of the archive file, there shall be two 512-byte blocks filled with binary zeros, interpreted as an end-of-archive indicator.

A schematic of an example archive with global extended header records and two actual files is shown in the following list. In the list, the second file in the archive has no extended header preceding it, presumably because it does not need extended attributes.

<table>
<thead>
<tr>
<th>Table 45: Schematic of an archive.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header block</strong></td>
</tr>
<tr>
<td>USTAR Header [typeflag=g]</td>
</tr>
<tr>
<td>Global Extended Header Data</td>
</tr>
<tr>
<td>USTAR Header [typeflag=x]</td>
</tr>
<tr>
<td>Extended Header Data</td>
</tr>
<tr>
<td>USTAR Header [typeflag=0]</td>
</tr>
<tr>
<td>Data for File 1</td>
</tr>
<tr>
<td>USTAR Header [typeflag=0]</td>
</tr>
<tr>
<td>Data for File 2</td>
</tr>
<tr>
<td>Block of binary zeros</td>
</tr>
</tbody>
</table>

**pax header block**

**Description**

The pax header block is identical to the USTAR header block described in “tar - Format of tar archives” on page 978, except that two additional typeflag values are defined:

- \textbf{x}
  
  Represents extended header records for the following file in the archive (which shall have its own USTAR header block). The format of these extended header records shall be as described in “pax extended header” on page 976.

- \textbf{g}
  
  Represents global extended header records for the following files in the archive. The format of these extended header records shall be as described in “pax extended header” on page 976. Each value shall affect all subsequent files that do not override that value in their own extended header record and until another global extended header record is reached that provides another value for the same field. The typeflag g global headers should not be used with interchange media that could suffer partial data loss in transporting the archive.

For both of these types, the size field shall be the size of the extended header records in octets. The other fields in the header block are not meaningful to this version of the pax utility. However, if this archive is read by a pax utility conforming to the ISOPOSIX-2:1993 standard, the header block fields are used to create a regular file that contains the extended header records as data. Therefore, header block field values should be selected to provide reasonable file access to this regular file.

A further difference from the USTAR header block is that data blocks for files of typeflag 1 (the digit one) (hard link) might be included, which means that the size field may be greater than zero. Archives created by pax -o linkdata shall include these data blocks with the hard links.
**pax extended header**

**Description**

A pax extended header contains values that are inappropriate for the USTAR header block because of limitations in that format: fields requiring a character encoding other than that described in the ISO/IEC646:1991 standard, fields representing file attributes not described in the USTAR header, and fields whose format or length do not fit the requirements of the USTAR header. The values in an extended header add attributes to the following file (or files; see the description of the typeflag g header block in “pax header block” on page 975) or override values in the following header blocks, as indicated in the list of extended header keywords.

An extended header shall consist of one or more records, each constructed as follows:

```
%d %s=%s\n
```

The extended header records shall be encoded according to the ISO/IEC10646-1:2000 standard (UTF-8). The `<length>` field, `<blank>`, equals sign, and `<newline>` shown shall be limited to the portable character set, as encoded in UTF-8. The `<keyword>` and `<value>` fields can be any UTF-8 characters. The `<length>` field shall be the decimal length of the extended header record in octets, including the trailing `<newline>`.

The field shall be one of the entries from the list in “Extended header keywords” on page 525 or a keyword provided as an implementation extension. Keywords consisting entirely of lowercase letters, digits, and periods are reserved for future standardization. A keyword shall not include an equals sign. In the list of keywords, the notations "files" or "blocks" is used to acknowledge that a keyword affects the following single file after a typeflag x extended header, but possibly multiple files after typeflag g. Any requirements in the list for pax to include a record when in write or copy mode shall apply only when such a record has not already been provided through the use of the `-o` option. When used in copy mode, pax shall behave as if an archive had been created with applicable extended header records and then extracted.

If the `<value>` field is zero length, it shall delete any header block field, previously entered extended header value, or global extended header value of the same name.

If a keyword in an extended header record (or in a `-o` option-argument) overrides or deletes a corresponding field in the USTAR header block, pax shall ignore the contents of that header block field.

Unlike the USTAR header block fields, NULLs shall not delimit `<value>`s; all characters within the `<value>` field shall be considered data for the field. None of the length limitations of the USTAR header block fields in USTAR Header Block shall apply to the extended header records.

---

**queuedefs - Queue description for at, batch, and cron**

**Description of the dequeue file**

The queuedefs file describes the characteristics of the queues managed by the clock daemon `cron`. Each line in the file that is not a comment uses the following format to describe a queue:

```
q . [njob] [nicen] [nwaitw]
```

where the fields are:

- **q**
  
  Specifies the name of the queue. Jobs started by `at` default to queue `a`; jobs started by `batch` default to queue `b`, and `crontab` files default to queue `c`. Queue names can be any single-byte character except a space, tab, newline, null, or number sign (`#`).

- **njob**
  
  Specifies the maximum number of jobs that can be run in the queue simultaneously. If more than `njob` jobs are ready to run, `cron` runs the first `njob` jobs immediately, and runs the others as current jobs terminate. The default value is 100.
nice

Specifies the nice value (see nice) that cron assigns to all jobs in the queue that are not run by a user ID with appropriate privileges. The default value is 2.

nwait

Specifies the number of seconds that cron is to wait before it reschedules a job that was deferred because there were more than njob jobs running in that job's queue, or because more than 25 jobs were running in all queues. The default value is 60.

Lines beginning with a number sign (#) are comments, and are ignored.

Following is a sample queuedefs file:

```
# Sample queuedefs file
#
a.5j3n
b.3j1n90w
```

It indicates that the a queue, for at jobs, can have a maximum of five jobs running simultaneously. crontab runs the jobs with a nice value of 3. Because there is no nwait field for this queue, if cron cannot run a job because too many other jobs are running, it waits 60 seconds before trying to run it again.

This file also states that the b queue, for batch jobs, can have a maximum of three jobs running simultaneously. cron runs the jobs with a nice value of 1. If cron cannot run a job because too many other jobs are running, it waits 90 seconds before trying to run it again. All other queues can run up to 100 jobs simultaneously; cron runs these jobs with a nice value of 2 and, if it cannot run a job because too many other jobs are running, it waits 60 seconds before trying to run it again.

Related information for queuedefs

The at, batch, and crontab commands.

---

**tags - Format of the tags file**

**Description**

When you use the vi :tag or ex :tag command, or the ex –t, more –5, vi –t, option, that utility looks for a file called tags in the current directory. This lets you quickly locate various points of interest in a C program which can span more than one source file. These points of interest are tags.

The tags file contains tags for function definitions, preprocessor macro definitions, and typedef definitions.

For each tag, the tags file contains one line in the following form:

```
tagname  sourcefile  address
```

The tagname field is the name of the C function, macro, or typedef. The sourcefile field has the name of the source file containing the tag named tagname. The address field is an editor address within sourcefile to reach the tag definition. This is either a line number in the file or a regular expression (enclosed in ? or / characters) that uniquely matches the line of source code where the tag appears. A tab character separates each field.

For vi or more to use the tags file correctly, it must be sorted by tagname using the POSIX locale's collation sequence.

**Related information**

The more, sort, and vi commands.
tar - Format of tar archives

Description

tar reads and writes headers in either the original TAR format from UNIX systems or the USTAR format that is defined by the POSIX 1003.1 standard.

The pax command reads and writes headers in any of the tar formats.

The tar command supports both the older UNIX-compatible tar formats and the extended USTAR format. The -X option needs to be used to enable extended USTAR format. The extended USTAR format allows more information to be stored and supports longer path names. There is also a non-portable OS390 format (-S option) which also allows storing of additional file attributes and longer path names.

A tar archive, in either format, consists of one or more blocks, which are used to represent member files. Each block is 512 bytes long; you can use the -b option with tar to indicate how many of these blocks are read or written (or both) at once.

Each member file consists of a header block, followed by zero or more blocks containing the file contents. The end of the archive is indicated by two blocks that are filled with binary zeros. Unused space in the header is left as binary zeros.

The header information in a block is stored in a printable ASCII form, so that tar archives are easily ported to different environments. If the contents of the files on the archive are all ASCII, the entire archive is ASCII.

Table 46 on page 978 shows the UNIX format of the header block for a file:

Table 46: Archive file: UNIX-compatible format

<table>
<thead>
<tr>
<th>Field width</th>
<th>Field Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>name</td>
<td>Name of file</td>
</tr>
<tr>
<td>8</td>
<td>mode</td>
<td>File mode</td>
</tr>
<tr>
<td>8</td>
<td>uid</td>
<td>Owner user ID</td>
</tr>
<tr>
<td>8</td>
<td>gid</td>
<td>Owner group ID</td>
</tr>
<tr>
<td>12</td>
<td>size</td>
<td>Length of file in bytes</td>
</tr>
<tr>
<td>12</td>
<td>mtime</td>
<td>Modify time of file</td>
</tr>
<tr>
<td>8</td>
<td>chksum</td>
<td>Checksum for header</td>
</tr>
<tr>
<td>1</td>
<td>link</td>
<td>Indicator for links</td>
</tr>
<tr>
<td>100</td>
<td>linkname</td>
<td>Name of linked file</td>
</tr>
</tbody>
</table>

- A directory is indicated by a trailing / (slash) in its name.
- The link field is: 1 for a linked file, 2 for a symbolic link, 0 otherwise.

Tar determines that the USTAR format is being used by the presence of the null-terminated string USTAR in the magic field. All fields before the magic field correspond to those of the UNIX format, except that typeflag replaces the link field.

Table 47: Archive file: USTAR format

<table>
<thead>
<tr>
<th>Field width</th>
<th>Field name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>name</td>
<td>Name of file</td>
</tr>
<tr>
<td>8</td>
<td>mode</td>
<td>File mode</td>
</tr>
</tbody>
</table>
### Description of the header files

In the headers:

- The name field contains the name of the archived file. On USTAR format archives, the value of the prefix field, if non-null, is prefixed to the name field to allow names longer than 100 characters.
- The magic, uname, and gname fields are null-terminated character strings.
- The name, linkname, and prefix fields are null-terminated unless the full field is used to store a name (that is, the last character is not null).
- All other fields are zero-filled octal numbers, in ASCII. Trailing nulls are present for these numbers, except for the size, mtime, and version fields.
- prefix is null unless the file name exceeds 100 characters.
- The size field is zero if the header describes a link.
- The chksum field is a checksum of all the bytes in the header, assuming that the chksum field itself is all blanks.
- For USTAR, the typeflag field is a compatible extension of the link field of the older tar format. The following values are recognized:

<table>
<thead>
<tr>
<th>Flag</th>
<th>File type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or null</td>
<td>Regular file</td>
</tr>
<tr>
<td>1</td>
<td>Link to another file already archived</td>
</tr>
<tr>
<td>2</td>
<td>Symbolic link</td>
</tr>
<tr>
<td>3</td>
<td>Character special file</td>
</tr>
<tr>
<td>4</td>
<td>Block special file (not supported)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field width</th>
<th>Field name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>uid</td>
<td>Owner user ID</td>
</tr>
<tr>
<td>8</td>
<td>gid</td>
<td>Owner group ID</td>
</tr>
<tr>
<td>12</td>
<td>size</td>
<td>Length of file in bytes</td>
</tr>
<tr>
<td>12</td>
<td>mtime</td>
<td>Modify time of file</td>
</tr>
<tr>
<td>8</td>
<td>chksum</td>
<td>Checksum for header</td>
</tr>
<tr>
<td>1</td>
<td>typeflag</td>
<td>Type of file</td>
</tr>
<tr>
<td>100</td>
<td>linkname</td>
<td>Name of linked file</td>
</tr>
<tr>
<td>6</td>
<td>magic</td>
<td>USTAR indicator</td>
</tr>
<tr>
<td>2</td>
<td>version</td>
<td>USTAR version</td>
</tr>
<tr>
<td>32</td>
<td>uname</td>
<td>Owner user name</td>
</tr>
<tr>
<td>32</td>
<td>gname</td>
<td>Owner group name</td>
</tr>
<tr>
<td>8</td>
<td>devmajor</td>
<td>Device major number</td>
</tr>
<tr>
<td>8</td>
<td>devminor</td>
<td>Device minor number</td>
</tr>
<tr>
<td>155</td>
<td>prefix</td>
<td>Prefix for file name</td>
</tr>
</tbody>
</table>
utmpx

Description
Login accounting information is stored in two files:

- /etc/utmpx holds the current state of each item being accounted
- /etc/wtmp maintains the history of changes to each accounting item

Both files are arrays of the following binary records described in the form of a C data structure:

```
#include <sys/types.h>

struct utmpx
{
    char  ut_user[9] ;       /* user login name */
    char  ut_id[34] ;       /* unspecified initialization process ID */
    char  ut_line[33] ;     /* device name */
    pid_t ut_pid ;          /* process id */
    short int ut_type ;     /* type of entry */
    short int ut_version;   /* LE runtime level when boot record is written */
    #ifndef _LP64
    struct timeval     ut_tv;   /* time entry was made */
    #else
    struct __timeval32 ut_tv32; /* time entry was made */
    #endif

    #ifdef __LP64
    struct timeval   ut_tv;    /* time entry was made */
    #else
    struct __timeval32 ut_tv32; /* time entry was made */
    #endif
};
```

Related information
The cpio and tar commands

utmpx - Format of login accounting files

• In USTAR format, the uname and gname fields contain the name of the owner and group of the file, respectively.

Compressed tar archives are equivalent to the corresponding archive being passed to a 14-bit compress command.

z/OS extended USTAR special header summary (S and T are z/OS extensions. See “z/OS-extended USTAR support” on page 532 for more information.)

z/OS extended USTAR special header

A-Z
Available for custom usage

z/OS: UNIX System Services Command Reference
/*
 * #endif
 * struct ut_exit_status
 * {
 *     short ut_e_termination;       /* Process termination status
 *     short ut_e_exit ;            /* Process exit status
 * }
 * ut_exit ;                       /* The exit status of process marked DEAD_PROCESS. 
 * unsigned short ut_reserved1;    /* Reserved for future use 
 * char ut_host[1024] ;            /* host name, if remote
 * #ifdef
 * _LP64
 *     struct timeval ut_tv;   /* time entry was made
 * #else
 *     struct __timeval64 ut_tv64; /* time entry was made
 * #endif
 * }
 * #define EMPTY         0   /* Unused */
 * #define RUN_LVL       1   /* Set new run level */
 * #define BOOT_TIME     2   /* System boot */
 * #define OLD_TIME      3   /* Time of date change - delta */
 * #define NEW_TIME      4   /* Time of date change + delta */
 * #define INIT_PROCESS  5   /* Process started by &[/ETCDIR]/init */
 * #define LOGIN_PROCESS 6   /* Login process */
 * #define USER_PROCESS  7   /* User process */
 * #define DEAD_PROCESS  8   /* Contains exit status */
 * #define ACCOUNTING    9   /* Other accounting */
 *
 * Files
 * 
 * /etc/utmpx
 *  Reflects the current state of the accounting entries; for example, who is logged in, when the date was
 *  last set, and so on.
 * 
 * /etc/wtmp
 *  Contains a history of changes to any of the accounting entries.
 *
 * Related information
 * The who command
 *
 * uucp - Format of UUCP working files
 *
 * Description
 * UUCP uses three kinds of working files when handling UUCP requests, command, data, and execute.
 * All three files are stored in a subdirectory for each specific site, named after the site's name. For example,
 * because the UUCP spool directory is /usr/spool/uucp, then the directory /usr/spool/uucp/south is
 * used for all the command, data, and execute files that are associated with the remote site south.
**Command Files**

Command files are created by the mail routing agents uucp and uux. On UUCP sites, command files have names such as `C.targetA28B9`, where target is the name of the destination site, A is the job grade (as set by the `-g` option to uucp, and 28B9 is the sequence number or job identification number. (You can use the `-j` option on uucp and uux, as well as uustat to find the job identification number.)

In a command file, each line records one file transfer request. The fields are defined as follows:

**type**
- The type field can be one of the following:
  - **R**: Receive a file from remote to local site.
  - **S**: Send a file from local to remote site.

**source**
- The name of the source file.

**destination**
- The name of the file after the transfer completes, whether to the remote site (S request) or the local site (R request).
- Special characters such as the tilde (~) are still present, because they are expanded on the destination site.

**sender**
- The login name of the user who issued the command. This is normally your login name, though some programs (such as mail programs) use a different login name for their requests.

**options**
- The command options, which correspond to options of the uucp and uux commands.
  - **C**: Use the data file name as the source for the copy; this can only be used with the S request.
  - **c**: Use the source file name as the source for the copy.
  - **d**: Create intermediate target directories as required. This is the default.
  - **m**: Send mail to the user when the transfer is complete.
  - **n**: Send mail to the user specified by the notification name when the transfer is complete.

**datafile**
- The temporary file to be used if the source file was copied into the spool directory; it is only used with the S request. If C is one of the options, the data file is the name of the copy in the destination site's data spool directory. Otherwise, the placeholder name D.0 is used.

**file mode**
- The UNIX-style permission mode of the source file. It is only used with the S request. All files sent have mode 0666, plus whatever execute permissions the original file had. (For an explanation of the modes, see chmod.)

**notification**
- The login name of the person to be notified after the job request completes. It is used only with the S request if n is one of the options.

**Examples**

1. The command

```bash
grep -m /memos.001 /memos.002 south!~/
```
copies the files /memos.001 and /memos.002 root directory to the public UUCP directory on south. Assuming your user name is eve, a command file containing these lines is created in the UUCP spool directory /usr/spool/uucp/south:

```
S /memos.001 ~/memos.001 eve -mcd D.0 0777
S /memos.002 ~/memos.002 eve -mcd D.0 0777
```

2. The command

```
uucp south!~/index ~/
```

generates a command file on your site in the UUCP spool directory /usr/spool/uucp/south containing this line:

```
R ~ /index ~/index eve -cd
```

**Data files**

Data files contain data to be transferred to the remote site. They are created by uucp if the –C option is used, and by uux and mail programs.

On UUCP sites, data files have names like D. source9873001, where source is the name of the site that the data file originated from (the local site for an S request, or the remote site for an R request), 98B3 is the sequence number, and 001 is the subsequence number, used when a request generates more than one data file.

Data files created by uucp contain files to be copied. Data files created by uux which contain commands for the remote site become execute files at their destination.

Mail sites typically create two data files, one containing the message and the other containing the command to run the mail routing agent on the remote site.

**Examples**

UUCP data files contain data to be copied. The contents of uux data files and commands that generate remote commands are execute files intended for other sites. For example, a mail message to north generates two data files in the UUCP spool directory /usr/spool/uucp/north

```
D.north000A001
X.northX000A002
```

These working files are created:

```
D.north000A001  Text of mail message
X.northX000A002  Execute file
```

The execute file contains the uux request for the mail routing program to be run on north.

**Execute files**

*Execute files* are data files containing commands that are created on other sites and copied to your site. The files are treated as execute files when they arrive at your site, where the commands are run by uuxqt.

On UUCP sites, execute files are named as:

```
X.remotX28A3003
```

where remot is the first five characters of the destination site's name, X is the job grade (execute files always have the grade X), and 28A3 is the sequence number.

Each execute file contains one command, and the necessary information to run the command. The type of information on each line is identified by the first character in the line. Not all lines are used in all files, and
not all UUCP implementations support all of these lines. The first line in an execute file must be a U line, and the last line must be a C line.

# Indicates a comment. Comments and unrecognized commands are ignored.

C command
Requests that command be run. command is a string that includes the program and arguments. This line must be present and must be the last line in the execute file.

E Processes the command with execve(). If the E line is present, uuxqt runs a fork()/exec() sequence, unless the command contains a shell metacharacter. In that case, uuxqt invokes a shell to run the command.

e Processes the command by the POSIX shell. It is intended to handle commands that require special processing. If the e line is present, uuxqt invokes the defined shell to run the command.

F filename [xqtnam...
This command originated with user eve on north. It requests that \texttt{rmail} be run with the argument \texttt{bob} on the target site. The file \texttt{D.south49Z3} is required to run the command and is used as standard input for the command.

\textbf{Portability}

X/Open Portability Guide.

\textbf{Related information}

\texttt{uucico}, \texttt{uucp}, \texttt{uux}, \texttt{uuxqt}
uucp
Appendix I. Format of the TZ environment variable

Command format

Format
TZ= standardHH[:MM[:SS]] [daylight[HH[:MM[:SS:]]] [,startdate[/starttime],enddate[/endtime]] ]

Description
All commands assume that times stored in the file system and returned by the operating system are stored using Universal Time Coordinated (UTC), hereafter referred to as the universal reference time. The mapping from the universal reference time to local time is specified by the TZ (time zone) environment variable.

The value of the TZ environment variable has these fields (two required and three optional):

- **standard**
  An alphabetic abbreviation for the local standard time zone; for example, GMT, EST, MSEZ.

- **HH[:MM[:SS]]**
  The time offset westward from the universal reference time. A leading minus sign (-) means that the local time zone is east of the universal reference time. An offset of this form must follow standard and can also optionally follow daylight. An optional colon (:) separates hours from optional minutes and seconds.

  If daylight is specified without a daylight offset, daylight saving time is assumed to be one hour ahead of the standard time.

- **[daylight]**
  The abbreviation for your local daylight saving time zone. If the daylight field is missing, the conversion to daylight saving time is disabled. The number of hours, minutes, and seconds your local daylight saving time is offset from UTC when daylight saving time is in effect. If the daylight saving time abbreviation is specified, and the offset omitted, the offset of one hour is assumed.

- **[,startdate[/starttime],enddate[/endtime]]**
  A rule that identifies the start and end of daylight saving time, specifying when it should be in effect. Both the startdate and enddate must be present, and must either take the form Jn, n, or Mm.w.d.:

  - Jn is the Julian day n (1 <= n <= 365). Leap days are not counted. In all years, including leap years, February 28 is day 59 and March 9 is day 60. It is impossible to implicitly refer to the occasional February 29.

  - n is the zero-based Julian day (0 <= n <= 365). Leap days are counted, and it is possible to refer to February 29.

  For non-leap years:
  - January 1 (Day 0)
  - February 28 (Day 58)
  - March 1 (Day 59)
  - December 1 (Day 364)

  For leap years:
  - January 1 (Day 0)
  - February 28 (Day 59)
TZ environment variable

- March 1 (Day 60)
- December 1 (Day 365)

- $Mm.w.d$ defines the day ($0 \leq d \leq 6$) of week $w$ ($1 \leq w \leq 5$) of month $m$ ($1 \leq m \leq 12$) of the year.
  Week 5 has the last day ($d$) in month $m$, which may occur in either the fourth or fifth week. Week 1 is the first week in which the $d$th day occurs. Day zero is Sunday.

Neither $starttime$ nor $endtime$ are required. If they are omitted, their values default to 02:00:00. If this daylight saving time rule is omitted altogether, the values in the rule default to the standard rules for American daylight saving time.

When the TZ variable is not set, time conversions behave as if TZ were set to TZ=GMT0.

**Portability**

This interpretation of the TZ environment variable is a superset of that supported by UNIX System V.

**Reference**

The locale, date and touch commands.
Appendix J. Environment variables

This information contains a partial list of environment variables.

- For the c89/cc/c++ environment variables, refer to the c89/cc/c++ command.
- For the xlc environment variables, refer to the “xlc: Compiler invocation using a customizable configuration file” on page 847 command.
- For the mailx environment variables, refer to the “mailx - Send or receive electronic mail” on page 409 command.
- For the tcsh environment variables, refer to the “tcsh: Invoke a C shell” on page 671 command.
- For the tso environment variables, refer to the “tso - Run a TSO/E command from the shell” on page 752 command.
- For the tsocmd environment variables, refer to the tsocmd command.
- For the vi environment variables, refer to the “vi - Use the display-oriented interactive text editor” on page 807 command.

For a list of built-in environment variables, refer to Table 29 on page 606. (Built-in environment variables are predefined variables that are set up with default values when you start the shell.)

A list of Commonly used environment variables is in z/OS UNIX System Services Planning.
Environment Variables
Several utilities allow the user to specify an MVS data set name in place of a z/OS UNIX file name. See “Utilities that support MVS data set names” on page 992 for the current list. This topic describes the syntax for specifying an MVS data set name. Because MVS data set names generally contain single quotation marks and parentheses, which can be misinterpreted by the shell, care needs to be taken to correctly escape these characters.

What follows are general rules for specifying MVS data set names. Consult the description of each utility for more specific instructions or exceptions.

- MVS data sets are distinguished from z/OS UNIX files by preceding them with two slashes (/ /). For example, to specify the MVS data set name PROGRAM.OUTPUT, enter:

  //PROGRAM.OUTPUT

  If the double slashes were not used, the name would be interpreted as the file path name PROGRAM.OUTPUT in the current working directory.

- Unless a utility specifically provides an option to disable uppercasing, the default approach is to make all MVS data set names uppercase before processing. For example, the following are all equivalent methods for specifying the MVS data set PROGRAM.OUTPUT:

  //program.output
  //ProGram.OutPut
  //PROGRAM.OUTPUT

- The single quotation mark (‘) and parentheses (() ) metacharacters are typically used to specify fully qualified MVS names and PDS/PDSEs, respectively. These characters, however, are metacharacters that will be incorrectly interpreted by the shell. To prevent this situation, they must be escaped. The simplest approach is to place the entire name within double quotation marks (" ). Alternatively, these characters can be escaped by preceding each with a backslash (\). For example:

  To specify the fully qualified MVS data set 'SMITH.PROGRAM.OUTPUT':

  "//smith.program.output"
  //"smith.program.output"

  To specify the fully qualified partitioned data set 'SMITH.PROGRAM.SOURCE(FILE1)):

  "//smith.program.source(file1)"
  //"smith.program.source\{file1\}\"

  To specify the non-qualified partitioned data set PROGRAM.SOURCE(FILE1):

  "\program.source(file1)"
  //smith.program.source\{file1\}

Restrictions:

- Shell commands that support MVS data sets (such as cp, mv, pax, tar, and c89) cannot process data sets that are defined as DSNTYPE=LARGE due to the restriction imposed by fopen().

- In order to use the //DD:DDNAME format, the dataset must be allocated in the address space that the following command will run in. However, in the login shell environment, the user cannot use the //DD:DDNAME format due to the restriction that the address space of the allocated dataset is not always the same as the running command.
Utilities that support MVS data set names

The following utilities support the use of MVS file names. Consult the description for each utility for limitations and exceptions:

• automount
• c89
• cp
• mv
• pax
• tar

MVS data sets that are defined with DSNTYPE=LARGE are not supported.
Appendix L. Controlling text conversion for z/OS UNIX shell commands

This information describes the various methods for controlling text conversion for the z/OS UNIX shell and utilities.

Using automatic code set conversion

Most commands that perform file input and output allow automatic code set conversion of files that are tagged as text with a code set. For example, a file with ISO8859-1 (ASCII) content that is tagged with TXT, ISO8859-1 can be converted to IBM-1047 (EBCDIC) for processing by z/OS UNIX shells and utilities. Automatic conversion is controlled by configuration parameters and environment variables. For more information about automatic conversion, see z/OS UNIX System Services Planning.

Restriction: For commands that can specify text conversion (see “Specifying the text conversion” on page 994, automatic conversion can take place between IBM-1047 and any code sets that Unicode Service supports. For sh, automatic conversion is supported with the limitation of the SBCS code pages of the locale and SBCS shell scripts. For tsch, automatic conversion is supported with the limitation of the SBCS code page of the locale. For other commands, automatic conversion can only be supported between the IBM-1047 and ISO8859-1 code sets.

Shell redirection and automatic conversion

In the z/OS shell or the tcsh shell, when the shell is redirecting the standard input (stdin), standard output (stdout), or standard error (stderr), the default behavior is no automatic conversion of tagged files and no tagging of files created by the redirection. However, shell variables can be used to control the automatic conversion. For more information about shell redirection and automatic conversion, see Shell variables (shell variables for automatic conversion) and “tcsh shell and environment variables” on page 697 (using tcsh shell variables to control automatic conversion).

You can also use shell variables for commands in a pipeline. For example, they can be used to tag the standard output of each command that is writing to a pipeline or to tag the standard input of each command that is reading from a pipeline.

Disabling automatic conversion

Many commands that perform file input/output, by default, allow automatic conversion of tagged files. Some of these commands can disable automatic conversion using the -B option. The explicit use of the -B option on the command overrides any automatic conversion as well as any text conversion indicated by the _TEXT_CONV environment variable.

The -B option applies to the input text file that is processed by the command. Other files used by the command, such as a file list or configuration information, are not affected by the -B option. When a command allows the use of the standard input (stdin) in place of the input text file, the -B option will disable automatic conversion of the standard input (stdin) if it is not associated with a terminal.

Following are commands can be used to disable automatic conversion using the -B option.

<table>
<thead>
<tr>
<th>cat</th>
<th>grep</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmp</td>
<td>head</td>
</tr>
<tr>
<td>comm</td>
<td>more</td>
</tr>
</tbody>
</table>
Some commands that perform file input/output and, by default, allow automatic conversion of tagged files, allow the automatic code set conversion of files that are tagged as text with a code set. These commands use the filecodeset and pgmcodeset option (-W option to specify the text conversion. The explicit use of the filecodeset or pgmcodeset option (-W option on the command will override any automatic conversion, any use of the -B option to disable automatic conversion, as well as any text conversion indicated by the _TEXT_CONV environment variable.

The filecodeset and pgmcodeset option (-W option) applies to the input text file that is processed by the command. Other files that are used by the command, such as a file list or configuration information, are not affected by the filecodeset and pgmcodeset option (-W option). When a command allows the use of the standard input (stdin) in place of the input text file, the filecodeset and pgmcodeset option (-W option) will be applied to the standard input (stdin) if it is not associated with a terminal.

The coded character set can be specified by using the code set name or by using the numeric coded character set identifier (CCSID). The command iconv -l lists existing numeric CCSIDs along with their corresponding code set names.

**Note:** BPXK_UNICODE_TECHNIQUE, _BPXK_UNICODE_SUB and _BPXK_UNICODE_MAL specifies the Unicode Services actions to take for the translation operation. For more information about those environment variables, see z/OS UNIX System Services Planning.

**Note:** Text conversion can take place between pgmcodeset IBM-1047 and any filecodeset that Unicode Service supports.

Following are commands that can specify text conversion using the filecodeset or pgmcodeset option (-W option).

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>cut</td>
<td>pack</td>
</tr>
<tr>
<td>diff</td>
<td>paste</td>
</tr>
<tr>
<td>dircmp</td>
<td>sed</td>
</tr>
<tr>
<td>ed</td>
<td>strings</td>
</tr>
<tr>
<td>egrep</td>
<td>tail</td>
</tr>
<tr>
<td>ex</td>
<td>unexpand</td>
</tr>
<tr>
<td>expand</td>
<td>uniq</td>
</tr>
<tr>
<td>fgrep</td>
<td>vi</td>
</tr>
<tr>
<td>file</td>
<td>wc</td>
</tr>
</tbody>
</table>

---

**Specifying the text conversion**

<table>
<thead>
<tr>
<th>cut</th>
<th>pack</th>
</tr>
</thead>
<tbody>
<tr>
<td>diff</td>
<td>paste</td>
</tr>
<tr>
<td>dircmp</td>
<td>sed</td>
</tr>
<tr>
<td>ed</td>
<td>strings</td>
</tr>
<tr>
<td>egrep</td>
<td>tail</td>
</tr>
<tr>
<td>ex</td>
<td>unexpand</td>
</tr>
<tr>
<td>expand</td>
<td>uniq</td>
</tr>
<tr>
<td>fgrep</td>
<td>vi</td>
</tr>
<tr>
<td>file</td>
<td>wc</td>
</tr>
</tbody>
</table>
Using the _TEXT_CONV environment variable

Use the _TEXT_CONV environment variable if you know that all input text files for a given set of commands require the same text conversion. It applies to commands that support the filecodeset and pgmcodeset option (-W option) and the -B option (disable automatic conversion of tagged files).

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Description</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>_TEXT_CONV</td>
<td>Contains text conversion information for commands that support a text conversion specification. Multiple value keywords are separated by a comma. Additional information for value keywords that require it are placed within parentheses immediately after the value. For example: FILECODESET(ISO8859-1)</td>
<td>FILECODESET, PGMCODESET, DISABLE</td>
</tr>
</tbody>
</table>

Use the _TEXT_CONV environment variable if it is not possible to use the filecodeset and pgmcodeset option (-W option) and the -B option (disable automatic conversion of tagged files). If any of those options are specified on a command, the specified command option is used to determine the text conversion. If none of those options are specified on the command, the _TEXT_CONV environment variable is used to determine the text conversion. If none of those options are specified and the environment variable is not set, then automatic code set conversion occurs, if enabled.

The valid value keywords for the _TEXT_CONV environment variable correspond to the filecodeset and pgmcodeset option (-W option) and the -B option (disable automatic conversion of tagged files). When a value keyword is included in the _TEXT_CONV environment variable, the behavior of the command will be as if the corresponding option was specified on the command. The corresponding value keywords are as follows:

<table>
<thead>
<tr>
<th>Command option</th>
<th>Corresponding _TEXT_CONV value keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>-W filecodeset</td>
<td>FILECODESET</td>
</tr>
<tr>
<td>-W pgmcodeset</td>
<td>PGMCODESET</td>
</tr>
<tr>
<td>-B</td>
<td>DISABLE</td>
</tr>
</tbody>
</table>

The FILECODESET and PGMCODESET value keywords require a supported coded character set to be specified within parenthesis. Multiple value keywords are separated with a comma. If the DISABLE value keyword is used along with either the FILECODESET or PGMCODESET value keywords, the DISABLE value keyword is ignored.

If the specified coded character set for the FILECODESET or PGMCODESET value keyword is not valid, an error condition will occur for all commands that support the filecodeset and pgmcodeset option (-W option), as if the invalid code set were specified on the command.

If the _TEXT_CONV environment variable is used by a command and it contains a value keyword other than FILECODESET, PGMCODESET, or DISABLE, an error message will be issued and the command ends. If the _TEXT_CONV environment variable is used by a command and it contains a syntax error, an error message will be issued and the command ends.
Some examples of specifying text conversion using the _TEXT_CONV environment variable:

1. To indicate text conversion from the ASCII code set ISO8859-1 to the EBCDIC code set IBM-1047 using the _TEXT_CONV environment variable:
   
   ```bash
   export _TEXT_CONV="FILECODESET(ISO8859-1),PGMCODESET(IBM-1047)"
   ```

2. To indicate text conversion from the ASCII code set 819, but allow the default value for PGMCODESET using the _TEXT_CONV environment variable:
   
   ```bash
   export _TEXT_CONV="FILECODESET(819)"
   ```

3. To use the _TEXT_CONV environment variable to disable automatic conversion:
   
   ```bash
   export _TEXT_CONV="DISABLE"
   ```

**Note:** When exporting the _TEXT_CONV environment variable, it is very important to understand that all commands that support the corresponding options performs the requested text conversion, regardless of the input text file that is being used. For example, if automatic conversion is enabled and the input file is a tagged file, the automatic conversion is ignored and the text conversion indicated by the _TEXT_CONV environment variable is used. However, if a command specifies any of the corresponding options, that option will override the _TEXT_CONV environment variable.

The coded character set can be specified by using the code set name or by using the numeric coded character set identifier (CCSID). The command `iconv -l` lists existing numeric CCSIDs along with their corresponding code set names.

**Note:** `_BPXK_UNICODE_TECHNIQUE`, `_BPXK_UNICODE_SUB` and `_BPXK_UNICODE_MAL` specify the Unicode Services actions to take for the translation operation. For more information about those environment variables, see [z/OS UNIX System Services Planning](https://www.ibm.com/systems/ux/zos/unixsystemservices/planning).

**Restriction:** Text conversion can take place between pgmcode set IBM-1047 and any filecode set that Unicode Service supports.

Following are commands that are affected by the FILECODESET and PGMCODESET value keywords of the _TEXT_CONV environment variable.

<table>
<thead>
<tr>
<th>cat</th>
<th>grep</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmp</td>
<td>head</td>
</tr>
<tr>
<td>comm</td>
<td>more</td>
</tr>
<tr>
<td>cut</td>
<td>paste</td>
</tr>
<tr>
<td>diff</td>
<td>sed</td>
</tr>
<tr>
<td>dircmp</td>
<td>strings</td>
</tr>
<tr>
<td>ed</td>
<td>tail</td>
</tr>
<tr>
<td>egrep</td>
<td>unexpand</td>
</tr>
<tr>
<td>ex</td>
<td>uniq</td>
</tr>
<tr>
<td>expand</td>
<td>vi</td>
</tr>
<tr>
<td>fgrep</td>
<td>wc</td>
</tr>
<tr>
<td>file</td>
<td></td>
</tr>
</tbody>
</table>

Following are commands that are affected by the DISABLE value keyword of the _TEXT_CONV environment variable.

<table>
<thead>
<tr>
<th>cat</th>
<th>grep</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmp</td>
<td>head</td>
</tr>
</tbody>
</table>
### Commands that prevent automatic conversion by default

Table 48 on page 997 lists commands that expect binary data, so they prevent automatic conversion.

<table>
<thead>
<tr>
<th>Command</th>
<th>Special behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>comm</td>
<td>more</td>
</tr>
<tr>
<td>cut</td>
<td>pack</td>
</tr>
<tr>
<td>diff</td>
<td>paste</td>
</tr>
<tr>
<td>dircmp</td>
<td>sed</td>
</tr>
<tr>
<td>ed</td>
<td>strings</td>
</tr>
<tr>
<td>egrep</td>
<td>tail</td>
</tr>
<tr>
<td>ex</td>
<td>unexpand</td>
</tr>
<tr>
<td>expand</td>
<td>uniq</td>
</tr>
<tr>
<td>fgrep</td>
<td>vi</td>
</tr>
<tr>
<td>file</td>
<td>wc</td>
</tr>
</tbody>
</table>

**cksum**
- Allows automatic conversion with `-T`.

**compress**
- Allows automatic conversion on the file being read.
- Disables automatic conversion on the compressed file and allows the automatic tagging of the file as binary.

Because file tag information cannot be preserved in the compressed file, you can lose data if translation does not occur on input. If you want translation to occur, change the file tag, or disable automatic conversion with the `_BPXK_AUTOCVT` environment variable.

**dd**
- Prevents automatic conversion.
- If you specify `conv=ascii`, `conv=ebcdic`, or `conv=ibm`, and the input is tagged as text, dd issues a warning message if the file tag does not match the expected output.

**gencat**
- Prevents automatic conversion.

**mkcatdefs**
- Prevents automatic conversion.

**od**
- Allows automatic conversion with `-T`.

**sum**
- Allows automatic conversion with `-T`.

**uncompress** and **zcat**
- Disables conversion on the compressed file being read.
- Allows conversion on the uncompressed file being written.

Because the compressed file must always be binary, IBM recommends that you do not provide the option to allow translation of a compressed file on input. However, if you need to do this, you must perform it manually using `iconv`.

**unpack** and **pcat**
- Prevents automatic conversion.

**uudecode**
- Prevents automatic conversion.
<table>
<thead>
<tr>
<th>Command</th>
<th>Special behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuencode</td>
<td>Prevents automatic conversion</td>
</tr>
</tbody>
</table>

Table 48: Commands that disallow automatic conversion by default (continued)
Appendix M. Additional dbx documentation

The following topics are dbx help texts that are not particular to a specific command. They can be viewed within dbx by using the help command. You can also view them as man pages, using the traditional man syntax, with "dbx" and the topic title in the place of a command name. For example, man dbxexecution would display the execution page. Within dbx, help execution would display the same page.

execution: Controlling execution

The dbx utility allows you to set breakpoints (stopping places) in the target program. After entering dbx, you can specify which lines or addresses are to be breakpoints and then run the program with dbx. When the program reaches a breakpoint, it halts and reports that it has reached a breakpoint. You can then use dbx subcommands to examine the state of your program.

For execution controlling commands, see: run, rerun, stop, status, catch, ignore, cont, step, next, and return commands.

files: Accessing source files

Accessing source files:

/<regular-expression>/
?<regular-expression>[
?]

Search forwards or backwards, respectively, in the current source file for the given regular-expression. Both forms of search wrap around. The previous regular expression is used if no regular expression is given to the current command.

See also: edit, file, func, list, and use commands.

scope: Scope

When displaying variables and expressions, dbx resolves names first using the static scope of the current function. The dynamic scope is used if the name is not defined in the first scope. If static and dynamic searches do not yield a result, an arbitrary symbol is chosen and the system prints the following message:

[using <module.variable>]

The <module.variable> is the name of an identifier qualified with a block name. You can override the name resolution procedure by qualifying an identifier with a block name. Source files are treated as modules named by the file name without the language suffix (such as, the .c suffix on a C language program).

threads: Thread display and control

If execution is stopped for any one thread, the entire process and all other threads in the process also stop. The dbx events such as breakpoints are not specific to any one thread. If one thread hits a breakpoint, all threads and the process stop. An automatic way to ensure that other threads do not hit breakpoints set by next(i) or step(i) is to set the variable $hold_next. dbx will then hold all threads except the current thread during those operations, then unhold all threads after the operation is complete. Holding all threads, or holding a thread that may release a mutex will cause the user program to deadlock. Conditional breakpoints can be used to specify breakpoints for any one particular thread by checking the execution state of the thread.
For example:

```plaintext
'stop at 42 if $t2==$current'
```

will set a breakpoint at line 42 only for thread two.

For manipulation of thread-oriented dbx variables ($c<n>$, $t<n>$, $l<n>$, $current$, $hold_next$, $cv_events$, $mv_events$, $tv_events$, and $lv_events), see the assign, print, set, and whatis commands.

---

**usage: Basic command usage**

Basic dbx command usage:

- **run**
  Begin or restart execution of the program.

- **print <exp>**
  Print the value of the expression.

- **where**
  Print currently active functions (stack trace).

- **stop at <line>**
  Set a breakpoint at the line.

- **stop in <proc>**
  Set a breakpoint when a particular function is called.

- **cont**
  Continue execution.

- **step**
  Single step one line.

- **next**
  Step to next line (skip over calls).

- **trace <line#>**
  Trace execution of the line.

- **trace <proc>**
  Trace calls to the procedure.

- **trace <var>**
  Trace changes to the variable.

- **trace <exp> at <line#>**
  Print <exp> when <line> is reached.

- **status**
  Print trace and stops that are in effect.

- **delete <number>**
  Remove trace or stop of given number.

- **whatis <name>**
  Print the declaration of the name.

- **list <line>, <line>**
  List source lines.

- **registers**
  Display register set.

- **quit**
  Exit dbx.
variables: "Set" variables

The following variables for the set subcommand have special meanings:

$asciichars
   Any dbx operation that displays the value of a character will interpret the binary representation of the character as ASCII.

$asciistrings
   Any dbx operation that displays the value of a string will interpret the binary representation of the string as ASCII.

$c<n>
   Condition variables.

$catchbp
   Catches breakpoints during the execution of the next command.

$charset="destCodePage,srcCodePage"
   Converts character strings before displaying them. The character strings are converted from the code page srcCodePage to destCodePage. The destCodePage must be IBM-1047. The default setting is not to convert the character strings.

$commandedit
   Enables the command-line facility.

$current
   Defined as a constant with the value of the current thread.

$cv_events
   Notifies the user but does not stop when a condition variable event is processed. The following trace information is sent to the user for the different events:

   (dbx) cont
   .
   cv initialize, object=0x2e04567
   cv wait, object=0x2e04567, mutex=0x2d04567, thid=0x0102030405060708
   cv unwait, object=0x2e04567, mutex=0x2d04567, thid=0x0102030405060708
   cv destroy, object=0x2e04567
   .

$dll_loads
   Set by default. dbx processes symbolics for DLLs as they are loaded.

$dll_loadstop
   Set by default. dbx stops the function call that caused the DLL to be loaded. If the DLL was loaded due to a variable reference or an explicit load, dbx stops at the source line that caused the DLL to be loaded.

$expandunions
   Displays values of each part of variant records or unions.

$expressionexhaustivesearch
   Searches all scopes in a user's program to determine and verify the scope for an expression. Selecting this option might degrade performance.

$fr<n>
   Hexadecimal floating-point register.

$frb<n>
   Binary floating-point register.
dbx: variables

$frd<n>
Decimal floating-point register

$hexchars
Prints characters as hexadecimal values.

$hexin
Input is interpreted in hexadecimal format.

Restriction: The $hexin variable is only supported in dbx command-line mode and does not affect the interpretation of GUI input. If the user of the GUI debugger wants input to be interpreted in hexadecimal format, the input must be prefixed with "0x".

$hexints
Prints integers in hexadecimal format instead of decimal format.

$historypage
Specifies the number of history items to be traversed when using the page up and page down keys.

$history_unique
Prevents consecutive duplicate commands from being saved to the history list.

$historywindow
Specifies the number of commands to display and retain in the history list.

$hold_next
Automatically holds all threads except the current thread during next, nexti, step or stepi command execution. If not set, all threads resume execution and might hit the breakpoint set by the next, nexti, step, or step command execution.

$1<n>
Read/write locks variables.

$listwindow
Specifies the number of lines to list around a function and to list when the list subcommand is used without parameters.

$lv_events
Notifies the user but does not stop when a read/write lock object event is processed. The following trace information is sent to the user for the different events:

```plaintext
(dbx) cont
.
lv initialize, object=0x2d04567
lv wait, object=0x2d04567, thid=0x0102030405060708
lv unwait, object=0x2d04567, thid=0x0102030405060708
lv lock, object=0x2d04567, thid=0x0102030405060708
lv unlock, object=0x2d04567, thid=0x0102030405060708
lv relock, object=0x2d04567, thid=0x0102030405060708
lv unrelock, object=0x2d04567, thid=0x0102030405060708
lv destroy, object=0x2d04567
.
```

$m<n>
Specifies mutex variables.

$maxstring
Specifies the maximum number of characters to be displayed when printing a string. String printing stops when $maxstring characters are printed. Set to zero to completely display strings. The default value is zero.

$mv_events
Notifies the user but does not stop when a mutex object event is processed. The following trace information is sent to the user for the different events:
$noargs
Omits arguments from subcommands, such as where, up, down, and dump.

$noflregs
Does not display the binary floating point representation of the floating point registers with the registers subcommand.

$nofldregs
Does not display the decimal floating point representation of the floating point registers with the registers subcommand.

$noflregs
Does not display the hexadecimal floating point representation of the floating point registers with the registers subcommand.

$novregs
When set, omits the display of vector registers from the registers subcommand.

$octin
Interprets input in octal format. The $octin variable is only supported in dbx command-line mode and does not affect the interpretation of GUI input. If the user of the GUI debugger wants input to be interpreted in octal format, the input must be prefixed with 0.

$octints
Prints integers in octal format.

$pc
Program counter register.

$psw
First word of the program status word register.

$psw0
First word of the program status word register.

$psw1
Second word of the program status word register.

$r<n>
General register.

$r_precision
Sets the amount of precision, in bytes, to use when displaying an integer value. Possible values are 4 and 8.

$repeat
Repeats the previous command if no command was entered.

$showbases
Displays the base class data when a derived class is printed.

$sigblock
Blocks all signals from reaching the program being debugged.

$sticky_debug
Recognizes sticky bit programs and DLLs in the loadmap.
$t<n>
Thread variables
$tv_events

Notifies the user but does not stop when a thread object event is processed. Trace information similar to the following example is sent to the user for the different events:

(dbx) cont
.
IPT create, thid=0x1234567890123456, stack=5200
IPT exit, thid=0x1234567890123456
tv create, thid=0x1234567890123456, created thid=0x1234567890123422, stack=5200
tv created, thid=0x1234567890123456, stack=5200
tv exit, thid=0x1234567890123456
tv wait, thid=0x1234567890123456, joining thid=0x1234567890123422
tv unwait, thid=0x1234567890123456, joined thid=0x1234567890123422

$unsafeassign

Turns off strict type checking between the two sides of an assign subcommand.

$unsafebounds

Turns off subscript checking on arrays.

$unsafegoto

Turns off the goto subcommand destination checking.
Appendix N. Shell commands changed for UNIX03

z/OS UNIX is UNIX95 conformant, with extensions to commands using formats (such as uppercase option letters) to avoid conflicts with subsequent UNIX standards. The specification of new options and changed command behavior by SUSv3 (also known as UNIX03) has resulted in conflicts with IBM’s extensions.

The _UNIX03 shell variable was introduced in z/OS 1.8 as a means of controlling whether certain shell commands behave according to Single UNIX Specifications, version 3 (SUSv3). _UNIX03=YES is only needed when an option or behavior conflicts with an existing z/OS implementation, and the SUSv3 behavior is desired. Additional command changes introduced in z/OS 1.9 have behavior conflicts controlled by the _UNIX03 variable. If _UNIX03 is not set (or set to "NO"), the z/OS 1.9 commands will maintain compatibility with prior releases.

Therefore, system programmers should make sure that _UNIX03 is not set in system-wide profiles (or setup scripts), including:

/etc/profile
/etc/csh.cshrc
/etc/csh.login
/etc/rc

Users who want the SUSv3 conformant behavior can set _UNIX03=YES in their own profile files, or in specific scripts or command invocations where it is needed.

Table 49 on page 1005 lists the commands that were changed for UNIX03. (It is not a complete list of changes to shell commands.) For more detail, see the specific command description.

<table>
<thead>
<tr>
<th>Table 49: UNIX shell commands and _UNIX03</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
</tr>
<tr>
<td>awk</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>bc</td>
</tr>
<tr>
<td>cp</td>
</tr>
<tr>
<td>Command</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>cksum</td>
</tr>
<tr>
<td>ed</td>
</tr>
<tr>
<td>file</td>
</tr>
<tr>
<td>mailx</td>
</tr>
<tr>
<td>od</td>
</tr>
<tr>
<td>Command</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>pax</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sed</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sum</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>tr</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>uudecode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>uuencode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Appendix O. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgcenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed email message to mhvrcfs@us.ibm.com.

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- z/OS TSO/E Primer
- z/OS TSO/E User's Guide
- z/OS ISPF User's Guide Vol I

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.
Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3\* FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* \* FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

* indicates an optional syntax element that is repeatable
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:
1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

**+ indicates a syntax element that must be included**

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.
This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road
reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

**Rights**

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

**IBM Online Privacy Statement**

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user’s name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at ibm.com/privacy and IBM’s Online Privacy Statement at ibm.com/privacy/details in the section entitled “Cookies, Web Beacons and Other Technologies,” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at ibm.com/software/info/product-privacy.

**Policy for unsupported hardware**

Various z/OS elements, such as DFSMS, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.
Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs that use z/OS UNIX System Services (z/OS UNIX).

Standards

In the following statement, the phrase *this text* refers to portions of the system documentation.

Portions of this text are reprinted and reproduced in electronic form in the z/OS, from IEEE Std 1003.1, 2004 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, copyright 2001-2004 by the Institute of Electrical and Electronics Engineers, Inc., and The Open Group. In the event of any discrepancy between these versions and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained at The Open Group (www.opengroup.org/unix).

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (" or "), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

UNIX is a registered trademark of The Open Group in the United States and other countries.

Acknowledgments

InterOpen Shell and Utilities is a source code product providing POSIX.2 (Shell and Utilities) functions to the z/OS UNIX services offered with MVS. InterOpen/POSIX Shell and Utilities is developed and licensed by Mortice Kern Systems (MKS) Inc. of Waterloo, Ontario, Canada.
Special Characters

_BPX_BATCH_SPAWN environment variable
description of 950

_BPX_BATCH_UMASK environment variable
description of 950

_BPX_SPAWN_SCRIPT environment variable
description of 950

_magic number
    #1 950

_BPX_TERMPATH environment variable
used by chcp 126, 128

_TAG_REDIR_ERR = BIN tcsh environment variable
description of 713

_TAG_REDIR_ERR = TXT tcsh environment variable
description of 712

_TAG_REDIR_ERR=BIN environment variable
description of 610

_TAG_REDIR_ERR=TXT environment variable
description of 610

_TAG_REDIR_IN shell variable 610

_TAG_REDIR_IN=BIN environment variable
description of 610

_TAG_REDIR_IN=BIN tcsh environment variable
description of 712

_TAG_REDIR_IN=TXT environment variable
description of 610

_TAG_REDIR_IN=TXT tcsh environment variable
description of 712

_TAG_REDIR_OUT = TXT environment variable
description of 610

_TAG_REDIR_OUT=BIN environment variable
description of 610

_TAG_REDIR_OUT=BIN tcsh environment variable
description of 712

_TAG_REDIR_OUT=TXT environment variable
description of 712

_TAG_REDIR_OUT=TXT tcsh environment variable
description of 712

_TEXT_CONV environment variable
used by vi 833

   explanation of 1
:(colon) shell command 150

? subcommand for dbx 203

.(dot) script
   returning from 565

.exrc file
   used by vi 833

.profile file 613

[ (left bracket) shell command 368

/subcommand for dbx 203

/bin directory
   setting up special files in the 886

/bin/mail file
   used by calendar 113

/dev/mt/0m file
   used by tar 666

/etc/auto.master file (continued)
used by automount 29

/etc/csh.cshrc
   used by tcsh login 671

/etc/csh.login
   used by tcsh login 671

/etc/inetd.conf file
   used by the inetd daemon 346

/etc/inetd.pid file
   used by the inetd daemon 348

/etc/magic file
   explanation of 970
   used by file 308

/etc/mailx.rc file
   configuration settings 181
   used by mailx 411, 421, 426

/etc/profile file
   used by the login shell 589

/etc/rc file
   used by automount 28

/etc/recover file
   used by exrecover 298

/etc/recover/$LOGNAME/VIn* file
   used by exrecover 298

/etc/recover/$LOGNAME/VIt* file
   used by exrecover 298

/etc/startup.mk file
   used by make 443

/etc/suid_us.profile
   used by sh 613

/etc/utmpx file
   used by who 843

/etc/yylex file
   used by lex 379

/etc/yylex.c file
   used by lex 378

/etc/yyparse.c file
   used by yacc 874

/tmp/e* file
   used by ed 279

/tmp/sh* file
   description of 613

/tmp/VIt* file
   used by exrecover 297

/tmp/VIn* file
   used by exrecover 297

/tmp/VIt* file
   used by exrecover 297

/usr/lib/config file
   used by uucc 784

/usr/lib/cron/at.allow file
   used by cron 181

/usr/lib/cron/at.deny file
   used by cron 181

/usr/lib/cron/cron.allow
   used by cron 181

/usr/lib/cron/cron.deny file
/usr/lib/cron/cron.deny file (continued)
used by cron 182
/usr/lib/cron/queuedefs file
used by cron 182
/usr/lib/lib.b file
used by bc 65
/usr/lib/libla file
used by lex 379
/usr/lib/liblxp.a file
used by lex 379
/usr/lib/uucp file
used by ucc 784
/usr/lib/uucp/config
used by ucc 784
/usr/lib/uucp/config file
used by ucc 789
/usr/man/%L/man-0-9/*.book file
used by man 455
/usr/man/%L/whatis file
used by man 455
/usr/spool/cron file
used by cron 181
/usr/spool/cron/atjobs file
used by cron 181
/usr/spool/cron/crontabs file
used by cron 181
/usr/spool/cron/log file
used by cron 181
/usr/spool/cron/pid
used by cron 181
/usr/spool/uucp spool directory 981
/usr/spool/uucp/.Sequence file
used by ucc 789
/usr/spool/uucp/LOGFILE file
used by ucc 789
/usr/spool/uucp/south directory 981
/usr/spool/uucp/public file
used by ucc 789
& shell operator 591
#!/
magic number 955
$HOME/.sh_history file
used by fc, history, r 302
$HOME/.exrc file
used by vi 833
$HOME/mbox file
used by mail 410
used by mailx 426
$MAILDIR file
used by mailx 426
$MAILRC file
used by mailx 426
$TMPDIR/pg* file
used by pg 542

Numerics

/usr/man/%L/man-0-9/*.stft file
used by man 455
3270 alarm
controlling the 910, 914, 918
3270 passthrough mode
used to invoke the TSO/E OBROWSE command 493

3270 terminals
specifying the use of DBCS 912

A

abnormal condition
trapping 750
abnormal interrupt
trapping 750
absolute movement command (for vi) 811
absolute movement commands
list of 811
access control list (ACL)
updating 583
access permission
changing 132
access time
setting, for destination files 166
accessibility
contact IBM 1009
features 1009
ACL primary operators
test shell command 737
action
explanation of 43
address
removing breakpoints from 208
address space
displaying list of 205
AF_ UENT stack
configuring the 159
ALARM function key for OMVS command 914
ALARM option of OMVS command 910
ALARM subcommand of OMVS command 918
alias
creating tracked aliases 337
detecting 767
removing definitions 767
those specified by the name argument 247
alias shell command 11
alias subcommand for dbx command 204
allnet environment variable
used by mailx 423, 427
alloc tcsh shell command 716
allocate
MVS standard files as z/OS UNIX files
using the BPXBATCH command 950
standard files
using the BPXBATCH command 884
using the BPXBATCH command 950
stdenv as z/OS UNIX files
using the BPXBATCH command 950
stderr as z/OS UNIX files
using the BPXBATCH command 950
stdin as z/OS UNIX files
using the BPXBATCH command 950
stdout as z/OS UNIX files
using the BPXBATCH command 950
ALLOCATE TSO/E command 897
allocating data sets 30
allocating file systems 30
allow
allow (continued)
  messages 456
amblist shell command 14
ampm shell variable
description of 698
append
  user's commands to a file
displaying
  list of mutex objects 244
appointment
displaying 113
ar shell command 15
arbitrary-precision arithmetic calculation language
  using the 52
archive
  copying files from directory 176
creating 176
extracting
  components from the 511
contents 176
tapes 664
archive file
  cpio format 969
manipulating 664
reading 510, 664
tar format 978
writing 510, 664
archive library
  creating 15
displaying symbol table 489
maintaining 15
ARGC (built-in variable for awk) 39
args subcommand for the dbx command 204
argument
  changing dates for 743
concatenating in the current shell environment 284
evaluating
  as an expression 294
in the current shell environment 284
obtaining from a list of parameters 331
printing 546
removing 567
writing to standard output 270
argv shell variable
description of 698
arithmetic calculation
  calculating to arbitrary precision 52
arithmetic expression
  evaluating 376
arrange
  items on command line 2
  options 1
array element (awk variable) 35
as shell command
  options 20
asa shell command 23
ASCII code pages for the terminal
  setting, resetting, or querying 126
ASCII to EBCDIC conversion 252
asid subcommand for dbx command 205
ask environment variable
  used by mailx 423
askbcc environment variable
  used by mailx 423
asksub environment variable
  used by mailx 423
assemble
  z/OS C and z/OS C++ source files 79
assign
  aliases for dbx subcommands 204
  values to variables 205
assign subcommand for dbx command 205
assistance technologies 1009
at shell command
  submitting jobs to cron 183
attribute of files
  listing 399
audit attribute
  changing 124
audit flag
  changing 124
autocorrect shell variable
description of 698
autoexpand shell variable
description of 698
autolist shell variable
description of 698
autoloader shell command 27
autoloader functions
description of 596
autologout shell variable
description of 698
automatic code set conversion
  disabling 993
  shell redirection 993
  using 993
automatic conversion
  ls 399
tcsh shell 712
automatic scrolling
  controlling 914, 918
  turning off 918
automatic, periodic, and timed events 694
automount
  allocating data sets 30
  allocating file systems 30
automount facility
  configuring the 27
automount shell command 27
AUTOMOVE 894
autoprint environment variable
  used by mailx 412, 418, 424
AUTOSCROLL function key for OMVS command 914
AUTOSCROLL function of OMVS command
  setting the 910
AUTOSCROLL option of OMVS command 910, 914
AUTOSCROLL subcommand of OMVS command 918
awk shell command
  action 43
  ARGC built-in variable 39
  arrays 36
  BEGIN 43
  built-in arithmetic functions 40
  built-in string functions 41
  built-in variables
    FILENAME 39
awk shell command (continued)
built-in variables (continued)
  FNR 39
  NF 39
  NR 39
comments 35
conditions 43
END 43
ENVIRON environment 37
examples 45
FILENAME file being read 40
FNR field separator string 39
functions 43
gline 40
NF field in current record 40
NR number of records read 40
OFMT output number format 44
OFS output field separator 44
operators 37
ORS output record separator 44
patterns 43
processing programs 34
RLENGTH built-in variable 41
RS record separator character 39
RSTART built-in variable 41
statements 43
SUBSEP 36
SYMTAB symbol table 37
system functions 42
variables 35

B
background job
  scheduling 182
backquoting 602
BACKSCR function key for OMVS command 914
BACKSCR subcommand of OMVS command 918
backslash shell variable
description of 698
backup files 664
backward retrieve function of OMVS command 916
bang environment variable
  used by mailx 420, 424
basename shell command 50
basic regular expression
  list of commands using 945, 946
batch environment
  running shell scripts and z/OS XL C/C++ applications
    under MVS 949
batch job
  submitting
    using the BPXBATCH command 884
    using the BPXBATCH program 949
  submitting for background processing 651
batch shell command
  submitting jobs to cron 183
bc shell command
  built-in functions 62
  built-in variables 54
  dynamic scoping 61
  specifying numbers in different bases 54
  between-rule circular dependency 436
bg shell command 66
binary file
  decoding 791
  encoding for transmission 792
bindkey tcsh sell command 716
blind carbon copy 411
BOTTOM function key for OMVS command 914
BOTTOM subcommand of OMVS command 918
Bourne shell 589
BPXACOPY program
  automatic setting of permission bits during installation
    961
BPXBATCH program
  invoked in the OSHELL REXX exec 958
  invoking
    with OSHELL 925
BPXBATCH TSO/E command 884
BPXBATSL
  run program using local spawn 884, 949
BPXCOPY program 959
bpmtxshell text command 67
bpmtxshell system REXX command 931
BPXMTEXT TSO command 886
bptrace shell command 68, 69
BPXTRACE TSO command 886
BPXWISHISPF environment variable
  used by obrowse 493
  used by oedit 499
BPXWISHTZ environment variable 887
bpwrmigf shell command 74
bpwrmigf system REXX command 931
BPXWMIGF TSO/E command 886
BPXWPERM environment variable
  used by oedit 499
BPXWRFDF environment variable
  used by tsocmd 757
brackets
  explanation of 1
break
  lines 320
break shell command 77
break up
  files 635
breakpoint
  removing from addresses 208
broadcast message 838
browse
  files
    with the obrowse shell command 493
    files, with BPXBATCH 925
    files, with the ISPF shell 886
    z/OS UNIX file system files
      with the OBROWSE TSO/E command 897
build
  argument lists before running a command 869
  list of files 926
built-in functions
  for the bc shell command 62
built-in shell commands
  : (colon) 150
  [ 368
alias 11
bg 66
break 77
built-in shell commands (continued)

cd 118
colon (:) 150
description of 596
echo 270
exit 290
false 301
fc 302
getopts 331
hash 337
jobs 360
kill 364
let 376
print 546
pwd 559
read 560
test 736
time 741
times 742
true 752
type 761
typeset 761
ulimit 764
umask 768
unalias 767
wait 837
whence 841

built-in variable
for the bc command 54
builtins tcsh shell command 717
byte count
calculating and displaying
with the sum command 653

bytes
counting 839

C

C escape sequences 503

C++ 78
C++ shell command 78
C++_64 848
c++_x 848
c89 78
c89_64 848
c89_x 848
c89/cc/cc++ environment variable

CXXSUFFIX 95
CXXSUFFIX_HOST 95
DAMPLEVEL 95
DAMPNAME 95
DCB121M 96
DCB133M 96
DCB137 96
DCB137A 96
DCB3200 96
DCB80 96
DCBF2008 95
DCBU 95
DEBUG_FORMAT 96
ELINES 96
EXTRA_ARGS 96
IL6SYSIX 96
ILCTL 97
ILMSGSIX 97
ILNAME 97
ILSUFFIX 97
ILSUFFIX_HOST 97
ILSYSIX 97
ILSYSLIB 97
ILXSYSIX 97
ILXSYSLIB 97
INCDIRS 97
INCLIBS 97
ISUFFIX 98
ISUFFIX_HOST 98
IXXSUFFIX 98
IXXSUFFIX_HOST 98
L6SYSIX 98
L6SYSLIB 98
LIBDIRS 98
LISYSLIB 98
LXSYSIX 98
LXSYSLIB 98
MEMORY 99
NEW_DATACLAS 99
NEW_DSNTYPE 99
NEW_MGMTCLAS 99
NEW_SPACE 99
NEW_STORCLAS 99
NEW_UNIT 99
NOCMDOPTS 99
OPERANDS 99
OPTIONS 99
OSUFFIX 99
OSUFFIX_HOST 100
OSUFFIX_HOSTQUAL 100
OSUFFIX_HOSTRULE 100
PMEMORY 101
PMSGS 101
PNAMES 101
PSUFFIX 101
PSUFFIX_HOST 101
PSYSIX 101
PSYSLIB 101
PVERSION 101
SLIB_PREFIX 102
SNAME 102
SSUFFIX 102
SSUFFIX_HOST 102

1021
c89/cc/c++ environment variable (continued)

_SSYSLIB 102
_STEPS 102
_SUSRLIB 103
_TMPS 103
_WORK_DATACLAS 103
_WORK_DSNCTYPE 103
_WORK_MGMCCLAS 103
_WORK_SPACE 103
_WORK_STORAGECLAS 103
_WORK_UNIT 103
_XSUFFIX 103
_XSUFFIX_HOST 103
 Il6SYSLIB 97

cc89/cc/c++ shell command
~W option
  compiler, prelinker, IPA linker and link editor options 86
  DLL and IPA extensions 86
  environment variables 92
  options 80
  specifying
    system and operational information to c89/cc/c++/cxx 92

c99 848

c99 command 112

c99 shell command 112

c99_64 848

c99_x 848

cal shell command 112

calculate and display
  checksum for each input file
    with the cksum command 143
    with the sum command 653
  number of bytes in each input file
    with the cksum command 143
    with the sum command 653

calendar file 114

calendar shell command 113

cancel
  print queue requests 114

cancel shell command 114

captoinfo shell command 115

carriage control
  interpreting 23

case shell command 594

case subcommand for dbx command 206

catch subcommand for dbx command 207

Caution section
  explanation of 6

cc 78

c c shell command 118, 192

c_64 848

cc_x 848

cd shell command 118

CDPATH environment variable
  description of 606
  used by cd 119
  used by vi 828, 833

cdpath shell variable
  description of 698

ceebldx shell command 121

change
  crontab entries 182

change (continued)
  dates for arguments 743
  dbx command prompts 228
  file access times 743
  file modification times 743
  files
    using diff output 506
    functions 216
    group owners 129
    groups 484
    groups of directories 137
    groups of files 137
    mount mode 927
    next line to be displayed 222
    owners of directories 137
    owners of files 137
    priorities of running processes 563
    root directory 138
    source files 215
    user ID
      connected with sessions 648
    working directories 118

change ACLs
  setfacl 583

character
  escaping 597
  translating 748
  character conversion table
    specifying the 910
  character special files
    creating 889

characters
  converting from one code set to another 342
  counting 839

chaudit shell command 124

chcp shell command 126

cHECK
  conditions 736
  for spelling errors 633
  path names 509

checksum
  calculating and displaying
    with the sum command 653
  chgrp shell command 129

child process
  waiting for it to end 837

child process time
  displaying time accumulated 742

child shell environment 611

chlabel shell command 130

chmod shell command 132

chmount shell command
  change file system mount attributes 135

chown shell command 137

chroot shell command 138

chtag shell command 140

circular dependencies 436

cksum shell command 143

clear breakpoints at addresses 208

clear shell command 145

clear subcommand for dbx command 207

cleari subcommand for dbx command 208

clock daemon (cron) 180
clone output streams 735

close 735
  file descriptors 288
  shell sessions 914, 918
  standard output (stdout) 604

CLOSE function of OMVS command 914
CLOSE subcommand of OMVS command 918

cmd environment variable 288
  used by mailx 416, 424

cmp shell command 146
code set 416
  converting characters to another code set 342
col shell command 149
collect 288
  debugging information 912
colon (:) shell command 150
COLUMNS environment variable
  description of 606
  used by ed 278
  used by ls 405
  used by pg 539, 542
  used by ps 558
  used by sed 577
  used by shedit 616
  used by vi 833
COLUMNS tcsh environment variable
  description of 711
command shell variable
  description of 698
comm shell command 151
command 312
  aliases
    creating or displaying 11
  built-in 611
  changing prompts, for dbx 228
  constructing
    in the current shell environment 284
    with templates 869
  conventions 1
  creating aliases 11
  descriptions
    reading 1
  displaying
    aliases 11
    elapsed time 741
    suppressing command numbers 303
  editing 302, 615
  executing 138
  interpreting names 841
  names
    interpreting 841
  numbers
    suppressing 303
  options
    setting 579
    unsetting 579
  passing to shell for execution 238
  processing history lists 302
  prompts, changing for dbx 228
  reading descriptions 1
  reentering 302
  remote execution
    displaying information about 793
  running
command (continued)
  running (continued)
    after constructing an argument list 869
    at a different priority 485
    at a specified time 24
    from the shell 757
    simple 153
    using the OMVS interface 757
    using the TSO/E service routine 757
    when system is not busy 51
  setting options 579
  specifying command lines for another command 288
  substituting 602
  suppressing numbers 303
  sysvar 655
template 869
TSO/E
  ALLOCATE 897
  BPXBATCH 884
  ISHELL 886
  MKDIR 888
  MKNOD 889
  MOUNT 891
  OBROWSE 896
  OCOPY 897
  OEDIT 902
  OGET 903
  OGETX 906
  OMVS 909
  OPUT 920
  OPUTX 922
  OSHELL 925
  OSTEPLIB 926
  UNMOUNT 927
  unsetting options 579
command aliases
  displaying 11
command interpreter 428
code
command line
  specifying for another command 288
command mode 411, 807
command shell command 153
command substitution 602
commands
  nonfunctional
    cancel 114
    lpstat 399
  running
    on remote sites 802
  communicate
    with other users 845
compare
  directories
    with the dircmp command 263
    with the ISHELL command 886
    with the OSHELL command 925
  files
    with the cmp command 146
    with the diff command 257
    with the ISHELL command 886
    with the OSHELL command 925
terminfo database entries 348
compile
  link-edit object file 79
compile (continued)
  terminfo database entries 740
  UUCP configuration files 783
  z/OS C and z/OS C++ source file 79
compiler
  yacc 872
complete shell variable
description of 698
component directory 510
component file
definition of 663
compress
  files
    using Huffman coding 500
    using Lempel-Ziv compression 155
  spaces into tabs 771
compress shell command 155
concatenate
  arguments in the current shell environment 284
  corresponding or subsequent lines of files 502
  files 116
  lines 502
  lines of input files 502
condition
  explanation of 43
  testing for 736
  trapping abnormal 750
condition subcommand for dbx command 209
condition variable
displaying list of 209
conditional expression 443
confighfs shell command 157
configstk shell command 159
configstrm shell command 160
configuration file for xlc
default name 858
configuration files
  /usr/lib/config
    used by uucc 784
  Devices 784
  Dialcodes 784
  Dialers 784
  Permissions 784
  reading and compiling contents of UUCP 783
  Systems 784
configuration variable
  writing values to standard output 325
configure
  AF_LEINT stacks 159
  automount facility, the 27
  connecting to
    remote systems, with the uucico daemon 785
  console log
    saving messages in 393
  construct
    argument lists before running a command 869
    commands in the current shell environment 284
  cont subcommand of for command 209
contact
  z/OS 1009
context diff file 506
context-dependent movement commands (for vi) 812
continuation prompt 608
continue shell command 161
control
  3270 alarms 910, 914, 918
  automatic scrolling 914, 918
  AUTOSCROLL function of OMVS command 910
  display of function key settings 915
control character
  processing 149
  CONTROL function of OMVS command 914
control operator 595
conv environment variable
  used by mailx 410, 427
conventions for command descriptions 1
conversion buffer 254
convert
  characters from one code set to another 342
  from ASCII to EBCDIC 252
  from EBCDIC to ASCII 252
  from lowercase to uppercase 253
  from uppercase to lowercase 253
  from variable to fixed records 252
  CONVERT option of OMVS command 910
  copy
    archive files, with the tar command 664
    data read from standard input to standard output 747
    data sets into files, with BPXCOPY 959
    data with format conversion 251
    file descriptors 288
    files
      between UUCP systems 786
      from one directory to another 176
      selectively 189
      to MVS partitioned data set 906
      to target named by the last argument on command line 163
      to users on remote systems 801
      with BPXBATCH 925
      with data conversion 251
      with the ISPF shell 886
    in/out file archive 176
    MVS data sets
      members into z/OS UNIX file system directories 922
      members into z/OS UNIX file system files 920
      to another member or file 897
    standard input to each output file 735
    z/OS UNIX file system
      directories to MVS partitioned data set 906
      files into MVS data sets 903
      files to another member or file 897
copy mode 511
copytree REXX sample 162
correct shell variable
  description of 698
count
  bytes 839
  characters 839
  lines 839
  newlines 839
  words 839
cp shell command 163
cpio archive
  reading and writing 176
cpio archive format 969
cpio shell command 176
CPU time 742
create
    aliases for dbx subcommands 204
    archives 176
    character special files 889
    command aliases 11
    crontab entries 182
directories
    for each named directory argument 459
    with the MKDIR command 888
    FIFO special files 461
    hard link 381
    libraries 442
    library archives 15
    link to files 383
    message catalogs 322
tag files 187
    tracked aliases 337
create executable files 79
cron
    submitting jobs to 183
crontab
    changing entries 182
    creating entries 182
    editing entries 183
    obtaining output of entries 183
    shell command
    submitting jobs to cron 183
crt environment variable
    used by mailx 417, 424
csplit shell command 184
ctags shell command 187
cu shell command 188
current appointment
    displaying 113
current mail message 412
current operating system
    displaying name of the 768
current position pointer 808
current users
    displaying information about 842
current working directory
    changing to previous working directory 119
    displaying path name of the 559
    setting to value of the HOME environment variable 119
customize
    settings for function keys 914
cut shell command 189
cwd shell variable
    description of 698
cxx 78
cxx_64 848
cxx_x 848

D
daemons (continued)
    uuxqt 805
dash
    explanation of 1
data
    displaying after uncompressing 876
    manipulating 34
    reading 251
    refreshing 915
    removing from executable files 640
    transferring to remote sites 983
    writing 251
data file 983
data set
    copying
        between two files 897
        BPXCOPY program, with the 959
data set names
    specifying in the shell 991
database
    joining two 362
date
    displaying the 192
date shell command 192
    Daylight savings time
        used in the TZ environment variable 988
    DBCS mode
        specifying 912
    DBCS option of OMVS command 912
dbgld shell command
    examples 197
    options 196
    restrictions 196
dbx debug program
    defining values for variables 234
    searching for source files 248
dbx shell command
    creating aliases for subcommands 204
    displaying synopsis of 217
    reading subcommands from file 239
    set variables 1001
    subcommands
        ? 203
        / 203
        alias 204
        args 204
        asid 205
        assign 205
        case 206
        catch 207
        clear 207
        cleari 208
        condition 209
        cont 209
        delete 210
        detach 210
        display memory 211
        down 214
        dump 214
        edit 215
        file 215
        func 216
        goto 216
        gotoi 217
dbx shell command (continued)
subcommands (continued)
  help 217
  history 218
  ignore 218
  list 219
  listfiles 219
  listi 221
  map 221
  move 222
  multproc 223
  next 224
  nexti 225
  object 225
  onload 226
  pid 229
  print 228
  prompt 228
  quit 229
  record 231
  registers 231
  rerun 232
  return 233
  run 234
  set 234
  sh 238
  skip 239
  source 239
  status 240
  step 240
  stepi 241
  stop 241
  stopi 243
  trace 245
  tracei 246
  unalias 247
  unset 247
  up 247
  use 248
  whatis 249
  where 249
  whereis 250
  which 251

dbx shell command prompt
  changing 228

dd shell command 251
  deactivate
    function key 915
  dead.letter file
    used by mail 408, 409
    used by mailx 419
  DEBUG option of OMVS command 912
  debug programs
    changing interpretation of symbols 206
    with the dbx command 198
  debug session
    enabling or disabling multiprocess 223
    ending 229
  debugger
    using the 198
  debugging information
    collecting 912
    writing 912
  decode
    (continued)
      files packed by using Hoffman coding 780
      Huffman-packed files 538
      transmitted binary files 791
  default
    function key 916
  define
    local environments 391
    values for dbx variables 234
  delay program execution 624
  delete
    alias definitions 767
    aliases 247
    arguments 567
    attributes of variables and functions 781
    breakpoints at addresses 208
    directories 569
    directory entries 567, 776
    information from executable files 640
    stops
      from programs 210
      from source lines 207
    traces from program 210
    trailing part of file names 265
    values of variables and functions 781
    variables 247
  delete subcommand of the for command 210
  Description section
    explanation of 3
  destination file
    setting
      destination time 166
      modification time 166
  detach subcommand for dbx command 210
  detect
    aliases 767
    spelling errors 633
  dextract shell variable
    description of 698
  df
    in a sysplex 257
  df shell command 255
  diff output
    used when changing files 506
  diff shell command 257
  dircmp shell command 263
  directory
    /usr/spool/uucp 981
    /usr/spool/uucp/south 981
    changing
      access permission of 132
      audit attributes 124
      audit flags 124
      group owners 129
      modes 132
      owners and groups 137
    comparing
      with the ISHELL command 886
      with the OSHELL command 925
  copying files 176
  creating
    for each named directory argument 459
    with BPXBATCH 925
    with the ISPF shell 886
directory (continued)
  creating (continued)
    with the MKDIR command 888
  deleting, with BXPXBATCH 925
  deleting, with the ISPF shell 886
  listing files in a
    with ISHELL 886
    with OSHELL 925
  moving files to a different 472
  naming, with BXPXBATCH 925
  naming, with the ISPF shell 886
  removing
    entries 567, 776
    with the rmdir command 569
  setting owners and groups 137
  setting up special files in the /bin 886
directory (UUCP)
  searching public 796
directory substitution 598
dirname shell command 265
dirsfile shell variable
  description of 699
dirstack shell variable
  description of 699
disable multiprocess debugging 223
display
  active stop subcommands 240
  active trace subcommands 240
  aliases for dbx subcommands 204
  amount of free space on file system 255
  appointments 113
  arguments
    of programs 204
  attributes and contents of a symbolic link 925
  attributes and contents of symbolic links 886
  calendar 112
  changing next line to be displayed 222
  command aliases 11
  commands
    suppressing command numbers 303
    with the fc command 302
  commands in history list 218
  compressed files 538
  crontab entries 183
  current appointments 113
  currently exported variables 293
  data after uncompressing 876
  dates 192
  DBCS characters 9
  declaration of program components 249
  differences between two files 257
  elapsed time for a command 741
  environment variables 283, 579
  errnojr_value 121, 281
  extended attributes for files 299
  file attributes 886, 925
  files
    page by page 463
    files interactively 539
    first part of files 338
    for month or year 112
    formatted information from object and executable files 14
display (continued)
  information about
    open files, sockets, and pipes 877
  information about current users 842
  information about locales 386
  information about the OMVS command 918
  input files 539
  instructions in program 221
  last part of files 658
  lines common to two files 151
  list of active condition variables 209
  list of active mutex objects 223, 244
  list of active program and functions 249
  list of active read/write lock objects 230
  list of files
    of module 220
  list of functions associated with a program file 220
  list of UUCP systems 795
  load characteristics of program 221
  log information about UUCP events 793
  login information 842
  memory 211
  message catalogs 267
  messages from message catalogs 268
  names of
    current operating systems 768
    shell variables 579
    variables in procedures 214
  piped files 539
  process IDs 321
  process status 550
  processors 741
  qualifications
    of given identifiers 251
    of symbols 250
  specified number of lines in source files 219
  static system symbols 655
  status of pending UUCP transfers 798
  status of print queues 399
  strings in a binary file 638
  synopsis of dbx commands 217
  system time accumulated by commands 742
  terminal options 641
  times 192
  unprintable characters 116
  user time accumulated by the shell 742
  values of
    floating-point registers 231
    general-purpose registers 231
    instruction registers 231
    program status words (PSW) 231
    shell variables 579
    system control registers 231
    variables in procedures 214
  values of environment variables 547
display ACL entries
getfacl 329
display memory subcommand for dbx command 211
DISPLAY tcsh environment variable
  description of 711
display-oriented text editor
  vi 807
displaying
double-byte characters
corresponding to the 
kanji characters in 
Japanese text.

echo shell command 270

echo shell variable
description of 699

EBCDIC to ASCII. conversion 252

ECHO function key for OMVS command 915

ECHO option of OMVS command 912

echo shell command 270
echo shell variable
description of 699

ECHO subcommand of OMVS command 918

echo_style shell variable
description of 700

ed shell command 272
ed text editor
using the 272
ed.hup file
used by ed 279

edcmtxt shell command 281
edit
commands 302
crontab entries 183
files
with the edit shell command 498
files, with BPXBATCH 925
files, with the ISPF shell 886
message catalogs 322
z/OS UNIX files
with the OEDIT TSO/E command 902
edit shell variable (continued)
description of 700
edit subcommand for dbx command 215
editing subcommands
starting 572
editor
invoking 215
evaluator environment variable
used by mailx 415, 419
EDITOR environment variable
description of 606
used by crontab 183
used by shedit 616
editor initialization 832
EDITOR tshell environment variable
description of 711
egrep shell command 283, 333
electronic mail
sending and receiving 409
elif shell subcommand 595
ellipsis
explanation of 2
ellipsis shell variable
description of 700
else shell subcommand 595
emacs
enabling, with the EDITOR environment variable 606
enable multiprocess debugging 223
encode
binary files for transmission 792
files
using Huffman coding 500
end

dbx debugging sessions 229
jobs 364
processes 364
shell sessions 915, 919
shells 290
end of file 658
ENDPASSTHROUGH option of OMVS command 913
ENV environment variable
description of 606
used by sh 589, 591
used by vi 834
env shell command 283
environment
defining local 391
environment variable
_ACCEPTABLE_RC
used by c89/cc/c++ 93
_ASUFX
used by c89/cc/c++ 93
_ASUFX_HOST
used by c89/cc/c++ 93
_BPX_BATCH_SPAWN
description of 950
_BPX_BATCH_UMASK
description of 950
_BPX_SPAWN_SCRIPT
description of 950
_BPX_TERMPATH
used by chcp 126, 128
_CCMODE
used by c89/cc/c++ 93
environment variable (continued)

_CCN_32_RUNOPTS
  used by c89/cc/c++ 92
_CCN_64_RUNOPTS
  used by c89/cc/c++ 92
_CCN_IPA_WORK_SPACE
  used by c89/cc/c++ 92
_CLASSLIB_PREFIX
  used by c89/cc/c++ 93
_CLASSVERSION
  used by c89/cc/c++ 93
_CLIB_PREFIX
  used by c89/cc/c++ 94
_CMEMORY
  used by c89/cc/c++ 94
_CMSGS
  used by c89/cc/c++ 94
_CNAME
  used by c89/cc/c++ 94
_CSUFFIX
  used by c89/cc/c++ 94
_CSYSLIB
  used by c89/cc/c++ 94
_CVERSION
  used by c89/cc/c++ 95
_CXXSUFFIX
  used by c89/cc/c++ 95
_CXXSUFFIX_HOST
  used by c89/cc/c++ 95
_DAMPLEVEL
  used by c89/cc/c++ 95
_DAMPNAME
  used by c89/cc/c++ 95
_DCB121M
  used by c89/cc/c++ 96
_DCB133M
  used by c89/cc/c++ 96
_DCB137
  used by c89/cc/c++ 96
_DCB137A
  used by c89/cc/c++ 96
_DCB3200
  used by c89/cc/c++ 96
_DCB80
  used by c89/cc/c++ 96
_DCBF2008
  used by c89/cc/c++ 95
_DCBU
  used by c89/cc/c++ 95
_DEBUG_FORMAT
  used by c89/cc/c++ 96
_ELINES
  used by c89/cc/c++ 96
_EXTRA_ARGS
  used by c89/cc/c++ 96
_IL6SYSIX
  used by c89/cc 96
_IL6SYSLIB
  used by c89/cc 97
_ILCTL
  used by c89/cc 97
_ILMSGS
  used by c89/cc 97
_ILNAME
  (continued)
  used by c89/cc/c++ 97
_ILSUFFIX
  used by c89/cc/c++ 97
_ILSUFFIX_HOST
  used by c89/cc/c++ 97
_ILSYSIX
  used by c89/cc/c++ 97
_ILSYSLIB
  used by c89/cc/c++ 97
_ILXSYSIX
  used by c89/cc/c++ 97
_ILXSYSLIB
  used by c89/cc/c++ 97
_INCDIRS
  used by c89/cc/c++ 97
_INCLIBS
  used by c89/cc/c++ 97
_ISUFFIX
  used by c89/cc/c++ 98
_ISUFFIX_HOST
  used by c89/cc/c++ 98
_IXXSUFFIX
  used by c89/cc/c++ 98
_L6SYSIX
  used by c89/cc/c++ 98
_L6SYSLIB
  used by c89/cc/c++ 98
_LIBDIRS
  used by c89/cc/c++ 98
_LXSYSIX
  used by c89/cc/c++ 98
_LXSYSLIB
  used by c89/cc/c++ 98
_LXNAME
  used by c89/cc/c++ 98
_MEMORY
  used by c89/cc/c++ 99
_NEW_DATACLAS
  used by c89/cc/c++ 99
_NEW_DSNTYPE
  used by c89/cc/c++ 99
_NEW_MGMTCLAS
  used by c89/cc/c++ 99
_NEW_SPACE
  used by c89/cc/c++ 99
_NEW_STORCLAS
  used by c89/cc/c++ 99
_NEW_UNIT
  used by c89/cc/c++ 99
_NOCMDOPTS
  used by c89/cc/c++ 99
_OPERANDS
  used by c89/cc/c++ 99
_OPTIONS
  used by c89/cc/c++ 99
_OSUFFIX
  used by c89/cc/c++ 99
_OSUFFIX_HOST
  used by c89/cc/c++ 100
_OSUFFIX_HOSTQUAL
environment variable (continued)

__OSUFFIX_HOSTQUAL
used by c89/cc/c++
100

__OSUFFIX_HOSTRULE
used by c89/cc/c++
100

__PLIB_PREFIX
used by c89/cc/c++
100

__PMEMORY
used by c89/cc/c++
101

__PMSG
used by c89/cc/c++
101

__PNAME
used by c89/cc/c++
101

__PSUFFIX
used by c89/cc/c++
101

__PSUFFIX_HOST
used by c89/cc/c++
101

__SLIB_PREFIX
used by c89/cc/c++
102

__SNAME
used by c89/cc/c++
102

__SSUFFIX
used by c89/cc/c++
102

__SSUFFIX_HOST
used by c89/cc/c++
102

__SSYSLIB
used by c89/cc/c++
102

__STEPS
used by c89/cc/c++
102

__SUSRLIB
used by c89/cc/c++
103

__TAG_REDIR_ERR=BIN
description of 610

__TAG_REDIR_ERR=TXT
description of 610

__TAG_REDIR_OUT=BIN
description of 610

__TAG_REDIR_OUT=TXT
description of 610

__TEXT_CONV
used by vi 833

__TMPS
used by c89/cc/c++
103

__WORK_DATACLASS
used by c89/cc/c++
103

__WORK_DSNTYPE
used by c89/cc/c++
103

__WORK_MGMTCLASS
used by c89/cc/c++
103

__WORK_SPACE
used by c89/cc/c++
103

__WORK_UNIT
used by c89/cc/c++
103

__XSUFFIX
used by c89/cc/c++
103

__XSUFFIX_HOST
used by c89/cc/c++
103

allnet
used by mailx 423,
427

append
used by mailx 417

ask
used by mailx 423

askbcc
used by mailx 423

asksub
used by mailx 423

autocrypt
used by mailx 412,
418, 424

bang
used by mailx 420,
424

BPXWISHISP
used by obrowse 493

used by oedit 499

BPXWPERM
used by oedit 499

BPXWRDFD
used by tsocmd 757

CDPATH
description of 606
used by cd 119

used by vi 828, 833

cmd
environment variable (continued)

cmd (continued)
used by mailx 416, 424

COLUMNS
description of 606
used by ed 278
used by ls 405
used by pg 539, 542
used by ps 558
used by sed 577
used by shedit 616
used by vi 833

conv
used by mailx 410, 427

crt
used by mailx 417, 424
description of 606
displaying 283, 579
displaying the value of a 547
dot
used by mailx 424

editor
used by mailx 419

EDITOR
description of 606
used by crontab 183
used by shedit 616

ENV
description of 589, 606
used by sh 591
used by vi 834

ERRNO
description of 606
escape
used by mailx 419, 424

EXINIT
used by vi 832, 834

FCEDIT
description of 606

flipr
used by mailx 417, 424

folder
used by mailx 414, 417, 424

FPATH
description of 607

header
used by mailx 424

HISTFILE
description of 607
used by fc, history, r 302

HISTSIZE
description of 607
used by fc, history, r 302

hold

environment variable (continued)

hold (continued)
used by mailx 412, 424

HOME
description of 607
used by cd 119
used by crontab 183
used by mail 408
used by mailx 422
used by vi 828, 833, 834

IFS
description of 607
used by read 560, 561
used by sh 590

ignore
used by mailx 411, 424

ignoreeof
used by mailx 424

indent
used by mailx 425

indentprefix
used by mailx 420, 425

keep
used by mailx 425

keepsave
used by mailx 412, 425

LANG
description of 607

LIBPATH
description of 607
used by c89/cc/c++ 87

LINENO
description of 607

LINES
description of 607
used by pg 540, 542
used by vi 831

LOCPATH
description of 607

MAIL
used by mailx 422

MAILCHECK
description of 607

MAILDIR
used by mailx 422

MAILER
used by calendar 113

MAILPATH
description of 608

MAILRC
used by mailx 422

mailserv
environment variable (continued)
mailserv (continued)
    used by mailx 425, 427
MAKEFLAGS
    used by make 442, 444
MAKESTARTUP
    used by make 428, 442, 444
MANPAGER
    used by man 455, 456
MANPATH
    description of 608
    used by man 455, 456
MBOX
    description of 608
metoo
    used by mailx 425
NLSPATH
    description of 608
OLDPWD
    description of 608
    used by cd 120
onehop
    used by mailx 425, 427
OPTARG
    used by getopts 332
OPTIND
    used by getopts 332
outfolder
    used by mailx 425
page
    used by mailx 416, 425
pager
    used by mailx 417
PAGER
    used by man 455
PATH
    description of 608
    used by crontab 183
    used by vi 834
PPID
    description of 608
PRINTER
    used by lp 397, 398
prompt
    used by mailx 425
PS1
    description of 608
PS2
    description of 608
    used by read 561
PS3
    description of 608
PS4
    description of 608
PWD
    description of 608
    used by cd 120
quiet
    used by mailx 425
environment variable (continued)
RANDOM
    description of 608
record
    used by mailx 410, 415, 417, 425
REPLY
    used by read 561
replyall
    used by mailx 425, 427
save
    used by mailx 425
screen
    used by mailx 416, 426
SECONDS
    description of 609
sendmail
    used by mailx 426, 427
sendwait
    used by mailx 426, 427
setting 283
SHELL
    description of 609
    used by awk 51
    used by crontab 183
    used by ed 278
    used by mailx 417
    used by make 444
    used by vi 826, 834
showto
    used by mailx 426
sign
    used by mailx 419, 426
Sign
    used by mailx 426
STEPLIB
    description of 609
SYSEXEC
    used by tsocmd 758
SYSPROC
    used by tsocmd 758
TERM
    used by talk 662
    used by vi 808, 830, 834
terminfo 352
TERMININFO
    used by talk 662
    used by vi 834
TMOUT
    description of 609
TMP
    used by exrecover 297
TMP_VI
    used by vi 834
TMPDIR
    description of 609
    used by ed 278
    used by man 455
environment variable (continued)
TMPDIR (continued)
used by pg 542
used by vi 834
toplines
used by mailx 418, 426
TSOALLOC
used by tsocmd 758
TSOPROFILE
used by tsocmd 758
TZ
description of 609
used by cron 180
used by crontab 183
used by date 194
used by locale 391
used by ls 405
used by mail 408
used by mailx 415
used to specify system and
operational information to
c89/cc/cc++/cxx 92
used to specify system and
operational information to
xlc/xlC 849
VISUAL
description of 609
used by mailx 418, 420
used by shedit 616
environment variables
partial list 989
Environment Variables section
explanation of 3
ERRNO environment variable
description of 606
errnojr_value
displaying 121, 281
escape character
displaying current settings 919
specifying the 913
turning off display for settings 919
escape environment variable
used by mailx 419, 424
ESCAPE option of OMVS command 913
escape sequences 597
escaping characters 597
eval shell command 284
evaluate
arguments as expression 294
arguments in the current shell environment 284
arithmetic expression 376
shell expressions 150
ex command
regular expressions 831
special characters 826
ex mode
commands issued from 819
current position pointer 808
entering 819
starting session in 807
ex shell command
creating tag files for the 187
ex text editor
using the 285
Examples section
explanation of 3
exception condition
trapping 750
exec shell command 288
executable
reentrant 110
executable file
creating 79
displaying symbol table 489
execute
commands on remote sites 802
execute files 983
exhaustive mode 250
EXINIT environment variable
used by vi 832, 834
exit code
returning a nonzero 301
exit shell command 289
exit shell subcommand 614
exit status
returning values of 0 752
Exit Values section
explanation of 6
expand
compressed data written by Lempel-Ziv compression
770
tabs to spaces 290
expand shell command 290
export
aliases 12
evironment variables 293
export shell command 293
expr operators 294
expr shell command 294
expression
evaluating 294, 376
handling, for the dbx command 200
printing tracing information 245
expression values
printing 228
exrecover shell daemon 296
extattr shell command
extended attributes
setting, resetting, and displaying 299
extended ACL entries 403
extended attributes
APF | NOAPF 961
PROGCTL|NOPROGCTL 961
SHAREAS | NOSHAREAS 961
SHARELIB | NOSHARELIB 961
extended regular expression
list of commands using 945, 946
external link
identifying 400, 403
ln 384
extract
components from archives 511
contents of archive files 176
false shell command 301
fc shell command 302
FCEDIT environment variable
description of 606
feedback xix
fg shell command 304
fgrep shell command 333
Fibonacci sequence 613
field (awk variable) 35
FIFO special files
creating 461, 462
fgignore shell variable
description of 700
file
allocating
using the BPXBATCH program 950
backing up
archive files 664
backup 664
binary
decoding transmitted 791
encoding for transmission 792
browsing, with BPXBATCH 925
browsing, with the ISPF shell 886
calculating and displaying
byte counts 653
checksum 653
changing
access permission of 132
access times 743
audit attributes 124
audit flags 124
group owners 129
groups 137
modes 132
modification times 743
owners 137
source 215
using diff output 506
comparing two
with the cmp command 146
with the diff command 257
with the ISHELL command 886
with the OSHELL command 925
with the sum command 653
compressed
displaying 538
compressing
using Lempel-Ziv compression 155
concatenating lines into standard output 502
converting
from ASCII to EBCDIC 252
from EBCDIC to ASCII 252
copying
archive files 664
between sites 786
between two files 897
to target named by the last argument on command line 163
to users on remote systems 801
with BPXBATCH 925
with data conversion 251
file (continued)
copying (continued)
with the ISPF shell 886
creating
character special files for file systems 889
directories for 888
FIFO special 462
links to 383
with BPXBATCH 925
with the ISPF shell 886
deleting
information from 640
with BPXBATCH 925
with the ISPF shell 886
displaying
attributes of 886, 925
compressed 538
first part 338
interactively 539
last part of the 658
lines common to two files 151
page by page 463
specified number of lines in source 219
dumping to standard output 494
ingoing
with the oedit shell command 498
with the OEDIT TSO/E command 902
ingoing, with BPXBATCH 925
ingoing, with the ISPF shell 886
ingoing, with previous arguments 232
ingoing, with the OSHELL command 925
lines
numbering 487
list of
building, with the OSTEPLIB command 926
listing
attributes 399
names 399
maintaining
interdependent 427
program-generated 427
manipulating repeated lines 773
merging corresponding or subsequent lines of files 502
misspelled words
looking for 633
moving 472
naming, with BPXBATCH 925
naming, with the ISPF shell 886
object
displaying symbol table of an 489
output tags
used by ctags 188
used by uptime 783
passing small amounts to 271
processing 34
reading dbx subcommands from 239
removing
information from 640
renaming 472
running
object files, with previous arguments 232
with the ISHELL command 886
with the OSHELL command 925
file (continued)

searching
- backward for patterns 203
- for specified patterns 333
- for text strings 886, 925
- forward for patterns 203

sending
- paginated files to printer 543
- to other users 845

setting
- destination time 166
- groups 137
- modification time 166
- owners 137

showing differences between two 257

sorting
- in topological order 759

splitting 635

summarizing use of space 269
text
- comparing two 257
- concatenating 116
- counting items in 839
- displaying 116
- finding information in 34
- retrieving information from 34
- splitting 184

transfers
- displaying information about 793

uncompressing
- Huffman-coded 538

words 633

file cache
- managing 319

file descriptor
- closing 288
- copying 288
- opening 288

file formats
- magic 970
- queuedefs 976
- tags 977
- tar 978
- utmpx 980

file mode creation mask
- setting or returning 766

file name
- deleting trailing parts 265
- expanding on command line 271
- generation 604

file owner
- group
  - GID(group) 962
  - UID(owner) 962
- file recovery daemon for vi (exrecover) 296

file shell command
- using the magic file 970

file space
- summarizing use of 269

file subcommand for dbx command 215

file system
- unmounting from the shell 777

file system recovery
- TSO MOUNT 896

file systems
- browsing files in the
  with the obrowse shell command 493
- hierarchical
  - unmounting 929
- TFS
  - mounting 894
  - unmounting 929

file tag
- automount 27
- ls 399
- MOUNT TSO/E command 891
tcsh shell 712

file tag information
- changing 140

file tags
- changing 140

file transfer
daemon for (uucico) 785
requests
- processing, with the uucico daemon 785

file type
determining the 305

file-creation permission-code mask
- setting or returning 766

filec shell variable
description of 700

FILENAME built-in variable for awk 39

files
  .exrc
    - used by vi 833
  .profile 613
  /bin/mail
    - used by calendar 113
  /dev/mt/0m
    - used by tar 666
  /etc/auto/master
    - used by automount 29
  /etc/csh.cshrc
    - used by tcsh login 671
  /etc/csh.login
    - used by tcsh login 671
  /etc/inetd.conf
    - used by the inetd daemon 346
  /etc/inetd.pid
    - used by the inetd daemon 348
  /etc/magic
    - used by file 308
  /etc/mailx.rc
    - configuration settings 181
    - used by mailx 411, 421, 426
  /etc/profile
    - used by the login shell 589
  /etc/rc
    - used by automount 28
  /etc/recover
    - used by exrecover 298
  /etc/recover/$LOGNAME/VIn*
    - used by exrecover 298
  /etc/recover/$LOGNAME/VIt*
    - used by exrecover 298
  /etc/startup.mk
    - used by make 443
  /etc/suid_us.profile
files (continued)
/etc/suid_us.profile (continued) used by sh 613
/etc/utmpx used by who 843
/etc/yylex used by lex 379
/etc/yylex.c used by lex 378
/etc/yparse.c used by yacc 874
/tmp used by fc, history, r 303
tmp/e* used by ed 279
/tmp/sh* description of 613
/tmp/Vll* used by exrecover 297
/tmp/VI*n* used by exrecover 297
/tmp/VIlt* used by exrecover 297
/usr/lib/config used by uucc 784
/usr/lib/cron/at.allow used by cron 181
/usr/lib/cron/at.deny file used by cron 181
/usr/lib/cron/cron.allow used by cron 181
/usr/lib/cron/cron.deny used by cron 182
/usr/lib/cron/queuedefs used by cron 182
/usr/lib/lib.b used by bc 65
/usr/lib/lib.l.a used by lex 379
/usr/lib/libxp.a used by lex 379
/usr/lib/uucp used by uucc 784
/usr/lib/uucp/config used by uucc 784
/usr/man/%L/man-0-9/*.book used by man 455
/usr/man/%L/whatis used by man 455
/usr/spool/.Sequence used by uucp 789
/usr/spool/cron used by cron 181
/usr/spool/cron/atjobs used by cron 181
/usr/spool/cron/crontabs used by cron 181
/usr/spool/cron/log used by cron 181
/usr/spool/cron/pid used by cron 181
/usr/spool/uucp/LOGFILE used by uucp 789
files (continued)
/usr/spool/uucppublic used by uucp 789
$HOME/.sh_history used by fc, history, r 302
$HOME/.exrc used by vi 833
$HOME/mbox used by mailx 426
used by make 410
$MAILRC used by mailx 426
$TMPDIR/pg* used by pg 542
/usr/man/%L/man-[0-9]/*.stftpt used by man 455
calendar 114
dead.letter used by mail 408, 409
used by mailx 419
ed.hup used by ed 279
extended attributes displaying 299
HOME/.profile used by tcsh login 671
used by the login shell 589
l.output used by lex 379
lex.yy.c used by lex 379
lib.a used by yacc 873
libxp.a used by yacc 874
MAILDIR used by mailx 426
mailrc used by mailx 421
MapName used by automount 29
mbox used by mail 409
used by mailx 411
pk$* used by unpack 781
queuedefs used by cron 180
remove old 625
rsh used by the sh command 589
sh_history 613
terminfo.src used by tic 741
y.output used by yacc 874
y.tab.c used by yacc 874
y.tab.h used by yacc 874
Files section description of 5
filter numbering lines in a file 487
filter (continued)
   passing small amounts to 271
find
   group affiliation of invoking processes 344
     identical lines within files 151
     spelling errors 633
     user identity of invoking processes 344
find shell command 311
fixed records
   converting from variable records 252
fixed to variable-record conversion 253
flag options syntax 862
flipr environment variable
   used by mailx 417, 424
FLOAT
   C/C++ programs 81
     floating-point numbers 81
     select format of floating-point numbers 81
floating-point registers
   displaying values of 231
FNR built-in variable for awk 39
fold shell command 319
folder environment variable
   used by mailx 414, 417, 424
for loop
   exiting from, in a shell script 77
for shell subcommand 590
format files in paginated form 543
Format section
   explanation of 1
forward retrieve function of OMVS command 915
FPATH environment variable
   description of 607
fpath search 596
free space
   displaying amount of 255
fullword
   definition of, for vi 812
func subcommand for dbx command 216
function
   changing 216
   explanation of 60
   listing 249
   moving down the stack 214
   moving up the stack 247
   printing tracing information for 245
   unsetting values and attributes of 781
function key
   customizing settings for 914
   deactivating 915
   displaying current settings 919
   list of defaults 916
   setting
     controlling display of 915
   setting up
     to control display of the function key settings 915
     to enter subcommand mode 916
     to enter TSO/E command mode 916
     to return from subcommand mode to shell 916
     to scroll data backward 916
     turning off display of 919
function shell subcommand 596
functions shell command 320
fuser shell command 321
FWDRETR function key for OMVS command 915
FWDRETR option of OMVS command 915
G
gencat shell command
   preprocessing message source files 458
general-purpose registers
   displaying values of 231
generate
   file names 604
   programs for lexical tasks 378
generate source dependency information
   makedepend 447
get
   configuration values 325
   contents of archive files 176
   messages 456
getconf shell command 325
getfacl shell command 329
getopts shell command 331
gid shell variable
   description of 700
glob characters 604
glob patterns 604
globalization
   explanation of 965
gmacs
   enabling, with the EDITOR environment variable 606
GMT (Greenwich Mean time)
   used by the TZ environment variable 987
GONUMBER
   C/C++ programs 82
     debugging 82
     improved performance 82
gotoi subcommand for dbx command 217
Greenwich Mean Time (GMT)
   used by the TZ environment variable 987
grep shell command 333
group
   changing 484
   setting 886
group affiliation
   finding 344
   returning 344
group owner
   changing 129
   setting 129
group recipe
   explanation of 428
group shell variable
   description of 700
GROUP tcsh environment variable
   description of 711
H
HALFSCR function key for OMVS command 915
HALFSCR subcommand of OMVS command 918
hangup 614
hash shell command 337
head shell command 338
header environment variable 
  used by mailx 424
header line 411
HELP function key for OMVS command 915
help information 
  refreshing 915
  scrolling 
    backward 914, 918
    forward 914, 916, 918, 919
  half a screen forward 915, 918
help subcommand for dbx command 217
HELP subcommand of OMVS command 918
here-document 604
HFS
  invoking 
    vfs_pfsctl HFS functions 157
hide 
  data entered on the shell command line 915, 918
  OMVS command input area 915, 918
HIDE function key for OMVS command 915
HIDE subcommand of OMVS command 918
histchars shell variable 
  description of 700
histdups shell variable 
  description of 701
HISTFILE environment variable 
  description of 607
  used by fc, history, r 302
histfile shell variable 
  description of 701
histlits shell variable 
  description of 701
history 
  editing 615
history file 
  processing 302
  truncating the 302
history list 
  displaying commands in a 218
  processing for commands 302
history shell command 341
history shell variable 
  description of 701
history storage file (sh_history) 613
history subcommand for dbx command 218
HISTSIZE environment variable 
  description of 607
  used by fc, history, r 302
hold buffer 572
hold environment variable 
  used by mailx 412, 424
home directory 607
HOME environment variable 
  description of 607
  used by cd 119
  used by crontab 183
  used by mail 408
  used by mailx 422
  used by vi 828, 833, 834
home shell variable 
  description of 701
HOME tcsh environment variable 
  description of 711
HOME/.profile file 
  used by tcsh login 671
HOST tcsh environment variable 
  description of 711
HOSTTYPE tcsh environment variable 
  description of 711
HPATH tcsh environment variable 
  description of 711
Huffman coding 
  compressing files with 780
  uncompressing files 538
hyphen 
  explanation of 1

I
iconv shell command 342
id shell command 344
identifier 
  displaying qualifications of 251
  identifier (awk variable) 35
identify shell names 761
if shell subcommand 590
IFS environment variable 
  description of 607
  used by read 561
  used by sh 590
ignore environment variable 
  used by mailx 411, 424
ignore subcommand for dbx command 218
ignoreeof shell variable 
  description of 701
ignoreeof environment variable 
  used by mailx 424
illegal byte sequence 
  in DBCS strings 8
implicitcd shell variable 
  description of 701
improved debugging 
  GONUMBER 82
improved performance 
  XPLINK 88
in shell subcommand 595
in/out file archives 
  copying 176
indent environment variable 
  used by mailx 425
indentprefix environment variable 
  used by mailx 420, 425
inetd daemon 
  handling of requests by uucpd 790
inference rules 
  used by make 437
infocmp shell command 348
input 
  passing small amounts to filter or file 271
input file 
  concatenating lines 502
  displaying 539
  printing 397
input mode 411
inputmode shell variable 
  description of 701
insert mode 807

instruction
  displaying 221
  running 241

instruction register
  displaying values of 231

interactive shell 589, 672

Interactive System Productivity Facility 897

interactive text editor (vi) 807

intercept
  abnormal conditions and interrupts 750
  signals 750

interdependent file
  maintaining 427

internal field separator 607

Interprocedural Analysis (IPA) optimization
  explanation of 87

interprocess communication facility status
  reporting the 354

interrupt
  trapping abnormal 750

invalid byte sequence
  in DBCS strings 8

invoke
  BPXBATCH
    with OSHELL 925
      editor 215
      HFS functions 157
      shell 588
      TSO/E command mode 919
      utilities, ignoring the SIGHUP signal 492
      z/OS shell 910
  invokes
    vfs_pfsctl HFS functions 157

IPA
  enabling 86, 450
  explanation of 86, 450

IPA (Interprocedural Analysis) optimization
  explanation of 87

ipcrm shell command 353

ipcs shell command 354

ISHELL TSO command 886

ISPF (Interactive System Productivity Facility)
  browsing files
    with the obrowse shell command 493
  editing files
    with the oedit shell command 498
  entering TSO/E commands from 883
  invoking the shell
    with ISHELL 886

italic typeface
  explanation of 1, 2

J

JCL (job control language)
  example of, using the BPXCOPY program 962

job
  ending 364
  moving
    from background to foreground 304
    to background 66
  (continued)
    restarting a suspended 304
    returning list of, in current session 360
    running in background 66
    scheduling background 182
    waiting for it to end 837

jobs shell command 360

join shell command 362

join two databases 362

K

keep environment variable
  used by mailx 425

keepsave environment variable
  used by mailx 412, 425

key
  sorting 630

keyboard
  navigation 1009
  PF keys 1009
  shortcut keys 1009

kill shell command 364

L

l.output file
  used by lex 379

LALR(1) grammar
  converting 872

LANG environment variable
  converting description of 87

LANG tcsh environment variable
  description of 711

LC_ALL environment variable 3, 965

LC_COLLATE environment variable 965

LC_CTYPE environment variable 965

LC_CTYPE tcsh environment variable
  description of 711

LC_MESSAGES environment variable 3, 965

LC_MONETARY environment variable 965

LC_NUMERIC environment variable 965

LC_SYNTAX environment variable 965

LC_TIME environment variable 965

ld shell command 368

Lempel-Ziv compression
  compressing data with 155
  uncompressing data 770

let shell command 376

lex shell command 378

lex.yy.c file 379

lexical analyzer 378

lexical syntax
  reading description of 378

lexical tasks
  generating programs for 378

LIBPATH environment variable
  description of 607
  used by c89/cc/c++ 87

library
  creating 15
  maintaining 15
  making 442
library of objects
  displaying symbol table 489
liby.a file
  used by yacc 873
libyxp.a file
  used by yacc 874
Limits section
  explanation of 6
line
  breaking into shorter lines 320
  changing next line to be displayed 222
  numbering, in a file 487
  reading from standard input 560
line editor (ex) 807
line shell command 381
LINENO environment variable
  description of 607
lines
  counting 839
LINES environment variable
  description of 607
  used by pg 540, 542
  used by vi 831
LINES option of OMVS command 914
LINES tcsh environment variable
  description of 711
link
  creating, for files 383
link shell command 381
link-edit
  z/OS C and z/OS C++ object files 79
links
  creating 381
list
  active procedures and functions 249
  file attributes 399
  file names 399
  files in directories
    with ISHELL 886
    with OSHELL 925
  instructions in program 221
  process IDs 321
list mode 510
list subcommand for dbx command 219
listfiles subcommand for the dbx command 220
listflags shell variable
  description of 702
listfuncs subcommand for dbx command 220
listi subcommand for dbx command 221
listjobs shell variable
  description of 702
listlinks shell variable
  description of 702
listmax shell variable
  description of 702
listmaxrows shell variable
  description of 702
ln shell command 382
load characteristics
  displaying 221
local environment
  defining 391
  local spawn
    BPXBATS1 884, 949
locale
  converting source definitions for categories 391
  displaying information about 386
  giving it control over a category 965
  switching 9
locale shell command 386
localedef shell command 391
localization
  categories of 965
  explanation of 965
Localization section
  explanation of 3
LOCPATH environment variable
  description of 607
log information
  displaying about UUCP events 793
log messages 393
logger shell command 393
logging in 589, 671
login accounting information
  storing 980
login information
  displaying 842
login name
  returning 396
login password and password phrase
  changing the 502
login shell
  description of 589, 671
  system profile for the 613
  truncating history files 302
  user profile for the 613
loginsh shell variable 702
LOGNAME environment variable
  description of 607
  used by crontab 183
  used by logname 397
  used by mailx 422
logname shell command 396
logout shell variable 702
loop
  exiting from, in a shell script 77
lowercase
  converting to uppercase 253
lp shell command 397
LPDEST environment variable
  used by lp 397, 398
lpstat shell command 399
ls
  in a sysplex 405
  ls shell command 399

M
MACHINE tcsh environment variable
  description of 711
macro definitions 433
macro modifiers 433
magic file format
  used by the file command 305
magic number
  #! 955
mail
mail (continued)
  reading 406
  sending 406
  sending and receiving 409
MAIL environment variable
  used by mailx 422
mail shell command 406
mail shell variable
description of 702
MAILCHECK environment variable
description of 607
MAILDIR environment variable
  used by mailx 422
MAILER environment variable
  used by calendar 113
MAILPATH environment variable
description of 608
MAILRC environment variable
  used by mailx 422
mailrc file
  used by mailx 421
mailserv environment variable
  used by mailx 425, 427
mailx environment variable
  used by mailx 417
mailx shell command 409
maintain
  library archives 15
  program-generated and interdependent files 427
make
  directories
    for each named directory argument 459
    with the MKDIR command 888
  FIFO special files 461
  libraries 442
make shell command
  conditional expression 443
makedepend
  generate source dependency information 447
makedepend shell command 447
makefile
  contents of 432
MAKEFLAGS environment variable
  used by make 442, 444
MAKESTARTUP environment variable
  used by make 428, 442, 444
man page
  displaying 453
man shell command 453
manage
  file caches 319
manipulate
  repeated lines 773
  tar archive files 664
MANPAGER environment variable
  used by man 455, 456
MANPATH environment variable
description of 608
  used by man 455, 456
map subcommand for dbx command 221
MapName file
  used by automount 29
mark name 812
matchbeep shell variable (continued)
description of 702
matching strings
  searching for 333
MBOX environment variable
description of 608
mbox file
  used by mail 409
  used by mailx 411
memory
displaying 211
merge
  corresponding or subsequent lines of files 502
mesg shell command 456
message
  allowing 456
  broadcasting a 838
  header line 411
  logging 393
  receiving 456
  refusing 456
  sending to other users 845
message catalog
  creating 322
  displaying 267
  displaying messages from 268
  editing 322
  modifying 322
  piping from mkcatdefs to gencat 570
message queue
  removing 353
metarules 437
metoo environment variable
  used by mailx 425
migrating
  HFS file systems to zFS
    with bpxwmigf shell command 74
    with bpxwmigf system REXX command 931
    with BPXWMIGF TSO/E command 886
mkcatdefs shell command 458
mkdir shell command 459
MKDIR TSO/E command 888
mkfifo shell command 461
mknod shell command 462
MKNOD TSO/E command 889
mode
  changing 132
  command 411
  input 411
  modification time
    setting for destination files 166
modify
  message catalogs 322
more shell command
  creating tag files for the 187
mount
  a file system 468
  z/OS UNIX file system 886, 891
mount attributes
  changing
    from the shell 135
mount mode
  changing the 927
mount shell command 468
MOUNT TSO/E command  891
move
current function down the stack  214
current function up the stack  247
files  472
jobs from background to foreground  304
positional parameters  623
move subcommand for dbx command  222
movement commands (for vi)  811
MsgFile.h
mkcatdef output file  458
multihop name  786
multinode name  786
multiple volume support  178, 521
multiprocess debugging
  enabling or disabling  223
multproc subcommand for dbx command  223
mutex object
  display list of  244
  displaying list of  223
mutex subcommand for dbx command  223
mv shell command  472
MVS (Multiple Virtual Storage)
batch environment
  running shell scripts and z/OS XL C/C++ applications under 949
copying
data sets into z/OS UNIX file system directories  922
data sets into z/OS UNIX file system files  920
data sets to another member or file  897
sequential data sets into z/OS UNIX file system directories  922
sequential data sets into z/OS UNIX file system files  920
z/OS UNIX file system files to MVS data sets  903

N
ame of files
  listing  399
name, user
  displaying your  844
named pipe  462
national language system report  694
navigation
  keyboard  1009
newgrp shell command  484
newline
counting  839
next subcommand for dbx command  224
nexti subcommand for dbx command  225
NEXTSESS function key for OMVS command  915
NEXTSESS subcommand of OMVS command  918
NF built-in variable for awk  39
nice shell command  485
nl shell command  487
NLSPATH environment variable
  description of  608
nm shell command  489
NO function key for OMVS command  915
NOALARM function key for OMVS command  914
NOALARM option of OMVS command  910
NOAUTOCOMMAND of OMVS command  918
NOAUTOMOVE  894
NOAUTOSCROLL function key for OMVS command  914
NOAUTOSCROLL option of OMVS command  910
NOAUTOSCROLL subcommand of OMVS command  918
nobeep shell variable
  description of  702
noclobber shell variable
  description of  703
NODBCS option of OMVS command  912
NOECHO function key for OMVS command  915
NOECHO option of OMVS command  912
NOECHO subcommand of OMVS command  918
noglob shell variable
  description of  703
NOHIDE function key for OMVS command  915
NOHIDE subcommand of OMVS command  919
nohup shell command  492
nokanji shell variable
  description of  703
nonfunctional commands
  cancel  114
  lpstat  399
nonmatch shell variable
  description of  703
nonzero exit code
  returning  301
NOPFSHOW function key for OMVS command  915
NOPFSHOW option of OMVS command  917
NOPFSHOW subcommand of OMVS command  919
NOREBIND tcsh environment variable
  description of  712
NOSHAAREAS option of OMVS command  917
nostat shell variable
  description of  703
notify shell variable
  description of  703
NR built-in variable for awk  39
null command  150

O
object file
  displaying the symbol table of an  489
  loading for execution  225
  managing  427
  running with previous arguments  232
object library
  displaying symbol table  489
object manipulator commands (for vi)
  list of  815
object subcommand for dbx command  225
obrowse shell command  493
OBROWSE TSO/E command  896
obtain
  crontab entries  182
obtain options and their arguments  331
OCOPY TSO/E command  897
octal dump  494
od shell command  494
oedit shell command  498
OEDIT TSO/E command  901
OGET TSO/E command  903
OGETX TSO/E command  906
OLDPWD environment variable
  description of  608
OLDPWD environment variable (continued)
  used by cd 120
OMVS command
  list of subcommands 918
OMVS command input area
  hiding 918
  hiding or unhiding 915
  unhiding 919
OMVS interface
  running commands from the shell using the 757
onehop environment variable
  used by mailx 425, 427
online reference manual
  printing entries 453
  searching for entries 453
onload subcommand for dbx command 226
open file descriptors 288
open files
  displaying
    zlsosf system REXX command 931
    zlsosf TSO command 930
OPEN function key for OMVS command 915
OPEN subcommand of OMVS command 919
  operator
    control 595
    description of 595
    redirection 595
OPTARG environment variable
  used by getopt 332
OPTIND environment variable
  used by getopt 332
optional features 79
options
  explanation of 1
  obtaining from a list of parameters 331
  order of 1
Options section
  explanation of 3
OPUT TSO/E command 920
OPUTX TSO/E command 922
  order of items on command line 2
  order of options 1
OSHELL REXX exec 958
OSHELL TSO command 925
OSTEPLIB TSO/E command 926
OSTYPE tcsh environment variable
  description of 712
outfolder environment variable
  used by mailx 425
output file
  copying standard input to each 735
output stream
  cloning 735
output tags file
  used by ctags 188
  used by uptime 783
output, formatted
  writing 548
overlay commands 288
owd shell variable
  description of 703

P
pack shell command 500
page environment variable
  used by mailx 416, 425
pager environment variable
  used by mailx 417
PAGER environment variable
  used by man 455
paginated file
  formatting 543
  printing 543
parallel processing
  OpenMP environment variables 850
parameter
  positional
    description of 598
    setting 579
    shifting 623
    unsetting 579
  special
    description of 598
parameter substitution 598, 694
parent process
  returning to the 290
parse
  utility options 331
partitioned data set (PDS) 906
partitioned data set extended (PDSE) 906
pass
  command to shell for execution 238
  small amounts of input to filter or file 271
passwd shell command 502
password and password phrase
  changing the 502
paste shell command 502
patch shell command 506
PATH environment variable
  description of 608
  used by crontab 183
  used by vi 834
path name
  checking for validity and portability 509
  displaying 559
  returning
    directory components of 265
    nondirectory components of 50
path search 596
path shell variable
  description of 703
PATH tcsh environment variable
  description of 712
pathchk shell command 509
pattern
  rules for 594
  searching
    backward for a 203
    forward for a 203
pattern buffer 572
pax file format 973
pax shell command 510
pcat shell command 538
PDS (partitioned data set)
copying
PDS (partitioned data set) (continued)
copying (continued)
   members from MVS to files 922
   members to files 920
   z/OS UNIX file system directories or file to a 906
PDSE (partitioned data set extended)
copying
   members from MVS to files 922
   members to files 920
   z/OS UNIX file system directories or files to a 906
performance
   C/C++ programs
      FLOAT 81
      XPLINK 88
permission bits
   of files, setting 961
permissions 403
PF keys
   showing at the bottom of the screen 917
PFSHOW function key for OMVS command 915
PFSHOW option of OMVS command 917
PFSHOW subcommand of OMVS command 919
pg shell command 539
pid subcommand for dbx command 229
pipe
   creating 591
      output from mkcatdefs to gencat 570
piped file
   displaying 539
pipeline 591
pipes
   displaying
      zlsof system REXX command 931
      zlsof TSO command 930
pk$* file
   used by unpack 781
placeholder information in commands 2
Portability section
   explanation of 6
positional parameter 623
POSIX.1 standard parameter names 325
POSIX.2 standard parameter names 327
PPID environment variable
   description of 608
pr shell command 543
preprocess
   message source files 458
prevent changes to values of the name argument 562
PREVSESS function key for OMVS command 915
PREVSESS subcommand of OMVS command 919
print
   arguments 546
   expression values 228
   formatted output
      writing 548
   input files 397
   paginated files 543
   sections of online reference manuals 453
   terminal entries in the terminfo database 115
   terminfo database entries 348
   tracing information 245
print queue (continued)
   requests (continued)
      displaying status of 399
print shell command 546
print subcommand of for command 228
printenv shell command 547
printer
   sending files to 397
PRINTER environment variable
   used by lp 397, 398
printexitvalue shell variable
   description of 703
printf shell command 548
priorities of running processes
   changing 563
priority
   running commands at a different 485
procedure
   listing 249
   printing tracing information for 245
process
   changing priorities of running 563
   displaying
      status of 550
      time accumulated 742
   ending 364
   returning
      file-creation permission-code masks 766
      status of 550
      sending signals to 364
      setting
         file-creation permission-code masks 766
         resource limits 764
process IDs
   displaying 321
process list
   returning 360
processes
   displaying list of 229
   processing
      awk programs 34
      command history list 302
processor
   displaying 741
   processor time 742
program
   continuing execution
      from stopping point 239
   continuing execution without dbx control 210
   debugging 198
   delaying execution of 624
   deleting stops and traces from 210
   displaying
      declarations of components 249
      instructions 221
      load characteristics 221
   generating, for lexical tasks 378
   managing 427
   printing tracing information 245
   running
      object files 234
      program instructions 241
      source lines 240
      to next instruction 225

1044
program (continued)
running (continued)
  to next source line 224
  until return is reached 233
  with previous arguments 232
  stopping
  at a specific location 243
  when certain conditions are met 241
program counter address
changing 217
program file
displaying
  list of functions 220
program-generated file
maintaining 427
programming constructs 696
prompt
  continuation 608
  string 608
prompt environment variable
used by mailx 425
prompt shell variable
description of 707
prompt subcommand of for command 228
prompt2 shell variable
description of 703
prompt3 shell variable
description of 703
promptchars shell variable
description of 703
ps shell command 550
PS1 environment variable
description of 608
PS2 environment variable
description of 608
used by read 561
PS3 environment variable
description of 608
PS4 environment variable
description of 608
public directories (UUCP)
searching 796
pushdsilent shell variable
description of 704
pushdtohome shell variable
description of 703
PWD environment variable
description of 608
used by cd 120
pwd shell command 559
PWD tcsh environment variable
description of 712

Q
query
  ASCII/EBCDIC code pages for the terminal 126
  STREAM physical file system 160
queuedefs file
  used by cron 180
queuedefs file format 976
quick mode 250
quiet environment variable
used by mailx 425

quiet mode
turning on 808
quit
  sessions
    ending 915
    shell sessions 915, 919
QUIT function key for OMVS command 915
QUIT subcommand of OMVS command 919
quit subcommand of the for command 229
QUITALL function key for OMVS command 915
QUITALL option of OMVS command 919
QUITALL subcommand of OMVS command 919
quoting 597

R
r shell command 560
RACF (Resource Access Control Facility) 11
RANDOM environment variable
description of 608
read
  archive files 510, 664
  contents of UUCP configuration files 783
cpio archives 176
data 251
dbx subcommands from file 239
description of lexical syntax 378
electronic mail 409
  lines from standard input 560
  mail 406
read mode 511
read shell command 560
read/write lock objects
displaying list of 230
readonly shell command 562
readonly variable
used by vi 808, 825
readwritelock subcommand for dbx command 230
reason_code text
displaying
  bpxmtext shell command 67
  bpxmtext system REXX command 931
receive
electronic mail 409
  messages 456
recexact shell variable
description of 704
recipe line
  explanation of 428
recipes
  explanation of 432
recognize_only_executables shell variable
description of 704
record environment variable
used by mailx 410, 415, 417, 425
record separator character 39
record subcommand for the dbx command 231
recovery daemon
  for vi 296
redirection 3, 603
redirection operator 595
reenter commands 302
reentrancy 110
reference manual
reference manual (continued)  
online  
printing entries 453, searching for entries 453  
refresh  
data 915, help information 915  
REFRESH function key for OMVS command 915  
refuse  
messages 456  
regexp 945, 946  
registers subcommand for dbx 231  
regular expression  
examples 945, 946, explanation of 945, 946, features that apply to z/OS shell commands 945, 946, matching 333, supported by awk 36, used in ex 831  
Related Information section  
explanation of 6  
remote site  
running commands on 802, transferring data to 983  
remote system  
copying files to users on 801  
remote systems  
connecting to, with the uucico daemon 785  
REMOTEHOST tcsh environment variable  
description of 712  
remount  
specified file systems 927  
remove  
alias definitions 767, aliases 247, arguments 567, attributes of shell variables 781, attributes of variables and functions 781, breakpoints at addresses 208, crontab entries 183, directories 569, directory entries 567, 776, duplicate files 773, files 567, 776, information from executable files 640, message queues 353, old files 625, reverse line feeds 149, semaphore sets 353, shared memory identifiers 353, stops from programs 207, 210, traces from program 210, trailing part of file names 265, values of variables and functions 781, variables 247  
remove ACLs  
setfacl 583  
rename files 472  
renice shell command 563  
REPLY environment variable  
used by read 561  
replyall environment variable  
used by mailx 425, 427  
report  
report (continued)  
interprocess communication facility status 354, repeated lines in a file 773, request (file transfer)  
processing, with the uucico daemon 785, rerun subcommand for dbx command 232, reset  
ASCII/EBCDIC code pages for the terminal 126, Resource Access Control Facility (RACF) 11, restart suspended jobs 304, restricted shell 589  
retrieve  
saved input lines by going backward 916, saved input lines by going forward 915  
RETRIEVE function key for OMVS command 916, return  
directory components of path names 265, file mode creation masks 766, from . (dot) scripts 565, from shell functions 565, from subcommand mode to shell session 919, group affiliation of invoking processes 344, list of jobs in current session 360, login names 396, nonzero exit codes 301, path name of working directories 559, process status 550, to shell mode from TSO/3270 passthrough mode 913, to the parent process 290, to TSO/E 290, user ID of person who entered commands 396, user identity of invoking processes 344  
RETURN function key for OMVS command 916, return shell command 565, return subcommand for dbx command 233, return values of 0 752, reverse line feed  
removing the 149  
REXX  
OSHELL 958, REXX system commands 931, RLENGTH (awk built-in variable) 41, rlogin requests  
handling 346, rlogind program 346, 565, rm shell command 567, rmdir shell command 569, rmstar shell variable  
description of 704, root directory  
changing 138, root file system  
setting up directories for the 886, rprompt shell variable  
description of 704, rsh file  
description of 589, RSTART (awk built-in variable) 41, run  
commands  
after building an argument list 869, at a different priority 485, at a specified time 24, at specified dates and times 180
run (continued)
commands (continued)
on remote sites 802
when system is not busy 51
with the exec command 288
debug programs 198
executable files
with the BPXBATCH program 949
files, with the ISHELL command 886
files, with the OSHELL command 925
object files with previous arguments 232
program instructions 241
programs 234
shell scripts
with the BPXBATCH program 949
source lines 216, 240
run subcommand for dbx 234
rncat shell command 570
running processes
changing priorities of 563
runtime macros 434

S
save environment variable
used by mailx 425
save messages 393
saved input line
retrieving by going backward 916
retrieving by going forward 915
savedirs shell variable
description of 704
savehist shell variable
description of 704
SBCS mode
specifying the 912
scale value 54
sched tcsh shell variable
description of 704
schedule
background jobs 182
screen editor (vi) 807
screen environment variable
used by mailx 416, 426
script shell command 570
scroll
automatic
controlling 914, 918
data 919
data backward 914, 916, 918
data forward 914, 918
data half a screen forward 915, 918
help information backward 914, 918
help information forward 914, 916
help information half a screen forward 915
SCROLL function key for OMVS command 916
SCROLL subcommand of OMVS command 919
scrolling commands (for vi) 810
search
backward for patterns 203
directories 248
files for text strings 886, 925
for strings 333
forward for patterns 203
search (continued)
public UUCP directories 796
search path 596
search rules 596
SECONDS environment variable
description of 609
sections
meaning of, in command descriptions 1
sed noninteractive stream editor
starting the 572
sed shell command 572
select format of floating-point numbers
FLOAT 81
select loop
exiting from, in a shell script 77
select shell subcommand 595
semaphore set
removing 353
send
electronic mail 409
files to printer 397
mail 406
messages
to other users 845
paginated files to printer 543
signals to processes 364
sending to IBM
reader comments xix
sendmail environment variable
used by mailx 426, 427
sendmail environment variable
used by mailx 426, 427
sequential data set
copying to files 920, 922
serviceability
C/C++ programs
GONUMBER 82
session
specifying number to be started 917
starting
in ex mode 807
in vi mode 807
switching
to the next higher-numbered one 915, 918
to the previous (lower-numbered) session 915, 919
SESSION option of OMVS command 917
session, returning list of jobs in 360
set
ASCII/EBCDIC code pages for the terminal 126
command options 579
commands to be run at a specified time 24
export attributes for variables 293
file mode creation masks 766
positional parameters 579
priorities of running processes 563
process limits 764
STREAM physical file system 160
terminal options 641
terminal tab stops 656
set ACLs
setfacl 583
set option variables 827
set shell command 579
set subcommand for dbx command 234
set up directories for the root file system 886
existing groups 886
existing users 886
setfacl shell command 583
sh shell command
rsh file 589
sh subcommand for dbx command 238
sh_history file 613
SHAREAS option of OMVS command 917
shared file system
changing file system mount attributes 135
displaying amount of free space 255
mounting a file system 468
using df 257
using ls 405
shared memory identifier
removing 353
shedit shell command 615
shell
access to, giving users 925
alias command, and the 11
archive 612
arguments
evaluating 284
arrays 605
command lines 11
calendar 590
commands
running from TSO/E sessions 884, 958
running from TSO/E sessions, with OSHELL 958
comments
displaying variables 761
editing
interactive 615
ending 290
entering TSO/E commands from 883
evaluating
arguments 284
expressions 150
execution environment
removing aliases from 767
expressions
evaluating 150
functions
returning from 565
giving TSO/E users access to 925
identifying names 761
invoking 588
keywords 11
program
running in a separate address space 917
running in the TSO/E address space 917
removing attributes of shell variables 781
reserved word commands 592
returning
functions 565
running
programs in a separate address space 917
programs in a TSO/E address space 917
TSO/E commands from the 752, 757
scripts
exits from loops in a 77
running from TSO/E sessions, with BPXBATCH 884
shell (continued)
scripts (continued)
running from TSO/E sessions, with OSHELL 958
running, with the BPXBATCH program 949
running, with the dot command 266
skipping to the next iteration of a loop 161
sessions
closing 914, 918
ending 919
returning from subcommand mode 919
starting 915, 919
variables
displaying 761
removing attributes of 781
rules for 605
z/OS UNIX
giving TSO/E users access to 925
shell command
chmount 135
mount 468
skulker 625
unmount 777
shell command line
hiding data so secure data can be entered 915, 918, 919
SHELL environment variable
description of 609
used by awk 51
used by crontab 183
used by ed 278
used by mailx 417
used by make 444
used by vi 826, 834
shell mode
returning to, from TSO/3270 passthrough mode 913
shell pre-defined aliases
integer 353
shell predefined aliases
autoload 27
functions 321
history 341
stop 637
suspend 655
shell redirection
automatic code set conversion 993
shell tcsh shell variable
description of 704
shell variable
displaying
names of 579
values of 579
shift out
used in DBCS strings 8
shift positional parameters 623
shift shell command 623
SHLVL tcsh environment variable
description of 712
shlvl tcsh shell variable
description of 704
short circuit evaluation 38
shortcut keys 1009
show
arguments
of programs 204
show (continued)
  attributes and contents of a symbolic link 925
  attributes and contents of symbolic links 886
  currently exported variables 293
  declaration of program components 249
  differences between two files 257
  elapsed time for a command 741
  environment variables 283
  file attributes 886, 925
  first part of files 338
  information about locales 386
  instructions in program 221
  lines common to two files 151
  list of active program and functions 249
  list of files
    of module 220
  memory 211
  names of
    variables in procedures 214
  path name of working directories 559
  process status 550
  processors 741
  qualifications
    of given identifiers 251
    of symbols 250
  status of print queues 399
  system time accumulated by commands 742
  terminal names 560
  user time accumulated by the shell 742
  values of
    variables in procedures 214
showto environment variable
  used by mailx 426
SIGHUP signal
  ignored when utility is invoked 492
sign environment variable
  used by mailx 419, 426
Sign environment variable
  used by mailx 426
signal
  intercepting 750
  sending to processes 364
  trapping
    starting 207
    stopping 218
signal handling 694
simple command 595
single-byte character set (SBCS)
  when you must use 7
single-byte characters
  converting 342
site
  transferring data to remote 983
skip subcommand for dbx 239
skulker shell command 625
sleep shell command 624
socket
  identifying file types 403
sockets
  displaying
    zlsof system REXX command 931
    zlsof TSO command 930
sort
files
spell shell command 633
spell errors
looking for 633
split
files 635
text file 184
split shell command 635
spool directory
/usr/spool/uucp 981
standard environment variables (stdenv)
allocating as files for passing input
using the BPXBATCH command 884
using the BPXBATCH program 950
standard error (stderr)
allocating as files for passing input
using the BPXBATCH command 884
using the BPXBATCH program 950
standard input (stdin)
allocating as files for passing input
using the BPXBATCH command 884
using the BPXBATCH program 950
standard output (stdout)
allocating as files for passing input
using the BPXBATCH command 884
using the BPXBATCH program 950
closing 604
copying
data read from 748
to each output file 735
explanation of 3
reading 116
reading lines from 560
step subcommand for dbx command 240
stepi subcommand for dbx command 241
STEPLIB environment variable
description of 609
stop
dbx debug session 229
pending UUCP transfers 798
program at a specific location 243
removing from program 210
removing from source lines 207
shell 290
stop shell command 637
stop subcommand for dbx command
displaying 240
stopi subcommand for dbx command 243
STREAM physical file system
set and query
configstrm shell command 160
string
displaying in a binary file 638
searching for 333
strings shell command 637
strip shell command 640
stty shell command 641
stub commands
cancel 114
cu 188
lpstat 399
su shell command 648
SUBCOMMAND function key for OMVS command 916
subcommand mode
setting up
function key to enter 916
setting up function key to return from 916
submit
  batch jobs
    using the BPXBATCH command 884
z/OS batch jobs that run shell commands
  using the BPXBATCH command 884
submit shell command 651
subscribe-in-array condition 37
substitute
  commands 602
directories 598
sum shell command 653
summarize
  use of file space 269
summary of changes
  z/OS UNIX
    as updated February, 2015 xxiii
V2R2 xxii
V2R3 xx
Summary of changes xxiii
suppress command numbers 303
suspend program execution 624
suspend shell command 655
switch
  locales 9
to the next higher-numbered session 915, 918
to the previous (lower-numbered) session 915, 919
symbol
  changing interpretation of 206
displaying qualifications of 250
symbol table
  displaying the 489
symbol table used in awk 37
symbolic link
  displaying attributes and contents of 886, 925
ln 384
symbolic links
  SYMLINK
    linkname 960
SYMPATH 960
symlinks tcsh shell variable
description of 709
SYMTAB symbol table 37
synopsis of dbx commands
displaying 217
syntax
  explanation of 1
syntax, lexical
  reading description of 378
SYSEXEC environment variable
  used by tsocmd 758
syplex
  moving file systems in a sysplex 893
  unmounting a file system 777
SYSPROC environment variable
  used by tsocmd 758
SYSROOT
dummy file system 894
system control registers
  displaying values of 231
System REXX commands
  bpxmtext 931
  bpxwmigf 931
  zlsf 931
sysvar shell command 655
T
  tab
    compressing from spaces 771
    expanding to spaces 290
tab stop
  setting 656
tabs shell command 656
tag files
  creating 187
tags file format 977	
tail shell command 658
talk
  to another user 662
talk shell command 661
tape archive 664
tar archive files
  manipulating 664
tar file format 978	
tar shell command 663, 664
target 429
tcsh
  built-in commands 695
call
  command execution 687
  command syntax 679
  programming constructs 696
  signal handling 694
tcsh environment variable
  _TAG_REDIR_ERR=BIN
description of 713
  _TAG_REDIR_ERR=TXT
description of 712
  _TAG_REDIR_IN=BIN
description of 712
  _TAG_REDIR_IN=TXT
description of 712
  _TAG_REDIR_OUT=BIN
description of 712
COLUMNS
description of 711
DISPLAY
description of 711
EDITOR
description of 711
GROUP
description of 711
HOME
description of 711
HOST
description of 711
HOSTTYPE
description of 711
HPATH
description of 711
LANG
description of 711
LC_CTYPE
description of 711
LINES
description of 711
MACHTYPE
description of 711
tcsh environment variable (continued)
  NOREBIND
    description of 712
  OSTYPE
    description of 712
  PATH
    description of 712
  PWD
    description of 712
  REMOTEHOST
    description of 712
  SHLVL
    description of 712
  TERM
    description of 712
  USER
    description of 712
  VENDOR
    description of 712
  VISUAL
    description of 712

tcsh files 713

tcsh shell
  Tcsh shell command 714
  alloc 716
  bindkey 716
  builtins 717
  @ (at) shell command 714
  alias shell command 11
  automatic, periodic, and timed events 694
  bg shell command 66
  break shell command 77
  cd shell command 118
  colon (:) shell command 150
  command-line editor 672, 673, 675
  colon (:) shell command 150
  editing
  eval shell command 284
  exec shell command 288
  exit shell command 290
  fg shell command 304
  history shell command 341
  jobs shell command 361
  kill shell command 364
  ls–F shell command 727
  National language system report 694
  newgrp shell command 484
  nice shell command 485
  nohup shell command 492
  printenv shell command 547
  problems and limitations 713
  set shell command 579
  status reporting 693
  stop shell command 637
  substitutions 680
  suspend shell command 655
  time shell command 741
  umask shell command 766
  unalias shell command 767
  unset shell command 782
  wait shell command 837

tcsh shell command
  alloc 716
  bindkey 716
  builtins 717
  @ (at) shell command 714
  alias shell command 11
  automatic, periodic, and timed events 694
  bg shell command 66
  break shell command 77
  cd shell command 118
  colon (:) shell command 150
  command-line editor 672, 673, 675
  colon (:) shell command 150
  editing
  eval shell command 284
  exec shell command 288
  exit shell command 290
  fg shell command 304
  history shell command 341
  jobs shell command 361
  kill shell command 364
  ls–F shell command 727
  National language system report 694
  newgrp shell command 484
  nice shell command 485
  nohup shell command 492
  printenv shell command 547
  problems and limitations 713
  set shell command 579
  status reporting 693
  stop shell command 637
  substitutions 680
  suspend shell command 655
  time shell command 741
  umask shell command 766
  unalias shell command 767
  unset shell command 782
  wait shell command 837

tcsh shell variable (continued)
  argv
    description of 698
  autocorrect
    description of 698
  autoexpand
    description of 698
  autoexpand
    description of 698
  autolist
    description of 698
  autologout
    description of 698
  autocorrect
    description of 698
  backslash
    description of 698
  cdprompt
    description of 698
  cdprompt
    description of 698
  correct
    description of 698
  cwd
    description of 698
  dextract
    description of 698
  dextract
    description of 698
  dirfile
    description of 699
  dirstack
    description of 699
  dunique
    description of 699
  echo
    description of 699
  echo_style
    description of 700
  edit
    description of 700
  fignore
    description of 700
  filec
    description of 700
  gid
    description of 700
  group
    description of 700
  histchars
    description of 700
  histdup
    description of 701
  histfile
    description of 701
  histfile
    description of 701
  histlit
    description of 701
  history
    description of 701
  home
    description of 701
  ignoreeof
    description of 701
  implicitcd
    description of 701
  inputmode
    description of 701
  listflags
tcsh shell variable (continued)
listflags (continued)
description of 702
listjobs
description of 702
listlinks
description of 702
listmax
description of 702
listmaxrows
description of 702
logins
description of 702
logout
description of 702
mail
description of 702
matchbeep
description of 702
nobeep
description of 702
noclobber
description of 703
noglob
description of 703
nokanji
description of 703
nonomatch
description of 703
nostat
description of 703
notify
description of 703
owd
description of 703
path
description of 703
printexitvalue
description of 703
prompt
description of 707
prompt2
description of 703
prompt3
description of 703
promptchars
description of 703
pushsilent
description of 704
pushd
description of 703
reexec
description of 704
recognize_only_executables
description of 704
rmstar
description of 704
rprompt
description of 704
savedirs
description of 704
savehist
description of 704
sched
tcsh shell variable
sched (continued)
description of 704
shell
description of 704
shlvl
description of 704
status
description of 704
symlinks
description of 709
tcsh
description of 700, 704
term
description of 704
time
description of 710
tperiod
description of 704
tty
description of 704
uid
description of 704
user
description of 704
verbose
description of 705
version
description of 706
visiblebell
description of 707
watch
description of 707
who
description of 707
wordchars
description of 707
tcsh tcsh shell variable
description of 704
tee shell command 735
template for commands 869
temporary files
/tmp/sh*
description of 613
remove 625
TERM environment variable
used by talk 662
used by vi 808, 830, 834
TERM tcsh environment variable
description of 712
term tcsh shell variable
description of 704
terminal
changing characteristics of 746
sending messages to a 845
setting, resetting, or querying ASCII/EBCDIC code pages
126
terminal entry
printing 115
terminal name
displaying 760
terminal options
displaying 641
setting 641
terminal tab stop
setting 656
terminfo database
printing terminal entries in the 115
terminfo database entries
comparing 346
compiling 740
printing 348
TERMININFO environment variable
used by talk 662
used by vi 834
terminfo.src file
used by tic 741
test condition 736
test shell command 736
text conversion
specifying 994
text editor
ex 285
using the ed 272
vi 807
text file
comparing two 257
concatenating 116
counting items in 839
displaying 116
finding information in 34
retrieving information from 34
showing differences between two 257
splitting 184
text insertion commands (for vi) 816
TFS file systems
unmounting 929
then shell subcommand 595
then statement
using null shell statement 150
thread
displaying information about 244
thread subcommand for dbx command 244
tic shell command 740
time
displaying 192
time program 741
time sharing option extensions 883
time shell command 741
time tcsh shell variable
description of 710
time zone
setting 987
times shell command 742
TMOU T environment variable
description of 609
TMP environment variable
used by exrecover 297
TMP_VI environment variable
used by vi 834
TMPDIR environment variable
description of 609
used by ed 278
used by exrecover 297
used by man 455
used by pg 542
used by vi 834
token
(token (continued)
description of 595
TOP function key for OMVS command 916
TOP subcommand of OMVS command 919
topolines environment variable
used by mailx 418, 426
topological sort 759
touch shell command 743
tperiod tcsh shell variable
description of 704
tput shell command 746
tr shell command 747
trace
removing from program 210
trace subcommand for dbx command
displaying 240
tracei subcommand for dbx command 246
tracing
activating
BPXTRACE TSO command 886
deactivating
BPXTRACE TSO command 886
turning on 246
tracing information
printing 245
tracked alias
creating a 337
transfers, UUCP
displaying status of pending 798
translate characters 748
trap
abnormal conditions and interrupts 750
signals
starting 207
stopping 218
trap shell command 750
ture shell command 752
TSO function key for OMVS command 916
TSO MOUNT
file system recovery 896
tso shell command 752
TSO subcommand of OMVS command 919
TSO/3270 passthrough mode
returning to shell mode 913
TSO/E (Time Sharing Option Extensions)
command mode
invoking the 919
commands
ISHELL 886
MKDIR 888
MKNOD 889
MOUNT 891
OBROWSE 896
OCOPY 897
OEDIT 902
OGET 903
OGETX 906
OMVS 909
OPUT 920
OPUTX 922
OSHELL 925
OSTEPLIB 926
UNMOUNT 927
entering commands from 883
TSO/E (Time Sharing Option Extensions) (continued)
giving users access to z/OS UNIX and shell 925
invoking BPXBATCH from the 925
returning to the 290
running
commands from the shell using the 752, 757
setting up function key to enter mode 916
TSOALLOC environment variable
used by tsocmd 758
tsocmd shell command 757
TSPROFILE environment variable
used by tsocmd 758
tsort shell command 759
tty shell command 760
tty tcsh shell variable
description of 704
turn off
automatic scrolling 918
type shell command 760
typeset shell command 761
TZ environment variable
description of 609
setting time zones with 987
used by cron 180
used by crontab 183
used by date 194
used by locale 391
used by ls 405
used by mail 408
U
unconvdef shell command 763
uid tcsh shell variable
description of 704
ulimit shell command 764
umask shell command 766
unalias shell command 767
unalias subcommand for dbx command 247
uname shell command 768
uncompress
data 876
Huffman-coded files 538
uncompress shell command 770
undo change 277
unexpand shell command 771
unhide
data entered on the shell command line 919
OMVS command input area 915, 919
uniq shell command 773
unique lines 773
Universal Time Coordinated (UTC)
used by the TZ environment variable 987
UNIX C shell 671
unlink shell command 776
unmount
a file system 777
TFS file systems 929
z/OS UNIX file system 886, 929
UNMOUNT 894
unmount shell command 777
UNMOUNT TSO/E command 927
unpack shell command 780
unprintable characters (continued)
displaying 116
unset
attributes of variables and functions 781
command options 579
positional parameters 579
values of variables and functions 781
unset shell command 781
unset subcommand for dbx command 247
until loop
exiting from, in a shell script 77
until shell subcommand 595
up subcommand for dbx command 247
update
data 915
upppercase
converting to lowercase 253
uptime shell command 783
Usage Notes section
explanation of 6
use subcommand for dbx command 248
user
sending messages to a 845
setting up 886
talking to another user 662
user ID
changing to superuser 648
user ID (UID)
returning 396
setting to owner 137
user identity
finding 344
returning 344
user interface
ISPF 1009
TSO/E 1009
user name
displaying your 844
USER tcsh environment variable
description of 712
user tcsh shell variable
description of 704
users
displaying information about current 842
usrspooluucp spool 981
usrspooluucpssouth 981
UTC (Universal Time Coordinated)
used by the TZ environment variable 987
utility
invoking, while ignoring the SIGHUP signal 492
parsing options 331
utmpx file format 980
uucp shell command 783
uucico daemon 784
UUCP
configuration file
reading contents of 783
copying files between systems 786
displaying
list of systems 795
status of transfers 798
events
displaying 793
searching public directories 796
UUCP (continued)
spool directory (/usr/spool/uucp) 981
transfers
displaying status of 798
starting or stopping 798
validating requests by the uucpd program 790
working files
command 981
data 981
execute 981
UUCP file transfer daemon 785
uucp shell command
processing file transfer requests 785
uucpd daemon
handling of uucp requests 790
uudecode shell command 791
uuencode shell command 792
uulog shell command 793
uuname shell command 795
uupick shell command 796
uustat shell command 797
uuto shell command 801
uux shell command
processing file transfer requests 785
uuxqt daemon 805

V
value
defining, for dbx variables 234
displaying, for registers 231
variable
assigning
attributes and variables to 761
values to 205
attributes 762
bc command, for the 54
built-in, for the bc shell command 54
condition
displaying list of 209
deleting 247
description of 598
displaying
currently exported variables 293
list of 761
names of variables in procedures 214
values of variables in procedures 214
environment
displaying 283
listing their attributes 761
parameters used by shell 598, 694
printing tracing information 245
readonly
used by vi 808, 825
setting export attributes 293
unsetting values and attributes of 781
used in awk 35
variable records
converting to fixed records 252
variable to fixed-record conversion 252
VENDOR tcsh environment variable
description of 712
verbose tcsh shell variable
description of 705
version tcsh shell variable
description of 706
vi command
cmd mode 807
editor initialization 832
entering ex command mode 819
file recovery daemon for 296
fullword
definition of 812
insert mode 807
insert mode commands 818
set option variables 827
starting sessions in vi mode 807
word
definition of 812
vi file recovery daemon 296
vi mode
absolute movement commands
list of 811
context-dependent movement commands 812
current position pointer 808
display conventions 808
object manipulator commands
list of 815
scrolling commands 810, 811
starting session in 807
text insertion commands 816
vi shell command
creating tag files for the 187
visiblebell tcsh shell variable
description of 707
VISUAL environment variable
description of 609
used by mailx 418, 420
used by shedit 616
VISUAL tcsh environment variable
description of 712

W
wait
for child process to end 837
for jobs to end 837
wait shell command 837
wall shell command 838
watch tcsh shell variable
description of 707
wc shell command 839
whatis subcommand for dbx command 249
whence shell command 841
where subcommand for dbx command 249
whereis subcommand for the dbx command 250
which subcommand for the dbx command 251
while loop
exiting from, in a shell script 77
while shell subcommand 590, 595
who shell command 842
who tcsh shell variable
description of 707
whoami shell command 844
wildcard characters 604
within-rule circular dependency 436
word
counting 839
word (continued)
definition of, for vi 812
description of 595
wordchars tcsh shell variable
description of 707
words
misspelled
looking for 633
working directory
changing
to directory 118
to previous working directory 119
displaying path name of the HOME environment variable 119
working files
format of UUCP 981
WRAPDEBUG option of OMVS command 917
write
archive files 510, 664
arguments to standard output 270
configuration values to standard output 325
cpio archives 176
data 251
displaying debugging information 912
formatted output 548
to other users 845
write mode 511
write shell command 844
writedown shell command 846

X
xargs shell command 869
xlc 848
xlC 848
xlC shell command 869
xlc_64 848
xlC_64 848
xlc_\_x 848
xlC_\_x 848
xlc/xlC shell command
environment variables 849
specifying
system and operational information to xlc/xlC 849
xlC++ 848
xlC++ shell command 869
xlC++_64 848
xlC++\_x 848
XPLINK
C/C++ programs 88
extra performance linkages 88
improved performance 88
xtrace 762

Y
y.output file
used by yacc 874
y.tab.c file
used by yacc 874
y.tab.h file
used by yacc 874
yacc compiler (continued)
using the 872
yacc shell command 872
YYDEBUG option 873

Z
z/OS C and z/OS C++ source files
using the c89 command to compile, assemble, and link-edit 79
z/OS shell
invoking the 910
z/OS UNIX
summary of changes for V2R1 as updated February, 2015 xxiii
summary of changes for V2R2 xxii
summary of changes for V2R3 xx
z/OS UNIX file system
browsing files in the 897
copying
between two files 897
data sets into MVS data sets 903
directories to PDS or PDSE 906
files to PDS or PDSE 906
MVS data set members 920, 922
creating 886
mounting 886, 891
unmounting 886, 929
z/OS UNIX shell commands
changed for UNIX03 1005
z/OS UNIX System Services
giving TSO/E users access to 925
managing functions with the ISPF shell
with ISHELL 886
setting up functions with the ISPF shell
with ISHELL 886
zcat shell command 876
zlsof shell command 877
zlsof REXX command 931
zlsof TSO command 930