

IBM XL Fortran for Linux, V15.1.5



Getting Started with XL Fortran for Little Endian Distributions

Version 15.1.5

IBM XL Fortran for Linux, V15.1.5



Getting Started with XL Fortran for Little Endian Distributions

Version 15.1.5

Note

Before using this information and the product it supports, read the information in "Notices" on page 35.

First edition

This edition applies to IBM XL Fortran for Linux, V15.1.5 (Program 5765-J10; 5725-C75) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

© Copyright IBM Corporation 1996, 2016.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this document	v
Who should read this document.	v
How to use this document.	v
How this document is organized	v
Conventions	vi
Related information	x
IBM XL Fortran information	x
Standards and specifications	xi
Other IBM information	xii
Technical support	xii
How to send your comments	xii
Chapter 1. Introducing XL Fortran	1
Commonality with other IBM compilers	1
Operating system and hardware support	1
A highly configurable compiler	2
Language standard compliance	3
Source-code migration and conformance checking	3
Utilities and commands.	3
Advance Toolchain 9.0 support	4
Program optimization	4
Shared memory parallelization	5
Diagnostic reports	6
Symbolic debugger support	6
Chapter 2. What's new for IBM XL Fortran for Linux, V15.1.5	7
Intrinsic procedures	7
Compiler options and directives	8
OpenMP support	8
Performance and optimization	10
Other enhancements	10
Chapter 3. Migration of your applications	13
Migrating from Linux for big endian distributions to Linux for little endian distributions	13
Resolving the compatibility issues of IPA object files	13
Chapter 4. Enhancements added in earlier releases	15
Enhancements added in Version 15.1.4	15
Compiler options and directives	15

CUDA Fortran support	16
Performance and optimization	17
Other enhancements	17
Enhancements added in Version 15.1.3	17
Language features	18
OpenMP support	19
Intrinsic procedures	19
Other enhancements	20
Enhancements added in Version 15.1.2	20
Fortran 2008 features	20
Language interoperability features.	21
OpenMP support	21
Intrinsic procedures	21
Commands	21
Compiler options	22
Chapter 5. Setting up and customizing XL Fortran	23
Using custom compiler configuration files	23
Chapter 6. Developing applications with XL Fortran	25
Compiler phases.	25
Editing Fortran source files	25
Compiling with XL Fortran	26
Invoking the compiler	26
Specifying compiler options	28
XL Fortran input and output files	29
Linking your compiled applications with XL Fortran	30
Dynamic and static linking	30
Running your compiled application	31
XL Fortran compiler diagnostic aids	31
Debugging compiled applications	32
Determining which level of XL Fortran is being used.	32
Appendix. Accessibility features for IBM XL Fortran for Linux	33
Notices	35
Trademarks	37
Index	39

About this document

This document contains overview and basic usage information for the IBM® XL Fortran for Linux, V15.1.5 compiler.

Who should read this document

This document is intended for Fortran developers who are looking for introductory overview and usage information for XL Fortran. It assumes that you have some familiarity with command-line compilers, basic knowledge of the Fortran programming language, and basic knowledge of operating system commands. Programmers new to XL Fortran can use this document to find information about the capabilities and features unique to XL Fortran.

How to use this document

Throughout this document, the `xlf` compiler invocation is used to describe the behavior of the compiler. You can, however, substitute other forms of the compiler invocation command if your particular environment requires it, and compiler option usage remains the same unless otherwise specified.

While this document covers information such as configuring the compiler environment, and compiling and linking Fortran applications using the XL Fortran compiler, it does not include the following topics:

- Compiler installation: see the *XL Fortran Installation Guide*.
- Compiler options: see the *XL Fortran Compiler Reference* for detailed information about the syntax and usage of compiler options.
- The Fortran programming language: see the *XL Fortran Language Reference* for information about the syntax, semantics, and IBM implementation of the Fortran programming language.
- Programming topics: see the *XL Fortran Optimization and Programming Guide* for detailed information about developing applications with XL Fortran, with a focus on program portability and optimization.

How this document is organized

Chapter 1, “Introducing XL Fortran,” on page 1 contains information about the features of the XL Fortran compiler at a high level.

Chapter 2, “What's new for IBM XL Fortran for Linux, V15.1.5,” on page 7 describes features and enhancements added to IBM XL Fortran for Linux, V15.1.5.

Chapter 3, “Migration of your applications,” on page 13 lists important considerations when you migrate your applications that were compiled with other versions of XL Fortran.

Chapter 4, “Enhancements added in earlier releases,” on page 15 describes enhancements added in earlier releases. These enhancements also apply to the current release.

Chapter 5, “Setting up and customizing XL Fortran,” on page 23 describes how to set up and customize the compiler according to your own requirements.

Chapter 6, “Developing applications with XL Fortran,” on page 25 consists of repeating cycles of editing, compiling, linking, and running.

Conventions

Typographical conventions

The following table shows the typographical conventions used in the IBM XL Fortran for Linux, V15.1.5 information.

Table 1. *Typographical conventions*

Typeface	Indicates	Example
lowercase bold	Invocation commands, executable names, and compiler options.	The compiler provides basic invocation commands, xlf , along with several other compiler invocation commands to support various Fortran language levels and compilation environments. The default file name for the executable program is a.out .
<i>italics</i>	Parameters or variables whose actual names or values are to be supplied by the user. Italics are also used to introduce new terms.	Make sure that you update the <i>size</i> parameter if you return more than the <i>size</i> requested.
<u>underlining</u>	The default setting of a parameter of a compiler option or directive.	nomaf <u>maf</u>
monospace	Examples of program code, reference to program code, file names, path names, command strings, or user-defined names.	To compile and optimize myprogram.f, enter: xlf myprogram.f -03.
UPPERCASE bold	Fortran programming keywords, statements, directives, and intrinsic procedures. Uppercase letters may also be used to indicate the minimum number of characters required to invoke a compiler option/suboption.	The ASSERT directive applies only to the DO loop immediately following the directive, and not to any nested DO loops.

Qualifying elements (icons and bracket separators)

In descriptions of language elements or programming models, this information uses icons and marked bracket separators to delineate segments of text as follows:

Table 2. *Qualifying elements*













Icon	Bracket separator text	Meaning
 F2008	Fortran 2008 begins /	The text describes an IBM XL Fortran implementation of the Fortran 2008 standard. ¹
F2008 	Fortran 2008 ends	

Table 2. Qualifying elements (continued)







Icon	Bracket separator text	Meaning
 	Fortran 2003 begins / Fortran 2003 ends	The text describes an IBM XL Fortran implementation of the Fortran 2003 standard, and it applies to all later standards. ¹
 	TS 29113 begins / TS 29113 ends	The text describes an IBM XL Fortran implementation of Technical Specification 29113, referred to as TS 29113. ¹
 	IBM extension begins / IBM extension ends	The text describes a feature that is an IBM XL Fortran extension to the standard language specifications.
 	CUDA Fortran begins / CUDA Fortran ends	The text describes CUDA Fortran, the CUDA Fortran support provided by IBM XL Fortran, or both.
 	GPU begins / GPU ends	The text describes the information that is relevant to offloading computations to the NVIDIA GPUs.

Note:

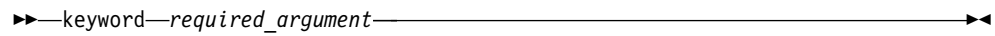
1. If the information is marked with a Fortran language standard icon or bracket separators, it applies to this specific Fortran language standard and all later ones. Otherwise, it applies to all Fortran language standards.

Syntax diagrams

Throughout this information, diagrams illustrate XL Fortran syntax. This section helps you to interpret and use those diagrams.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.
 - The  symbol indicates the beginning of a command, directive, or statement.
 - The  symbol indicates that the command, directive, or statement syntax is continued on the next line.
 - The  symbol indicates that a command, directive, or statement is continued from the previous line.
 - The  symbol indicates the end of a command, directive, or statement.
- Fragments, which are diagrams of syntactical units other than complete commands, directives, or statements, start with the  symbol and end with the  symbol.
- IBM XL Fortran extensions are marked by a number in the syntax diagram with an explanatory note immediately following the diagram.
- Program units, procedures, constructs, interface blocks and derived-type definitions consist of several individual statements. For such items, a box encloses the syntax representation, and individual syntax diagrams show the required order for the equivalent Fortran statements.

- Required items are shown on the horizontal line (the main path):



- Optional items are shown below the main path:



Note: Optional items (not in syntax diagrams) are enclosed by square brackets ([and]). For example, [UNIT=]u

- If you can choose from two or more items, they are shown vertically, in a stack. If you *must* choose one of the items, one item of the stack is shown on the main path.



If choosing one of the items is optional, the entire stack is shown below the main path.



- An arrow returning to the left above the main line (a repeat arrow) indicates that you can make more than one choice from the stacked items or repeat an item. The separator character, if it is other than a blank, is also indicated:



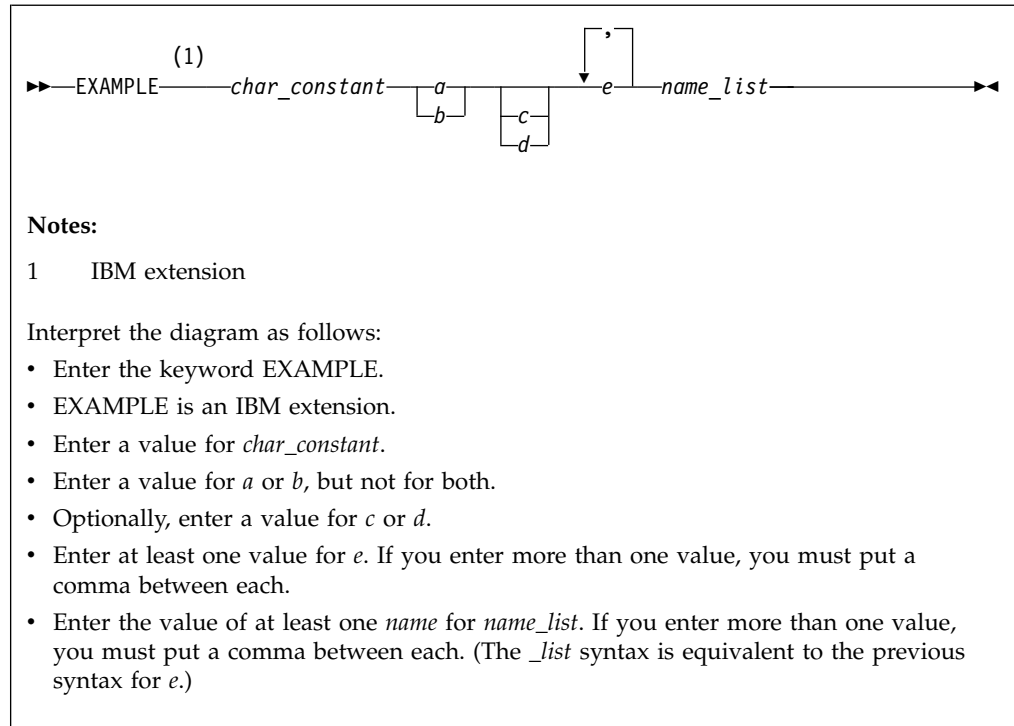
- The item that is the default is shown above the main path.



- Keywords are shown in nonitalic letters and should be entered exactly as shown.
- Variables are shown in italicized lowercase letters. They represent user-supplied names or values. If a variable or user-specified name ends in *_list*, you can provide a list of these terms separated by commas.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Sample syntax diagram

The following is an example of a syntax diagram with an interpretation:



How to read syntax statements

Syntax statements are read from left to right:

- Individual required arguments are shown with no special notation.
- When you must make a choice between a set of alternatives, they are enclosed by { and } symbols.
- Optional arguments are enclosed by [and] symbols.
- When you can select from a group of choices, they are separated by | characters.
- Arguments that you can repeat are followed by ellipses (...).

Example of a syntax statement

`EXAMPLE char_constant {a|b}[c|d]e[,e]... name_list{name_list}...`

The following list explains the syntax statement:

- Enter the keyword `EXAMPLE`.
- Enter a value for `char_constant`.
- Enter a value for `a` or `b`, but not for both.
- Optionally, enter a value for `c` or `d`.
- Enter at least one value for `e`. If you enter more than one value, you must put a comma between each.
- Optionally, enter the value of at least one `name` for `name_list`. If you enter more than one value, you must put a comma between each `name`.

Note: The same example is used in both the syntax-statement and syntax-diagram representations.

Examples in this information

The examples in this information, except where otherwise noted, are coded in a simple style that does not try to conserve storage, check for errors, achieve fast performance, or demonstrate all possible methods to achieve a specific result.

The examples for installation information are labelled as either *Example* or *Basic example*. *Basic examples* are intended to document a procedure as it would be performed during a basic, or default, installation; these need little or no modification.

Notes on the terminology used

Some of the terminology in this information is shortened as follows:

- The term *free source form format* often appears as *free source form*.
- The term *fixed source form format* often appears as *fixed source form*.
- The term *XL Fortran* often appears as *XLF*.

Related information

The following sections provide related information for XL Fortran:

IBM XL Fortran information

XL Fortran provides product information in the following formats:

- Quick Start Guide

The Quick Start Guide (`quickstart.pdf`) is intended to get you started with IBM XL Fortran for Linux, V15.1.5. It is located by default in the XL Fortran directory and in the `\quickstart` directory of the installation DVD.

- README files

README files contain late-breaking information, including changes and corrections to the product information. README files are located by default in the XL Fortran directory, and in the root directory and subdirectories of the installation DVD.

- Installable man pages

Man pages are provided for the compiler invocations and all command-line utilities provided with the product. Instructions for installing and accessing the man pages are provided in the *IBM XL Fortran for Linux, V15.1.5 Installation Guide*.

- Online product documentation

The fully searchable HTML-based documentation is viewable in IBM Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSAT4T_15.1.5/com.ibm.compilers.linux.doc/welcome.html.

- PDF documents

PDF documents are available on the web at <http://www.ibm.com/support/docview.wss?uid=swg27036672>.

The following files comprise the full set of XL Fortran product information:

Table 3. XL Fortran PDF files

Document title	PDF file name	Description
<i>IBM XL Fortran for Linux, V15.1.5 Installation Guide, GC27-6580-04</i>	install.pdf	Contains information for installing XL Fortran and configuring your environment for basic compilation and program execution.
<i>Getting Started with IBM XL Fortran for Linux, V15.1.5, SC27-6620-04</i>	getstart.pdf	Contains an introduction to the XL Fortran product, with information about setting up and configuring your environment, compiling and linking programs, and troubleshooting compilation errors.
<i>IBM XL Fortran for Linux, V15.1.5 Compiler Reference, SC27-6610-04</i>	compiler.pdf	Contains information about the various compiler options and environment variables.
<i>IBM XL Fortran for Linux, V15.1.5 Language Reference, SC27-6590-04</i>	langref.pdf	Contains information about the Fortran programming language as supported by IBM, including language extensions for portability and conformance to nonproprietary standards, compiler directives and intrinsic procedures.
<i>IBM XL Fortran for Linux, V15.1.5 Optimization and Programming Guide, SC27-6600-04</i>	proguide.pdf	Contains information on advanced programming topics, such as application porting, interlanguage calls, floating-point operations, input/output, application optimization and parallelization, and the XL Fortran high-performance libraries.
<i>Getting Started with CUDA Fortran programming using IBM XL Fortran for Linux, V15.1.5, GI13-3562-01</i>	getstart_cudaf.pdf	Contains detailed information about the CUDA Fortran support that is provided in XL Fortran, including the compiler flow for CUDA Fortran programs, compilation commands, useful compiler options and macros, supported CUDA Fortran features, and limitations.

To read a PDF file, use Adobe Reader. If you do not have Adobe Reader, you can download it (subject to license terms) from the Adobe website at <http://www.adobe.com>.

More information related to XL Fortran, including IBM Redbooks® publications, white papers, and other articles, is available on the web at <http://www.ibm.com/support/docview.wss?uid=swg27036672>.

For more information about the compiler, see the XL compiler on Power® community at <http://ibm.biz/xl-power-compilers>.

Standards and specifications

XL Fortran is designed to support the following standards and specifications. You can refer to these standards and specifications for precise definitions of some of the features found in this information.

- *American National Standard Programming Language FORTRAN, ANSI X3.9-1978.*
- *American National Standard Programming Language Fortran 90, ANSI X3.198-1992.*
- *ANSI/IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985.*
- *Federal (USA) Information Processing Standards Publication Fortran, FIPS PUB 69-1.*

- *Information technology - Programming languages - Fortran, ISO/IEC 1539-1:1991.* (This information uses its informal name, Fortran 90.)
- *Information technology - Programming languages - Fortran - Part 1: Base language, ISO/IEC 1539-1:1997.* (This information uses its informal name, Fortran 95.)
- *Information technology - Programming languages - Fortran - Part 1: Base language, ISO/IEC 1539-1:2004.* (This information uses its informal name, Fortran 2003.)
- *Information technology - Programming languages - Fortran - Part 1: Base language, ISO/IEC 1539-1:2010.* (This information uses its informal name, Fortran 2008. We currently provide partial support to this standard.)
- *Information technology - Further interoperability of Fortran with C, ISO/IEC TS 29113:2012.* (This information uses its informal name, Technical specification 29113, referred to as TS 29113. We currently provide partial support to this specification.)
- *Military Standard Fortran DOD Supplement to ANSI X3.9-1978, MIL-STD-1753* (United States of America, Department of Defense standard). Note that XL Fortran supports only those extensions documented in this standard that have also been subsequently incorporated into the Fortran 90 standard.
- *OpenMP Application Program Interface Version 3.1 (full support), OpenMP Application Program Interface Version 4.0 (partial support), and OpenMP Application Program Interface Version 4.5 (partial support),* available at <http://www.openmp.org>

Other IBM information

- *ESSL product documentation* available at http://www.ibm.com/support/knowledgecenter/SSFHY8/essl_welcome.html?lang=en

Technical support

Additional technical support is available from the XL Fortran Support page at http://www.ibm.com/support/entry/portal/product/rational/xl_fortran_for_linux. This page provides a portal with search capabilities to a large selection of Technotes and other support information.

If you cannot find what you need, you can send an email to compinfo@ca.ibm.com.

For the latest information about XL Fortran, visit the product information site at <http://ibm.biz/xlfortran-linux>.

How to send your comments

Your feedback is important in helping us to provide accurate and high-quality information. If you have any comments about this information or any other XL Fortran information, send your comments to compinfo@ca.ibm.com.

Be sure to include the name of the manual, the part number of the manual, the version of XL Fortran, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

Chapter 1. Introducing XL Fortran

IBM XL Fortran for Linux, V15.1.5 is an advanced, high-performance compiler that can be used for developing complex, computationally intensive programs, including interlanguage calls with C programs.

This section contains information about the features of the XL Fortran compiler at a high level. It is intended for people who are evaluating the compiler and for new users who want to find out more about the product.

Commonality with other IBM compilers

IBM XL Fortran for Linux, V15.1.5 is part of a larger family of IBM C, C++, and Fortran compilers. XL Fortran, together with XL C/C++, comprises the family of XL compilers.

These compilers are derived from a common code base that shares compiler function and optimization technologies for a variety of platforms and programming languages. Programming environments include IBM AIX®, IBM Blue Gene®/Q, and selected Linux distributions. The common code base, along with compliance with international programming language standards, helps support consistent compiler performance and ease of code portability across multiple operating systems and hardware platforms.

Operating system and hardware support

This section describes the operating systems and hardware that IBM XL Fortran for Linux, V15.1.5 supports.

IBM XL Fortran for Linux, V15.1.5, for little endian distributions supports the following operating systems:

- Ubuntu Server 14.04
- Ubuntu Server 14.10
- Ubuntu Server 16.04
- SUSE Linux Enterprise Server 12 (SLES 12)
- SUSE Linux Enterprise Server 12 Service Pack 1 (SLES 12 SP1)
- Red Hat Enterprise Linux 7.1 (RHEL 7.1)
- Red Hat Enterprise Linux 7.2 (RHEL 7.2)
- Red Hat Enterprise Linux 7.3 (RHEL 7.3)
- Community Enterprise Operating System 7 (CentOS 7)

See the README file and "Before installing XL Fortran" in the *XL Fortran Installation Guide* for a complete list of requirements.

The compiler, its libraries, and its generated object programs run on any IBM Power Systems™ server supported by your operating system distribution with the required software and disk space.

To exploit the various supported hardware configurations, the compiler provides options to tune the performance of applications according to the hardware type that runs the compiled applications.

A highly configurable compiler

You can use a variety of compiler invocation commands and options to tailor the compiler to your unique compilation requirements.

Compiler invocation commands

XL Fortran provides several commands to invoke the compiler, for example, `xl`, `xl90`, `xl95`, `xl2003`, `xl2008`, and `-CUDA Fortran xlcuf -CUDA Fortran`. Compiler invocation commands are provided to support most standardized Fortran language levels and many popular language extensions.

The compiler also provides corresponding `_r` versions of most invocation commands, for example, `xl_r`. The `_r` invocations instruct the compiler to link and bind object files to threadsafe components and libraries, and produce threadsafe object code for compiler-created data and procedures.

For more information about XL Fortran compiler invocation commands, see "Compiling XL Fortran programs" in the *XL Fortran Compiler Reference*.

Compiler options

You can choose from a large selection of compiler options to control compiler behavior. You can benefit from using different options for the following tasks:

- Debugging your applications
- Optimizing and tuning application performance
- Selecting language levels and extensions for compatibility with nonstandard features and behaviors that are supported by other Fortran compilers
- Performing many other common tasks that would otherwise require changing the source code

You can specify compiler options through a combination of environment variables, compiler configuration files, command line options, and compiler directive statements embedded in your program source.

For more information about XL Fortran compiler options, see "Summary of compiler options" in the *XL Fortran Compiler Reference*.

Custom compiler configuration files

The installation process creates a default compiler configuration file containing stanzas that define compiler option default settings.

If you frequently specify compiler option settings other than the default settings of XL Fortran, you can use makefiles to define your settings. Alternatively, you can create custom configuration files to define your own frequently used option settings.

For more information about using custom compiler configuration files, see "Using custom compiler configuration files" on page 23.

Language standard compliance

This topic describes the Fortran programming language specifications that IBM XL Fortran for Linux, V15.1.5 supports.

- Partial support for ISO/IEC TS 29113:2012 (referred to as the Technical Specification for further interoperability with C or TS 29113)
- Partial support for ISO/IEC 1539-1:2010 (referred to as Fortran 2008 or F2008)
- ISO/IEC 1539-1:2004 (referred to as Fortran 2003 or F2003)
- ISO/IEC 1539-1:1997 (referred to as Fortran 95 or F95)
- ISO/IEC 1539-1:1991(E) and ANSI X3.198-1992 (referred to as Fortran 90 or F90)
- ANSI X3.9-1978 (referred to as FORTRAN 77)

In addition to the standard language levels, XL Fortran supports the following language extensions:

- Language extensions to support vector programming
- Common Fortran language extensions defined by other compiler vendors, in addition to those defined by IBM
- Industry extensions that are found in Fortran products from various compiler vendors
- Extensions specified in SAA Fortran

See "Language standards" in the *XL Fortran Language Reference* for more information about Fortran language specifications and extensions.

Source-code migration and conformance checking

XL Fortran provides compiler invocation commands that instruct the compiler to inspect your application for conformance to a specific language level and warn you if constructs and keywords do not conform to the specified language level.

You can also use the `-qlanglvl` compiler option to specify a language level. If the language elements in your program source do not conform to the specified language level, the compiler issues diagnostic messages. Additionally, you can name your source files with common filename extensions such as `.f77`, `.f90`, `.f95`, `.f03`, or `.f08`, and then use the generic compiler invocations such as `xlf` or `xlf_r` to automatically select the language level appropriate to the filename extension.

Related information

 `-qlanglvl`

Utilities and commands

This topic introduces the main utilities and commands that are included with XL Fortran. It does not contain all compiler utilities and commands.

Utilities

install The **install** utility installs and configures IBM XL Fortran for Linux, V15.1.5 for use on your system.

xlf_configure

You can use the **xlf_configure** utility to facilitate the use of XL Fortran with IBM Advance Toolchain. For details, see "Using IBM XL Fortran for Linux, V15.1.5 with the Advance Toolchain" in *XL Fortran Compiler Reference*.

Commands

genhtml command

The **genhtml** command converts an existing XML diagnostic report produced by the **-qlistfmt** option. You can choose to produce XML or HTML diagnostic reports by using the **-qlistfmt** option. The report can help you find optimization opportunities. For more information about how to use this command, see **genhtml** command in the *XL Fortran Compiler Reference*.

Profile-directed feedback (PDF) related commands

cleanpdf command

The **cleanpdf** command removes all the PDF files or the specified PDF files from the directory to which profile-directed feedback data is written.

mergepdf command

The **mergepdf** command provides the ability to weigh the importance of two or more PDF records when combining them into a single record. The PDF records must be derived from the same executable.

showpdf command

The **showpdf** command displays the following types of profiling information for all the procedures executed in a PDF run (compilation under the **-qpdf1** option):

- Block-counter profiling
- Call-counter profiling
- Value profiling
- Cache-miss profiling, if you specified the **-qpdf1=level=2** option during the PDF1 step.

You can view the first two types of profiling information in either text or XML format. However, you can view value profiling and cache-miss profiling information only in XML format.

For more information, see **cleanpdf**, **mergepdf**, and **showpdf** in the *XL Fortran Compiler Reference*.

Advance Toolchain 9.0 support

IBM XL Fortran for Linux, V15.1.5 fully supports IBM Advance Toolchain 9.0, which is a set of open source development tools and runtime libraries. With IBM Advance Toolchain, you can take advantage of the latest POWER® hardware features on Linux, especially the tuned libraries.

For more information, see "Using IBM XL Fortran for Linux, V15.1.5 with Advance Toolchain" in the *XL Fortran Compiler Reference*.

Program optimization

XL Fortran provides several compiler options that can help you control the optimization and performance of your programs.

With these options, you can perform the following tasks:




- Select different levels of compiler optimizations.
- Control optimizations for loops, floating point, and other types of operations.

- Optimize a program for a particular class of machines or for a very specific machine configuration, depending on where the program will run.

Optimizing transformations can give your application better overall execution performance. XL Fortran provides a portfolio of optimizing transformations tailored to various supported hardware. These transformations offer the following benefits:

- Reducing the number of instructions executed for critical operations
- Restructuring generated object code to make optimal use of the Power Architecture[®] processors
- Improving the usage of the memory subsystem
- Exploiting the ability of the architecture to handle large amounts of shared memory parallelization

Related information

-  [Optimizing your applications](#)
-  [Optimization and tuning](#)
-  [Intrinsic procedures](#)

Shared memory parallelization

XL Fortran supports application development for multiprocessor system architectures.

You can use any of the following methods to develop your parallelized applications with XL Fortran:

- Directive-based shared memory parallelization (OpenMP, SMP)
- Instructing the compiler to automatically generate shared memory parallelization
- Message-passing-based shared or distributed memory parallelization (MPI)

The parallel programming facilities are based on the concept of threads. Parallel programming exploits the advantages of multiprocessor systems while maintaining a full binary compatibility with existing uniprocessor systems. This means that a multithreaded program that works on a uniprocessor system can take advantage of a multiprocessor system without recompiling.

For more information, see "Parallel programming with XL Fortran" in the *XL Fortran Optimization and Programming Guide*.

OpenMP directives

OpenMP directives are a set of API-based commands supported by XL Fortran and many other IBM and non-IBM C, C++, and Fortran compilers.

You can use OpenMP directives to instruct the compiler how to parallelize a particular block of code. The existence of the directives in the source removes the need for the compiler to perform any dependence analysis on the parallel code. OpenMP directives require the presence of Pthread libraries to provide the necessary infrastructure for parallelization.

OpenMP directives address the following important issues of parallelizing an application:

1. Clauses and directives are available for scoping variables. Generally, variables should not be shared; that is, each thread should have its own copy of the variable.
2. Work sharing directives specify how the work contained in a parallel region of code should be distributed across the threads.
3. Directives are available to control synchronization between threads.

Starting from IBM XL Fortran for Linux, V15.1.5, XL Fortran supports OpenMP API Version 4.0 and selected features of the OpenMP API Version 4.5 specification. For details, see “OpenMP support” on page 19.

Related information



Optimizing your applications



The OpenMP API specification for parallel programming

Diagnostic reports

The compiler listings, XML reports, and HTML reports provide important information to help you develop and debug your applications more efficiently.

Listing information is organized into optional sections that you can include or omit. For more information about the applicable compiler options and the listing itself, see "Understanding XL Fortran compiler listings" in the *XL Fortran Compiler Reference*.

You can also obtain diagnostic information from the compiler in XML or HTML format. The XML and HTML reports provide information about optimizations that the compiler performed or could not perform. You can use this information to reduce programming effort when tuning applications, especially high-performance applications. The report is defined by an XML schema and is easily consumable by tools that you can create to read and analyze the results. For detailed information about this report and how to use it, see "Using reports to diagnose optimization opportunities" in the *XL Fortran Optimization and Programming Guide*.

Symbolic debugger support

You can instruct XL Fortran to include debugging information in your compiled objects by using different levels of the **-g** or **-qdbg** compiler option.

The debugging information can be examined by **gdb** or any symbolic debugger that supports the DWARF debug format on Linux to help you debug your programs.

Related information



-g



-qdbg

Chapter 2. What's new for IBM XL Fortran for Linux, V15.1.5

This section describes features and enhancements added to IBM XL Fortran for Linux, V15.1.5.

Intrinsic procedures

This section describes intrinsic procedures that are new for IBM XL Fortran for Linux, V15.1.5.

New intrinsic procedures

Note: The following new intrinsic procedures are valid only when `-qarch` is set to target POWER9 processors.

VEC_ABSD(ARG1, ARG2)

Returns a vector that contains the absolute difference of the corresponding elements of the given vectors.

VEC_CMPNE(ARG1, ARG2)

Returns a vector containing the results of comparing each set of the corresponding elements of the given vectors for inequality.

VEC_CMPNEZ(ARG1, ARG2)

Returns a vector that contains the results of comparing each set of the corresponding elements of the given vectors for inequality, or the results of testing the corresponding element of given vectors for the value of zero.

VEC_CNTLZ(ARG1)

Counts the most significant zero bits of each element of the given vector.

VEC_CNTLZ_LSBB(ARG1)

Counts the leading byte elements of the given vector that have a least significant bit of 0.

VEC_CNTTZ(ARG1)

Counts the least significant zero bits of each element of the given vector.

VEC_CNTTZ_LSBB(ARG1)

Counts the trailing byte elements of the given vector that have a least significant bit of 0.

VEC_EXTRACT_EXP(ARG1)

Returns a vector that contains the exponent of the given vector.

VEC_EXTRACT_SIG(ARG1)

Returns a vector that contains the significand of the given vector.

VEC_LOAD_SPLATS(ARG1, ARG2)

Loads an 8-byte element from the memory address specified by the given displacement and the given pointer, and then splats it onto a vector.

VEC_RLMI(ARG1, ARG2, ARG3)

Returns a vector that contains each element of the given vector rotated left and inserted under a mask into another vector.

VEC_RLNM(ARG1, ARG2, ARG3)

Returns a vector that contains each element of the given vector rotated left and intersected with a mask.

VEC_SLV(ARG1, ARG2)

Left-shifts the elements of a given vector by a given number of bits.

VEC_SRV(ARG1, ARG2)

Right-shifts the elements of a given vector by a given number of bits.

VEC_XL_LEN(ARG1, ARG2)

Returns a vector that loads a given number of bytes from the given address.

Compiler options and directives

This topic describes new or changed compiler options and directives.

You can specify compiler options on the command line. You can also modify compiler behavior through directives embedded in your application source files. For detailed descriptions and usage information for XL Fortran compiler options and directives, see the *XL Fortran Compiler Reference*.

Compiler options

-qarch=pwr9

The **-qarch=pwr9** suboption is added to produce object code that contains instructions that run on the POWER9 hardware platforms.

> GPU -qoffload

The **-qoffload** option is added to enable support for offloading OpenMP target regions on an NVIDIA GPU. **GPU <**

-qport The **-qport=c_loc** and **-qport=noc_loc** suboptions are added. When **-qport=c_loc** is in effect, the C_LOC function from the ISO_C_BINDING module can accept arguments that have neither the **POINTER** nor the **TARGET** attribute.

-qtune=pwr9

The **-qtune=pwr9** suboption is added to tune the optimizations for the POWER9 hardware platforms.

Directives

IBM XL Fortran for Linux, V15.1.5 adds support for more OpenMP directives. For these newly supported OpenMP directives, see the summary in “OpenMP support.”

OpenMP support

IBM XL Fortran for Linux, V15.1.5 partially supports the OpenMP Application Program Interface Version 4.5 specification. The XL Fortran implementation is based on IBM's interpretation of the OpenMP Application Program Interface 4.5.

New directives

In addition to the existing OpenMP directives, IBM XL Fortran for Linux, V15.1.5 adds support for the following directives and their clauses:

Table 4. OpenMP device constructs

Directive	Clause
TARGET DATA	<ul style="list-style-type: none"> • IF • DEVICE • MAP
TARGET ENTER DATA	<ul style="list-style-type: none"> • IF • DEVICE • MAP
TARGET EXIT DATA	<ul style="list-style-type: none"> • IF • DEVICE • MAP
TARGET	<ul style="list-style-type: none"> • DEFAULTMAP • DEVICE • FIRSTPRIVATE • IF • MAP • PRIVATE
TARGET UPDATE	<ul style="list-style-type: none"> • DEVICE • FROM • IF • TO
DECLARE TARGET	<ul style="list-style-type: none"> • TO
TEAMS	<ul style="list-style-type: none"> • DEFAULT • FIRSTPRIVATE • NUM_TEAMS • PRIVATE • REDUCTION • SHARED • THREAD_LIMIT
DISTRIBUTE	<ul style="list-style-type: none"> • COLLAPSE • DIST_SCHEDULE • FIRSTPRIVATE • LASTPRIVATE • PRIVATE
DISTRIBUTE PARALLEL DO	Any clauses that are accepted by the DISTRIBUTE or PARALLEL DO directive except the LINEAR and ORDERED clauses.

The following directives as combined constructs are also supported. For more information, see Combined constructs.

- **TARGET PARALLEL**
- **TARGET PARALLEL DO**
- **TARGET TEAMS**
- **TARGET TEAMS DISTRIBUTE**
- **TARGET TEAMS DISTRIBUTE PARALLEL DO**

- **TEAMS DISTRIBUTE**
- **TEAMS DISTRIBUTE PARALLEL DO**

For detailed information about these directives, see topic Detailed descriptions of parallelization directives in the *XL Fortran Optimization and Programming Guide*.

New routines

In addition to the existing OpenMP routines, IBM XL Fortran for Linux, V15.1.5 supports the following OpenMP execution environment routines:

- `omp_get_default_device()`
- `omp_get_initial_device()`
- `omp_get_num_devices()`
- `omp_get_num_teams()`
- `omp_get_team_num()`
- `omp_is_initial_device()`
- `omp_set_default_device()`

For detailed information about these OpenMP routines, see topic Routines for OpenMP in the *XL Fortran Optimization and Programming Guide*.

New environment variables

IBM XL Fortran for Linux, V15.1.5 adds support for the following OpenMP environment variables:

- `OMP_DEFAULT_DEVICE = n`
- `XLSMPOPTS = TARGET = {MANDATORY | OPTIONAL | DISABLE}`

For more information, see topic Setting runtime options in the *XL Fortran Optimization and Programming Guide*.

Performance and optimization

This topic describes features and enhancements that assist performance tuning and application optimization.

Offloading computations to the NVIDIA GPUs

> GPU

You can offload compute-intensive parts of an application and associated data to the NVIDIA GPUs by using the device constructs that are supported by IBM XL Fortran for Linux, V15.1.5. You must specify the `-qoffload` and `-qsmp` options to enable support for offloading OpenMP target regions to the NVIDIA GPUs.

For more information, see topic Offloading computations to the NVIDIA GPUs in the *XL Fortran Optimization and Programming Guide*.

GPU <

Other enhancements

This section describes other features and enhancements in IBM XL Fortran for Linux, V15.1.5.

Allocatable scalars in some parallelization directives clauses

You can specify allocatable scalars in the FIRSTPRIVATE, COPYIN, COPYPRIVATE, and LASTPRIVATE clauses.

Chapter 3. Migration of your applications


This section lists important considerations when you migrate your applications that were compiled with other versions of XL Fortran.


Migrating from Linux for big endian distributions to Linux for little endian distributions


IBM XL Fortran for Linux, V15.1.1 or later is compatible with other versions of the compiler running on the POWER8® big endian systems. There are, however, some differences to consider.

- To help migrate programs from big endian systems, you can use the **-qaltivec=be** or **-qaltivec=le** option to toggle the vector element sequence in registers to big endian or little endian element order.
- To make big endian data files compatible in little endian systems, you can use the **-qufmt=be** option so that the I/O operations on unformatted data files use the big endian byte order.

Related information

 Program migration from big-endian systems

 -qufmt

 -qaltivec

Resolving the compatibility issues of IPA object files

It is recommended that you use the latest version of the compiler to compile and link the IPA object files to avoid compatibility issues. If any compatibility issues occur, you can try these resolutions.

IPA object files that are compiled using earlier versions but are linked by a newer version

When IPA object files that are compiled with earlier versions of compilers are linked by a newer version, errors might occur if the IPA object is compiled by one of the following compilers.

- XL Fortran, V15.1.2 or earlier
- XL C/C++, V13.1.2 or earlier

Try resolving the compatibility issue using one of the following methods:

- Recompile and link your object files with the latest XL compiler if you want to use IPA.
- Do not enable the **-qipa** option.

IPA object files that are compiled using newer versions but are linked by an earlier version

If IPA object files that are compiled with newer versions of compilers are linked by an earlier version, errors occur during the link step. You might be able to resolve the issue by recompiling and linking the IPA object files with the latest XL compiler.

For more information, see [Interprocedural analysis \(IPA\)](#)Interprocedural analysis (IPA).

Chapter 4. Enhancements added in earlier releases

This section describes enhancements added in earlier releases. These enhancements also apply to the current release.

Enhancements added in Version 15.1.4

This section describes features and enhancements added in IBM XL Fortran for Linux, V15.1.4. These features and enhancements apply to later releases as well.

Compiler options and directives

This topic describes new or changed compiler options.

You can specify compiler options on the command line. You can also modify compiler behavior through directives embedded in your application source files. For detailed descriptions and usage information for XL Fortran compiler options and directives, see the *XL Fortran Compiler Reference*.

Compiler options

-qstackprotect

The **-qstackprotect** suboptions are added to provide protection against malicious input data or programming errors that overwrite or corrupt the stack.

-O3, -Ofast

The **-O3** option is updated to imply **-qhot**. The **-Ofast** option, which is equivalent to the **-O3** option, is added.

> CUDA Fortran -qcuda

The **-qcuda** option is added to enable the compiler support for CUDA Fortran. **< CUDA Fortran**

> CUDA Fortran -qcudaerr

The **-qcudaerr** option is added to control whether the compiler automatically inserts error checking code for CUDA API calls. **< CUDA Fortran**

> CUDA Fortran -qpath=@, -qpath=n, -qpath=s, -qpath=w, -qpath=x

These **-qpath** suboptions are added to specify substitute path names for the following XL Fortran components:

- The PTX assembler
- The NVIDIA C compiler
- The XL intermediate language (W-Code) splitter
- The XL intermediate language (W-Code) to NVVM-IR translator
- The NVVM-IR to PTX translator **< CUDA Fortran**

-qpdf1, -qpdf2

When **-qpdf1** or **-qpdf2** is specified without a suboption, **exename** is the default suboption. When **-qpdf1=exename** is in effect, the generated PDF file is named **.<output_name>_pdf** by default, where **<output_name>** is the name of the executable file.

-qslmtags

The **-qslmtags** option is added to control whether IBM Software License Metric (SLM) Tags logging tracks compiler license usage.

-qsmp=omp

The behaviour of the **-qsmp=omp** option is changed to imply **-qdirective=SMP\\$\:\\$OMP:IBMP:IBMT:IBM***, which turns on the **SMP\$, \$OMP, IBMP, IBMT, and IBM*** trigger constants. In previous releases, the compiler only recognizes the **\$OMP** trigger constant when the **-qsmp=omp** option is in effect.

-W, -X The **-W** option is updated to pass one or more options to specific compiler components [CUDA Fortran](#), including the components that are used during compilation of CUDA Fortran programs [CUDA Fortran](#). The **-X** option is added to pass one option to a specific compiler component [CUDA Fortran](#), which can be any component that is used during compilation of CUDA Fortran programs [CUDA Fortran](#).

Directives

SIMD_PEEL

The **SIMD_PEEL** directive is added to instruct the compiler to peel a SIMDizable loop.

CUDA Fortran support

IBM XL Fortran for Linux, V15.1.4 supports the CUDA Fortran programming model, which is a subset of CUDA constructs, to exploit the NVIDIA GPUs.

CUDA is a parallel programming model and software environment that is developed by NVIDIA. It provides programmers with a set of instructions that enables GPU acceleration for data-parallel computations. The computing performance of many applications can be increased by using CUDA directly or by linking to GPU-accelerated libraries.

You can use the commonly used subset of CUDA Fortran that is provided by IBM XL Fortran for Linux to offload computations to the NVIDIA GPUs. In IBM XL Fortran for Linux, V15.1.4, some CUDA Fortran features are not supported. For the detailed information, see Reference and limitations for the supported CUDA Fortran in the *Getting Started with CUDA Fortran programming using XL Fortran*.

New invocation command, options, and file name extensions

To compile CUDA Fortran programs, you can use a newly introduced invocation command that enables CUDA Fortran support, **xlcf**, or specify a newly introduced compiler option, **-qcuda**, to enable CUDA Fortran support. You can find more information about new and changed options that are useful for CUDA Fortran programs in “Compiler options and directives” on page 15.

Files with the new file name extension **.cuf** or **.CUF** are also recognized as Fortran source files. When compiled by using **xlfr**, **.cuf** or **.CUF** files are recognized as CUDA Fortran source files, and CUDA Fortran support is enabled automatically.

System requirements

CUDA Fortran support that is provided in IBM XL Fortran for Linux, V15.1.4 requires the following hardware, operating system, and software:

- **Hardware**

You can use any IBM Power Systems server that has an NVIDIA GPU installed and is supported by your operating system distribution, for example, IBM POWER System S822LC for high performance computing and IBM POWER System S824L. For a complete list of the IBM Power Systems servers, see <http://www.ibm.com/systems/power/hardware/>.

- **Operating system**

You can use the following little endian operating systems supported by the IBM Power Systems servers:

- Ubuntu 14.04.x starting with Ubuntu 14.04.2
- Red Hat Enterprise Linux 7.2 (RHEL 7.2)

- **Software**

- CUDA Toolkit 7.5, which you can download from <https://developer.nvidia.com/cuda-downloads>

For more information, see the *Getting Started with CUDA Fortran programming using XL Fortran*.

Performance and optimization

This topic describes features and enhancements that assist performance tuning and application optimization.

Dumping snapshot PDF profiling information to files during execution

When using profile-directed feedback (PDF) optimization in previous releases, you could get PDF profiling information that is written to one or more PDF files only upon normal termination, and no PDF files could be generated upon abnormal termination. Now you can dump PDF profiling information to one or more PDF snapshot files during execution. This is especially useful if you want to save the generated PDF profiling information when the application is to be terminated abnormally. After defining the new environment variable `PDF_SIGNAL_TO_DUMP`, you can use it to trigger dumping PDF profiling information.

For more information, see "Dumping snapshot PDF profiling information to files during execution" in the *XL Fortran Optimization and Programming Guide*.

Other enhancements

This section describes other features and enhancements in IBM XL Fortran for Linux, V15.1.4.

Increased maximum length of fixed and free form statements

The maximum length of fixed and free form source statements is increased from 34,000 bytes to 65,000 bytes.

Enhancements added in Version 15.1.3

This section describes features and enhancements added in IBM XL Fortran for Linux, V15.1.3. These features and enhancements apply to later releases as well.

Language features

This topic lists new language features that are introduced in this release of XL Fortran.

> F2008

BIND attribute for an internal procedure

You can specify the **BIND** attribute without the **NAME=** specifier on an internal procedure. In previous releases, the **BIND** attribute could not be specified on an internal procedure.

DO CONCURRENT construct

With the **DO CONCURRENT** construct, you can specify that individual loop iterations have no interdependencies, in which case the execution order of the iterations can be indeterminate at the beginning of the execution of the **DO CONCURRENT** construct.

Polymorphic variable in an intrinsic assignment

In an intrinsic assignment *variable = expression*, *variable* now can be polymorphic. If *variable* is polymorphic, the following rules apply:

- *variable* must be allocatable.
- *variable* must be type-compatible with *expression*, or the declared types of *variable* and *expression* must conform.

Multiple allocate objects allowed on an ALLOCATE statement






You can allocate more than one *allocate_object* by using an **ALLOCATE** statement that contains the **SOURCE=** or **MOLD=** specifier. In previous releases, you could only allocate one *allocate_object* by using an **ALLOCATE** statement that contains the **SOURCE=** or **MOLD=** specifier.

VALUE attribute

You can specify the **VALUE** attribute on an array dummy argument that has either assumed shape or explicit shape. In the previous releases, you could not specify the **VALUE** attribute on array dummy arguments.

F2008 <

Related information in the *XL Fortran Language Reference*

-  Internal procedures
-  DO CONCURRENT construct (Fortran 2008)
-  Intrinsic assignment
-  ALLOCATE
-  VALUE (Fortran 2003)

OpenMP support

IBM XL Fortran for Linux, V15.1.3 fully supports the OpenMP Application Program Interface Version 3.1 specification and partially supports the OpenMP Application Program Interface Version 4.0 and 4.5 specifications. The XL Fortran implementation is based on IBM's interpretation of the OpenMP Application Program Interface Version 3.1, 4.0, and 4.5.

In addition to existing OpenMP routines, this version of XL Fortran now supports the following OpenMP 4.5 routines:

- `omp_get_num_places()`
- `omp_get_partition_num_places()`
- `omp_get_partition_place_nums(place_nums)`
- `omp_get_place_num_procs(place_num)`
- `omp_get_place_proc_ids(place_num, ids)`
- `omp_get_place_num()`

This version of XL Fortran now supports the following OpenMP 4.0 features:

- The `omp_get_proc_bind()` routine
- The **OMP_PLACES** environment variable

The following environment variables are extended to control the thread affinity policy:

- **OMP_DYNAMIC**
- **OMP_DISPLAY_ENV**
- **OMP_PROC_BIND**
- **OMP_THREAD_LIMIT**

The following directives are changed:

ATOMIC

The **ATOMIC** directive is extended to support sequentially atomic operations by specifying a new optional clause **seq_cst**. This clause forces atomically performed operations to include an implicit flush operation without a list.

PARALLEL, PARALLEL DO, PARALLEL SECTIONS, PARALLEL WORKSHARE

These directives are extended to support a new clause **proc_bind**. This clause specifies a policy for assigning threads to places within the current place partition.

Intrinsic procedures

This section describes intrinsic procedures that are new or changed for IBM XL Fortran for Linux, V15.1.3.

New intrinsic procedures

VEC_CIPHER_BE(ARG1,ARG2)

Performs one round of the AES cipher operation on an intermediate state ARG1 by using a given round key ARG2.

VEC_CIPHERLAST_BE(ARG1,ARG2)

Performs the final round of the AES cipher operation on an intermediate state ARG1 by using a given round key ARG2.

VEC_NCIPHER_BE(ARG1,ARG2)

Performs one round of the AES inverse cipher operation on an intermediate state ARG1 by using a given round key ARG2.

VEC_NCIPHERLAST_BE(ARG1,ARG2)

Performs the final round of the AES inverse cipher operation on an intermediate state ARG1 by using a given round key ARG2.

VEC_PMSUM_BE(ARG1,ARG2)

Performs an exclusive-OR operation by implementing a polynomial addition on each even-odd pair of the polynomial multiplication result of the corresponding elements.

VEC_SBOX_BE(ARG1)

Performs the SubBytes operation, as defined in *Federal Information Processing Standards FIPS-197*, on a given state ARG1.

VEC_SHASIGMA_BE(ARG1,ARG2,ARG3)

Performs a secure hash computation in accordance with *Federal Information Processing Standards FIPS-180-3*.

Changed intrinsic procedures**VEC_MULE(ARG1, ARG2)**

VEC_MULE(ARG1, ARG2) now supports INTEGER(4) and UNSIGNED(4) vector types of parameters.

VEC_MULO(ARG1, ARG2)

VEC_MULO(ARG1, ARG2) now supports INTEGER(4) and UNSIGNED(4) vector types of parameters.

Other enhancements

This section describes other features and enhancements in IBM XL Fortran for Linux, V15.1.3.

Software License Metric (SLM) Tags logging support

IBM XL Fortran for Linux, V15.1.3 fully supports IBM Software License Metric (SLM) Tags logging so that you can use IBM License Metric Tool (ILMT) to track compiler license usage.

For more information, see "Tracking compiler license usage" in the *XL Fortran Compiler Reference*.

Enhancements added in Version 15.1.2

This section describes features and enhancements added in IBM XL Fortran for Linux, V15.1.2. These features and enhancements apply to later releases as well.

Fortran 2008 features

This topic lists the Fortran 2008 features that are introduced in this release of XL Fortran.

New restriction on elemental procedures

If a dummy argument of an elemental procedure does not have the **VALUE** attribute, the dummy argument must have the **INTENT** attribute specified.

New restriction on the reference to an elemental function

In a reference to an elemental procedure, if any actual argument is an array, every actual argument that corresponds to an `INTENT(OUT)` or `INTENT(INOUT)` dummy argument must be an array.

Language interoperability features

XL Fortran implements selected language interoperability features, which accept programs that contain parts written in Fortran and parts written in the C language.

This version of XL Fortran provides support for the following language interoperability features as specified in TS 29113:

Assumed-length arguments of type character

`BIND(C)` procedures that have nonallocatable and nonpointer dummy arguments of type character with assumed length can interoperate with C functions having formal parameters that are pointers to type `CFI_cdesc_t`.

The `C_PTRDIFF_T` named constant in the `ISO_C_BINDING` module

The `C_PTRDIFF_T` named constant is added to the `ISO_C_BINDING` module. Fortran entities of type `INTEGER(C_PTRDIFF_T)` are interoperable with C entities of type `ptrdiff_t`.

OpenMP support

IBM XL Fortran for Linux, V15.1.2 fully supports the OpenMP Application Program Interface Version 3.1 specification and partially supports the OpenMP Application Program Interface Version 4.0 specification. The XL Fortran implementation is based on IBM's interpretation of the OpenMP Application Program Interface Version 3.1 and 4.0.

This version of XL Fortran supports the following OpenMP 4.0 features:

- Atomic update, atomic capture, and atomic swap
- `OMP_DISPLAY_ENV` environment variable

Intrinsic procedures

This section describes intrinsic procedures that are new for IBM XL Fortran for Linux, V15.1.2.


`VEC_MERGEE(ARG1,ARG2)`

Merges the values of even-numbered elements of two vectors.

`VEC_MERGEO(ARG1,ARG2)`

Merges the values of odd-numbered elements of two vectors.

Related information

 Intrinsic procedures

Commands

This section describes new, changed, or removed compiler commands.

`resetpdf`

This command has been removed. It is recommended that you use the

cleanpdf command instead. The behavior of the **resetpdf** command is the same as that of the **cleanpdf** command. For more information, see **-qpdf1**, **-qpdf2** in the *XL Fortran Compiler Reference*.

Compiler options

This topic describes new or changed compiler options.

-qfloat

The following suboptions are added:

subnormals

This suboption asserts to the compiler that the code uses subnormal floating point values, also known as denormalized floating point values.

nosubnormals

This suboption asserts to the compiler that the code does not use subnormal floating point values, also known as denormalized floating point values.

Whether or not you specify this suboption, the behavior of your program will not change, but the compiler uses this information to gain possible performance improvements. To use **-qfloat=subnormals** or **-qfloat=nosubnormals**, you must also specify the **-qarch=pwr8** and **-qtune=pwr8** options.

-qvisibility

Specifies the visibility attribute for external linkage symbols in object files.

Chapter 5. Setting up and customizing XL Fortran

This section describes how to set up and customize the compiler according to your own requirements.

For complete prerequisite and installation information for XL Fortran, see "Before installing XL Fortran" in the *XL Fortran Installation Guide*.

Using custom compiler configuration files

You can customize compiler settings and options by modifying the default configuration file or creating your own configuration file.

You have the following options to customize compiler settings:

- The XL Fortran compiler installation process creates a default compiler configuration file. You can directly modify this configuration file to add default options for specific needs. However, if you later apply updates to the compiler, you must reapply all of your modifications to the newly installed configuration file.
- You can create your own custom configuration file that either overrides or complements the default configuration file. The compiler can recognize and resolve compiler settings that you specify in your custom configuration files with compiler settings that are specified in the default configuration file. Compiler updates that might later affect settings in the default configuration file do not affect the settings in your custom configuration files.

Related information

 Using custom compiler configuration files

Chapter 6. Developing applications with XL Fortran

Fortran application development consists of repeating cycles of editing, compiling, linking, and running. By default, compiling and linking are combined into a single step.



Notes:

- Before you use the compiler, ensure that XL Fortran is properly installed and configured. For more information, see the *XL Fortran Installation Guide*.
- To learn about writing Fortran programs, refer to the *XL Fortran Language Reference*.

Compiler phases

A typical compiler invocation executes some or all of these activities in sequence. For link time optimizations, some activities are executed more than once during a compilation. As each compilation component runs, the results are sent to the next step in the sequence.

1. Preprocessing of source files
2. Compilation, which might consist of the following phases, depending on what compiler options are specified:
 - a. Front-end parsing and semantic analysis
 - b. Loop transformations
 - c. High-level optimization
 - d. Low-level optimization
 - e. Register allocation
 - f. Final assembly
3. Assembling the assembly (.s) files and the unpreprocessed assembler (.S) files after they are preprocessed
4. Object linking to create an executable application

 The compilation and linking processes of CUDA Fortran programs go through more phases than the typical process. For more information, see the *Getting Started with CUDA Fortran programming using XL Fortran*. 

To see the compiler step through these phases, specify the `-v` compiler option when you compile your application. To see the amount of time the compiler spends in each phase, specify `-qphsinfo`.

Editing Fortran source files

To create Fortran source programs, you can use any text editor available on your system.

Source programs must be saved using a recognized file name suffix. See “XL Fortran input and output files” on page 29 for a list of suffixes recognized by XL Fortran.

For a Fortran source program to be a valid program, it must conform to the language definitions specified in the *XL Fortran Language Reference*.

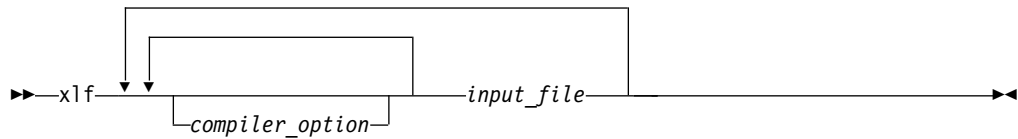
Compiling with XL Fortran

XL Fortran is a command-line compiler. Invocation commands and options can be selected according to the needs of a particular Fortran application.

Invoking the compiler

The compiler invocation commands perform all necessary steps to compile Fortran source files, assemble any `.s` and `.S` files, and link the object files and libraries into an executable program.

To compile a Fortran source program, use the following basic invocation syntax:



For most applications, compile with `xlf` or a threadsafe counterpart.

- If the file name extensions of your source files indicate a specific level of Fortran, such as `.f08`, `.f03`, `.f95`, `.f90`, or `.f77`, you can compile with `xlf` or the corresponding generic threadsafe invocations so that the compiler can automatically select the appropriate language-level defaults.
- **CUDA Fortran** If the file name extensions are `.cuf` or `.CUF`, you can compile source files with `xlf_r` so that CUDA Fortran support is enabled automatically.
CUDA Fortran
- If you compile source files whose file name extensions are generic, such as `.f` or `.F`, with `xlf` or corresponding generic threadsafe invocations, the compilation conforms to FORTRAN 77.

For more information about threadsafe counterparts, see "Compiling XL Fortran programs" in the *XL Fortran Compiler Reference*.

Invocation commands for different levels of Fortran

More invocation commands are available to meet specialized compilation needs, primarily to provide explicit compilation support for different levels and extensions of the Fortran language. These invocation commands do not consider the specific level of Fortran indicated by the source file name extensions, such as **CUDA Fortran** `.cuf` **CUDA Fortran**, `.f08`, `.f03`, `.f95`, `.f90`, or `.f77`.

CUDA Fortran The `xlcuf` invocation command enables CUDA Fortran support for any Fortran source files. **CUDA Fortran**

The invocation commands that relate to Fortran language levels are described as follows:

Table 5. Invocation commands and corresponding Fortran language standards

Language level	Invocation commands	Notes
Fortran 2008	<ul style="list-style-type: none"> • f2008 • xlf2008 	<p>These compiler invocations accept Fortran 90 free source form by default. To use fixed source form with these invocations, you must specify the -qfixed option.</p> <p>I/O formats are slightly different between these commands and the other commands. I/O formats for the Fortran 95 compiler invocations are also different from the I/O formats of Fortran 90 invocations. Switch to the Fortran 95 formats for data files whenever possible.</p>
Fortran 2003	<ul style="list-style-type: none"> • f2003 • xlf2003 	
Fortran 95	<ul style="list-style-type: none"> • f95 • xlf95 	
Fortran 90	<ul style="list-style-type: none"> • f90 • xlf90 	
FORTRAN 77	<ul style="list-style-type: none"> • f77 • fort77 	<p>Where possible, these compiler invocations maintain compatibility with existing programs by using the same I/O formats as FORTRAN 77 and some implementation behaviors that are compatible with earlier versions of XL Fortran.</p> <p>You might need to continue using these invocations for compatibility with existing makefiles and build environments. However, programs that are compiled with these invocations might not conform to the Fortran 2008, Fortran 2003, Fortran 95, or Fortran 90 language level standards.</p>

Compiling with full compliance to language standards

By default, these invocation commands do not conform completely to the corresponding language standards. If you need full compliance, compile with the following compiler option settings and specify the following runtime options before you run the program, with a command similar to the following examples:

Fortran 2008

Compiler options:

```
-qlanglvl=2008std -qnodirective -qnoescape -qextname
-qfloat=nomaf:nofold -qnoswapomp -qstrictieeeemod
```

Example of runtime options:

```
export XLF RTEOPTS="err_recovery=no:langlvl=2008std:
iostat_end=2003std:internal_nldelim=2003std"
```

Fortran 2003

Compiler options:

```
-qlanglvl=2003std -qnodirective -qnoescape -qextname
-qfloat=nomaf:nofold -qnoswapomp -qstrictieeeemod
```

Example of runtime options:

```
export XLF RTEOPTS="err_recovery=no:langlvl=2003std:
iostat_end=2003std:internal_nldelim=2003std"
```

Fortran 95

Compiler options:

```
-qlanglvl=95std -qnodirective -qnoescape -qextname  
-qfloat=nomaf:nofold -qnoswapomp
```

Example of runtime options:

```
export XLF RTEOPTS="err_recovery=no:langlvl=95std"
```

Fortran 90**Compiler options:**

```
-qlanglvl=90std -qnodirective -qnoescape -qextname  
-qfloat=nomaf:nofold -qnoswapomp
```

Example of runtime options:

```
export XLF RTEOPTS="err_recovery=no:langlvl=90std"
```

The default settings are intended to provide the best combination of performance and usability, so change them only when full compliance is required. Some of the options that are mentioned in the preceding tables are only required for compliance in specific situations. For example, you must specify **-qextname** only when an external symbol, such as a common block or subprogram, is named **main**.

The -qxlf2003 compiler option

The **-qxlf2003** compiler option provides compatibility with XL Fortran V10.1 and the Fortran 2003 standard for certain aspects of the language.

When you compile with the Fortran 2003 or Fortran 2008 compiler invocations, the default setting is **-qxlf2003=polymorphic**. This setting instructs the compiler to allow polymorphic items such as the CLASS type specifier and SELECT TYPE construct in your Fortran application source.

For all other compiler invocations, the default is **-qxlf2003=nopolymorphic**.

The -qxlf2008 compiler option

You can use the **-qxlf2008** compiler option for the following purposes:

- To enable language features specific to the Fortran 2008 standard when you compile with compiler invocations that conform to earlier Fortran standards
- To disable language features specific to the Fortran 2008 standard when you compile with compiler invocations that conform to the Fortran 2008 standard

When you compile with the Fortran 2008 compiler invocations, the default setting is **-qxlf2008=checkpresence**. This setting instructs the compiler to check dummy argument presence according to the Fortran 2008 standard.

For all other compiler invocations, the default is **-qxlf2008=nocheckpresence**.

See "Compiling XL Fortran programs" in the *XL Fortran Compiler Reference* for more information about compiler invocation commands available to you.

Specifying compiler options

Compiler options perform a variety of functions, such as setting compiler characteristics, describing the object code to be produced, controlling the diagnostic messages emitted, and performing some preprocessor functions.

You can specify compiler options in one or any combination of the following ways:

- On the command line
- In your source code using directive statements
- In a makefile
- In the stanzas found in a compiler configuration file

You can also pass options to the linker, assembler, and preprocessor.

Priority sequence of compiler options

Option conflicts and incompatibilities might occur when multiple compiler options are specified. To resolve these conflicts in a consistent manner, the compiler applies the following general priority sequence to most options:

1. Directive statements in your source file override command line settings.
2. Compiler option settings on the command line override configuration file settings.
3. Configuration file settings override default settings.

Generally, if the same compiler option is specified more than once on the command line when the compiler is invoked, the last option specified prevails.

Note: Some compiler options, such as the **-I** option, do not follow the priority sequence described above. The compiler searches any directories specified with **-I** in the `xl.cfg` file before it searches the directories specified with **-I** on the command line. The **-I** option is cumulative rather than preemptive. Other options with cumulative behavior are **-R** and **-l** (lowercase L).

Related information



Specifying options on the command line

XL Fortran input and output files

The topic describes the file types that are recognized by XL Fortran.

For detailed information about these and additional file types used by the compiler, see "Types of input files" in the *XL Fortran Compiler Reference* and "Types of output files" in the *XL Fortran Compiler Reference*.

Table 6. Input file types





Filename extension	Description
.f, .F, .f77, .F77, .f90, .F90, .f95, .F95, .f03, .F03, .f08, .F08,  .cuf, .CUF 	Fortran source files
.mod	Module symbol files
.smod 	Submodule symbol files
.o	Object files
.s	Assembler files
.so	Shared object or library files
Note:  Fortran 2008	

Table 7. Output file types

Filename extension	Description
a.out	Default name for executable file created by the compiler
.mod	Module symbol files
.smod 1	Submodule symbol files
.lst	Listing files
.o	Object files
.s	Assembler files
.so	Shared object or library files
Note: 1 Fortran 2008	

Linking your compiled applications with XL Fortran

By default, you do not need to do anything special to link an XL Fortran program. The compiler invocation commands automatically call the linker to produce an executable output file.

For example, you can use `xlf` to compile `file1.f` and `file3.f` to produce object files `file1.o` and `file3.o`; after that, all object files, including `file2.o`, are submitted to the linker to produce one executable.

```
xlf file1.f file2.o file3.f
```

Compiling and linking in separate steps

To produce object files that can be linked later, use the `-c` option.

```
xlf -c file1.f           # Produce one object file (file1.o)
xlf -c file2.f file3.f  # Or multiple object files (file2.o, file3.o)
xlf file1.o file2.o file3.o # Link object files with default libraries
```

Dynamic and static linking

You can use XL Fortran to take advantage of the operating system facilities for both dynamic and static linking.

Dynamic linking means that the code for some external routines is located and loaded when the program is first run. When you compile a program that uses shared libraries, the shared libraries are dynamically linked to your program by default. Dynamically linked programs take up less disk space and less virtual memory if more than one program uses the routines in the shared libraries. During linking, they do not require any special precautions to avoid naming conflicts with library routines. They might perform better than statically linked programs if several programs use the same shared routines at the same time. By using dynamic linking, you can upgrade the routines in the shared libraries without relinking. This form of linking is the default and no additional options are needed.

Static linking means that the code for all routines called by your program becomes part of the executable file. Statically linked programs can be moved to run on systems without the XL Fortran runtime libraries. They might perform better than dynamically linked programs if they make many calls to library routines or call many small routines. They do require some precautions in choosing names for data objects and routines in the program if you want to avoid naming conflicts with library routines.

Note: Dynamically and statically linked programs might not work if you compile them on one level of the operating system and run them on a different level of the operating system.

Running your compiled application

After a program is compiled and linked, you can run the generated executable file on the command line.

The default file name for the program executable file produced by the XL Fortran compiler is **a.out**. You can select a different name with the **-o** compiler option.

You should avoid giving your program executable file the same name as system or shell commands, such as `test` or `cp`, as you could accidentally execute the wrong command. If you do decide to name your program executable file with the same name as a system or shell command, you should execute your program by specifying the path name to the directory in which your executable file resides, such as `./test`.

To run a program, enter the name of the program executable file with runtime arguments on the command line.

Canceling execution

To suspend a running program, press **Ctrl+Z** while the program is in the foreground. Use the `fg` command to resume running.

To cancel a running program, press **Ctrl+C** while the program is in the foreground.

Setting runtime options

You can use environment variable settings to control certain runtime options and behaviors of applications created with the XL Fortran compiler. Some environment variables do not control actual runtime behavior, but they can have an impact on how your applications run.

For more information about environment variables and how they can affect your applications at run time, see the *XL Fortran Installation Guide*.

Running compiled applications on other systems

If you want to run an application developed with the XL Fortran compiler on another system that does not have the compiler installed, you need to install a runtime environment on that system or link your application statically.

You can obtain the latest XL Fortran Runtime Environment images, together with licensing and usage information, from the XL Fortran for Linux support page.

XL Fortran compiler diagnostic aids

XL Fortran issues diagnostic messages when it encounters problems compiling your application. You can use these messages and other information provided in compiler output listings to help identify and correct such problems.

For more information about listing, diagnostics, and related compiler options that can help you resolve problems with your application, see the following topics in the *XL Fortran Compiler Reference*:

- "Understanding XL Fortran compiler listings"
- "Error checking and debugging options"
- "Listings, messages, and compiler information options"

Debugging compiled applications

You can use a symbolic debugger to debug applications compiled with XL Fortran.

At compile time, you can use the **-g** or **-qlinedebug** option to instruct the XL Fortran compiler to include debugging information in compiled output. For **-g**, you can also use different levels to balance between debug capability and compiler optimization. For more information about the debugging options, see "Error checking and debugging" in the *XL Fortran Compiler Reference*.

You can then use **gdb** or any symbolic debugger that supports the DWARF debug format on Linux to step through and inspect the behavior of your compiled application.

Optimized applications pose special challenges when you debug your applications. For more information about debugging your optimized code, see "Debugging optimized code" in the *XL Fortran Optimization and Programming Guide*.

Determining which level of XL Fortran is being used

To display the version and release level of XL Fortran that you are using, invoke the compiler with the **-qversion** compiler option.

For example, to obtain detailed version information, enter the following command:
`xlf -qversion=verbose`

Appendix. Accessibility features for IBM XL Fortran for Linux

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully.

Accessibility features

IBM XL Fortran for Linux uses the latest W3C Standard, WAI-ARIA 1.0 (<http://www.w3.org/TR/wai-aria/>), to ensure compliance to US Section 508 (<http://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards>) and Web Content Accessibility Guidelines (WCAG) 2.0 (<http://www.w3.org/TR/WCAG20/>). To take advantage of accessibility features, use the latest release of your screen reader in combination with the latest web browser that is supported by this product.

The IBM XL Fortran for Linux online product documentation in IBM Knowledge Center is enabled for accessibility. The accessibility features of IBM Knowledge Center are described at http://www.ibm.com/support/knowledgecenter/doc/kc_help.html#accessibility.

Keyboard navigation

This product uses standard navigation keys.

Interface information

You can use speech recognition software like a Text-to-speech (TTS) tool to view the output generated by the compiler.

The IBM XL Fortran for Linux online product documentation is available in IBM Knowledge Center, which is viewable from a standard web browser.

PDF files have limited accessibility support. With PDF documentation, you can use optional font enlargement, high-contrast display settings, and can navigate by keyboard alone.

To enable your screen reader to accurately read syntax diagrams, source code examples, and text that contains the period or comma PICTURE symbols, you must set the screen reader to speak all punctuation.

Related accessibility information

To learn the accessibility features of the operation systems that are supported by IBM XL Fortran for Linux, see the following information:

- Ubuntu (<https://help.ubuntu.com/community/Accessibility>)
- SUSE Linux Enterprise Server (<https://www.suse.com>)
- Red Hat Enterprise Linux (<http://www.redhat.com>)
- Community Enterprise Operating System (<https://www.centos.org/>)

In addition to standard IBM help desk and support websites, IBM has established a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service
800-IBM-3383 (800-426-3383)
(within North America)

IBM and accessibility

For more information about the commitment that IBM has to accessibility, see IBM Accessibility (www.ibm.com/able).

Notices

Programming interfaces: Intended programming interfaces allow the customer to write programs to obtain the services of IBM XL Fortran for Linux.

This information was developed for products and services offered in the U.S.A. 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 give 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
U.S.A.

For license inquiries regarding double-byte (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.

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.

Licenses of this program who want 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:

Intellectual Property Dept. for Rational Software
IBM Corporation
5 Technology Park Drive
Westford, MA 01886
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1998, 2016.

This software and documentation are based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California. We acknowledge the following institution for its role in this product's development: the Electrical Engineering and Computer Sciences Department at the Berkeley campus.

PRIVACY POLICY CONSIDERATIONS:

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, or 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.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

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 <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.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 <http://www.ibm.com/software/info/product-privacy>.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

NVIDIA and CUDA are either registered trademarks or trademarks of NVIDIA Corporation in the United States, other countries, or both.

Index

A

accessibility
 interfaces 33
 keyboard navigation 33

B

built-in functions
 new features
 V15.1.2 21

C

code
 dynamic linking 30
 static linking 30
compilation
 activities 25
 introduction 26
compiler options
 conflicts 29
 new or changed features
 V15.1.2 22
 V15.1.4 15
 V15.1.5 8
 specifying 28
CUDA Fortran
 invocation commands 16
 system requirements 16

D

debugging
 applications 32
 options 32
directives
 new or changed features
 V15.1.4 15
 V15.1.5 8

E

execution
 applications 31

F

file types
 input files 29
 output files 29

I

intrinsic procedures
 new features
 V15.1.3 19
 V15.1.5 7

invocation
 applications 31
invocation commands
 syntax 26

L

languages
 extensions 3
 standards 3
linking
 applications 30

M

more enhancements
 new features
 V15.1.3 20
 V15.1.4 17
 V15.1.5 11

O

object files
 creating 30
 linking 30
optimization
 new features
 V15.1.4 17
 V15.1.5 10
 programs 4

P

performance
 optimizing transformations 4
problem resolution
 compiler options 32
 related links 32
programs
 running 31

S

shared memory parallelization
 methods 5
 OpenMP directives 5
source files
 editing 25
 suffixes 25

U

utilities
 introduction 3



Product Number: 5765-J10; 5725-C75

Printed in USA

SC27-6620-04

