IBM XL Fortran compilers features

December 2017

Chapter 1. Overview

The IBM® XL Fortran compilers are full-featured Fortran language compilers available for AIX® and Linux on Power® Systems. IBM XL Fortran supports the latest international programming language standards and industry specifications. More information about IBM XL Fortran is available at:

- Fortran compilers family page (www.ibm.com/software/products/en/fortcompfami)
- XL compilers on Power community (https://ibm.biz/xl-power-compilers)

Current IBM XL Fortran releases

Up to December 2017, the following IBM XL Fortran compiler products are available:

- On Linux:
  - IBM XL Fortran for Linux, V15.1.6
  - IBM XL Fortran for Linux, V15.1.5
  - IBM XL Fortran for Linux, V15.1.4
  - IBM XL Fortran for Linux, V15.1.3
  - IBM XL Fortran for Linux, V15.1.2
  - IBM XL Fortran for Linux, V15.1.1
  - IBM XL Fortran for Linux, V15.1
  - IBM XL Fortran for Linux, V14.1
  - IBM XL Fortran for Linux, V13.1
- On AIX:
  - IBM XL Fortran for AIX, V15.1.3
  - IBM XL Fortran for AIX, V15.1.2
  - IBM XL Fortran for AIX, V15.1
  - IBM XL Fortran for AIX, V14.1
  - IBM XL Fortran for AIX, V13.1

Chapter 2. IBM XL Fortran history

IBM XL Fortran was first released in 1989. From 1990, XL Fortran has been available on IBM Power architecture based hardware since the hardware's inception. Always available on the AIX operating system, XL Fortran was released for Linux operating systems in 2003. IBM announced the Blue Gene® program in December 1999, leading to the release of XL Fortran versions for Blue Gene®/L in 2006, Blue Gene®/P in 2007, and Blue Gene®/Q in 2012.

In 2014, IBM released an XL Fortran compiler that supports application development targeting Linux for little endian distributions running on IBM Power Systems™ with POWER8® processor and architecture. Starting from XL Fortran for Linux, V15.1.1, the little endian compiler is packaged together with the XL Fortran compiler that supports Power Systems running Linux distributions configured for the big endian architecture. A key strength of the IBM XL compilers on Linux is its ability to generate highly optimized code for execution on IBM Power Systems. You can create and port applications for execution on the next generation of IBM systems built on POWER8 technology, designed to handle big data and to drive modern workloads for cloud, mobile, and social.

Starting from V15.1.4, XL Fortran for Linux little endian distributions supports the CUDA Fortran programming model, which is a subset of CUDA constructs, to exploit the NVIDIA GPUs. You can use the commonly used subset of CUDA Fortran that is provided by the compiler to offload computations to the NVIDIA GPUs. Starting from V15.1.5, XL Fortran for Linux little endian distributions supports a number of OpenMP 4.5 device constructs to offload compute-intensive parts of an application and associated data to the NVIDIA GPUs.

A no-charge, fully functional Community Edition is also offered for developers who do not require official IBM support starting from V15.1.5.

The XL Fortran compiler contains industry-leading optimization technology that has continually evolved since the mid-1980s. The optimization technology in XL Fortran maximizes the performance of code executing on PowerPC® and POWER-series processors. The close relationship between IBM's compiler and chip design groups ensures that XL Fortran can take maximum advantage of IBM processor technology as it becomes available. Additionally, the compiler group has a unique opportunity to influence design decisions in the production of chips, so that XL Fortran can exploit its fullest potential. Starting from V15.1, IBM XL Fortran for AIX and IBM XL Fortran for Linux fully support POWER8 processors. Starting from V15.1.6, XL Fortran for Linux little endian distributions provides full exploitation of POWER® 9 technology.

IBM XL Fortran has been and continues to be the Fortran compiler of choice on AIX. XL Fortran is used to measure system performance and announce benchmark results, such as the SPEC benchmark. IBM XL Fortran shares common compilation technology with the IBM XL C and XL C++ compilers, specifically key components for optimization and targeting specific architectures. The IBM XL Fortran compiler is used to build parts of other key IBM products such as AIX. XL Fortran is not only a compiler for large, broad-spectrum applications. Over the years, customers have come to rely on IBM XL Fortran's stability, versatility, and performance to build critical applications both large and small.

Chapter 3. Standards support

Starting from V14.1, XL Fortran for AIX and XL Fortran for Linux fully implement the FORTRAN 77, Fortran 90, Fortran 95, and Fortran 2003 language standards and partially implement the Fortran 2008 language standard.

Further, starting from V15.1.3, XL Fortran for Linux for little endian distributions supports a partial implementation of the OpenMP Fortran API Version 4.0 and Version 4.5 as well as a complete implementation of the OpenMP Fortran API Version 3.1, which gives users the power and versatility to develop portable SMP applications and to use task level parallelization and loop parallelization.

Different invocation commands, such as **xlf**, **xlf90**, **xlf95**, **xlf2003**, and **xlf2008**, allow you to automatically supply default options for a particular Fortran standard. The compiler also implements a language-level compiler option, **-qlanglvl**, which can report non-conformant source constructs. Additionally, the IBM XL Fortran runtime library supports environment variables that control behavior for a particular standard, or report non-conformant usage while an application is executing.

IBM is a member of the ISO/IEC JTC1/SC22/WG5 Committee, INCITS PL22.3 (formerly J3) Fortran Technical Committee, and the OpenMP Language Architecture Review Board. These bodies are responsible for the Fortran language and the OpenMP API extension. IBM is thus in a position to understand and participate in the latest updates, clarifications, and recommendations to the Fortran standard and industry specifications.

Available TS 29113 features

Starting from V15.1, XL Fortran for AIX and Linux support the following selected language interoperability features from TS 29113:

- Interoperability of assumed-length arguments
- Interoperability of assumed-rank arguments
- Interoperability of assumed-type arguments
- Interoperability of allocatable and pointer arguments
- Interoperability of optional arguments
- Interoperable variables in asynchronous communication
- The ISO_Fortran_binding.h header file

For the latest status about TS 29113 support in XL Fortran, see https://ibm.biz/FortranTS29113Status.

Available Fortran 2008 standard features

Starting from V15.1.3, IBM XL Fortran for AIX and IBM XL Fortran for Linux little endian distributions support the following Fortran 2008 standard features:

Module enhancements

- Submodules provide additional structuring facilities for modules.

- To declare a module procedure interface body or define a separate module procedure, specify the MODULE prefix specifier for the procedure or SUBROUTINE statement.

Performance enhancements

- The DO CONCURRENT construct provides a means for programs to specify that individual loop iterations have no interdependencies.
- The CONTIGUOUS attribute provides a means for programs to specify restrictions on the storage layout of pointer targets and assumed-shape dummy arguments.

Data declaration

- A named constant array's shape can be implied by its value.
- A FORALL index variable can have its type and kind explicitly declared within the construct.
- In addition to derived types, the TYPE type specifier is extended to declare entities of intrinsic types.
- Multiple type-bound procedures can be declared in a single type-bound procedure statement.

Data usage and computation

- You can allocate more than one allocate_object by using an ALLOCATE statement that contains the SOURCE= or MOLD= specifier.
- The MOLD= specifier has been added to the ALLOCATE statement. In addition, you can omit the bounds in the ALLOCATE statement if you provide source_expr in the SOURCE= or MOLD= specifier.
- The real and imaginary parts of a complex entity can be accessed independently by using complex part designators.
- Variables now can be polymorphic in an intrinsic assignment.

Input and output

- NEWUNIT= in an OPEN statement automatically selects a unit number that does not interfere with other unit numbers that are selected by the program.

Execution control

- The BLOCK construct can contain declarations of objects with construct scope.
- The EXIT statement can transfer control from within more named executable constructs.
- The STOP statement can now take an integer or character constant expression as stop code. The ERROR STOP statement initiates error termination of a program while the STOP statement initiates normal termination of a program.

Intrinsic procedures

- The intrinsic procedures ACOS, ASIN, ATAN, COSH, SINH, TAN, and TANH can have arguments of type complex.
- The intrinsic procedures ACOSH, ASINH, and ATANH calculate the inverse hyperbolic cosine, sine, and tangent respectively.
- The intrinsic procedures DSHIFTL and DSHIFTR calculate combined left and right shifts.
- The intrinsic procedures ERF, ERFC, and ERFC_SCALED calculate the error procedure and its complement.
- The EXECUTE_COMMAND_LINE subroutine can be used to pass a command to the operating system for execution.
- The intrinsic procedure FINDLOC searches an array for a value.

- The intrinsic procedures GAMMA and LOG_GAMMA calculate the gamma procedure and its log.
- The intrinsic procedure HYPOT calculates the Euclidean distance.
- The intrinsic procedure IS_CONTIGUOUS(ARRAY) tests contiguity of an array.
- The intrinsic procedures LEADZ and TRAILZ return the number of leading and trailing zero bits in an integer.
- The intrinsic procedures MASKL and MASKR return simple left and right justified masks.
- A BACK= argument is added to the intrinsic procedures MAXLOC and MINLOC.
- The intrinsic procedures POPCNT and POPPAR return the number of 1 bits of an integer and its parity.
- The intrinsic procedures SHIFTA, SHIFTL, and SHIFTR perform shift operations.
- A RADIX= argument has been added to the SELECTED_REAL_KIND intrinsic procedure.

Intrinsic modules

- The procedures COMPILER_VERSION and COMPILER_OPTIONS in the ISO_FORTRAN_ENV intrinsic module return information about the program translation phase.
- The ISO_FORTRAN_ENV intrinsic module provides the following constants that are related to the Fortran environment: CHARACTER_KINDS, INT8, INT16, INT32, INT64, INTEGER_KINDS, IOSTAT_INQUIRE_INTERNAL_UNIT, LOGICAL_KINDS, REAL32, REAL64, REAL128, and REAL_KINDS.
- A RADIX= argument has been added to the IEEE_SELECTED_REAL_KIND intrinsic procedure.

Programs and procedures

- A dummy argument that has the POINTER and INTENT(IN) attributes can be argument that is associated with a nonpointer actual argument that has the TARGET attribute.
- Internal procedures and pointers to internal procedures can be used as actual arguments. Besides, internal procedures can be used as procedure pointer targets.
- If a dummy argument of an elemental procedure does not have the VALUE attribute, the dummy argument must have the INTENT attribute specified.
- 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.
- A separate module subprograms defines a separate module procedure that is declared by a corresponding module procedure interface body.
- Elemental procedures are no longer required to be pure in Fortran 2008. You can explicitly declare procedures with the IMPURE prefix specifier.
- The generic resolution rules are extended to distinguish between allocatable and pointer dummy arguments and between procedure and data dummy arguments.
- A double colon separator (::) can be used in PROCEDURE and MODULE PROCEDURE statements inside interface blocks.
- The procedure and SUBROUTINE keywords can be omitted from the END statement for a module or internal subprogram.
- You can specify the BIND attribute without the NAME= specifier for an internal procedure.

- You can specify the VALUE attribute on an array dummy argument that has either assumed shape or explicit shape.

For the latest status about Fortran 2008 support in XL Fortran, see https://ibm.biz/Fortran2008Status.

Available Fortran 2003 standard features

IBM XL Fortran started to deliver support for the Fortran 2003 standard as early as the V8.1 release. Starting from the V13.1 release, the XL Fortran compilers fully support the Fortran 2003 standard.

OpenMP API support

XL Fortran supports the OpenMP specification, as understood and interpreted by IBM as well as the POSIX 1003.1-1996 standard. IBM implementation of OpenMP in XL Fortran is an extension to the standard Fortran language.

OpenMP API Version 4.5 specification

Starting from V15.1.3, IBM XL Fortran for Linux for little endian distributions supports the following OpenMP V4.5 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()`

Starting from V15.1.5, IBM XL Fortran for Linux for little endian distributions adds support for the following OpenMP 4.5 features. Some features are useful for offloading computations to the NVIDIA GPUs.

- New directives
  - `TARGET DATA`
  - `TARGET ENTER DATA`
  - `TARGET ENTER DATA`
  - `TARGET`
  - `TARGET UPDATE`
  - `DECLARE TARGET`
  - `TEAMS`
  - `DISTRIBUTE`
  - `DISTRIBUTE PARALLEL DO`
  - Combined constructs
- New 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()`
- New environment variables
  - **OMP_DEFAULT_DEVICE = *n***
  - **XLSMPOPTS = TARGET = {MANDATORY | OPTIONAL | DISABLE}**

IBM XL Fortran for Linux, V15.1.6 for little endian distributions adds support for the following directives:

- `SIMD`
- `DO SIMD`
- `DISTRIBUTE SIMD`
- `DISTRIBUTE PARALLEL DO SIMD`
- Combined constructs

OpenMP API Version 4.0 specification

Starting from IBM XL Fortran for AIX, V15.1, and IBM XL Fortran for Linux, V15.1, the XL Fortran compiler supports the following OpenMP V4.0 features:

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

Starting from V15.1.3, IBM XL Fortran for Linux for little endian distributions provides additional OpenMP V4.0 support as follows:

- 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**

OpenMP API Version 3.1 specification

Starting from V14.1, IBM XL Fortran for AIX and IBM XL Fortran for Linux fully support OpenMP V3.1.

Language extensions

In addition to standards conformance, IBM XL Fortran features many common language extensions, as well as code-porting features. IBM XL Fortran implements IBM Fortran language extensions such as SAA Fortran, and a number of extensions added in VS Fortran. These extensions simplify porting existing Fortran code from other platforms to IBM XL Fortran compilation targets. Featured industry language extensions include:

- 128-bit floating-point data type
- Cray (integer) pointers
- Structure records
- Union maps
- **BYTE**, **STATIC**, **AUTOMATIC**
- **SIZEOF** intrinsic

IBM XL Fortran provides *xlfutility*, a Fortran 90 module that you can use in your applications to access operating system features such as system timers and process query functions without having to write the access code yourself. The Pthreads Library Module provides you with Fortran language interfaces to the AIX and Linux operating systems pthread libraries. You can use the f_pthread module to parallelize and thread-safe your code.

A rich set of Power-based functionality further enhances the compiler. IBM XL Fortran includes a large number of intrinsic procedures and directives to give you source-level access to hardware-level operations such as cache control, data prefetching, and hardware data-manipulation instructions. The *xlf_fp_util* module adds additional intrinsic functions that allow you to both control and query the hardware floating-point status and control register. The **VECTOR** intrinsic data type and dozens of intrinsic functions give you direct access to the powerful VMX, VSX vector instructions available in Power chips like the PowerPC 970, POWER6®, POWER7®, and POWER8.

IBM XL Fortran also includes the following directives that assist in the optimization of your applications in a number of different ways:

- Directives that transform source constructs, such as **SUBSCRIPTORDER** and **COLLAPSE**, that you can apply to more efficiently use memory at execution time.
- Directives to guide the compiler in optimization transformations such as loop unrolling.
- Directives that inform the compiler that certain constructs, such as **DO** loops, have particular behaviors or attributes. This allows the optimizer to identify additional opportunities for optimization that may not be possible otherwise.

Chapter 4. Offloading computations to the NVIDIA GPUs

The combination of the IBM POWER processors and the NVIDIA GPUs provides a platform for heterogeneous high-performance computing that can run several technical computing workloads efficiently. The computational capability is built on top of massively parallel and multithreaded cores within the NVIDIA GPUs and the IBM POWER processors. You can offload parallel operations within applications, such as data analysis or high-performance computing workloads, to GPUs.

Programming with supported OpenMP 4.5 device constructs

Starting from XL Fortran for Linux, V15.1.5 for little endian distributions, you can offload compute-intensive parts of an application and associated data to the NVIDIA GPUs by using the supported device constructs. For more information, see "OpenMP API support" on page 8.

You must specify the **-qoffload** option to enable the support for offloading OpenMP target regions to NVIDIA GPUs. For **-qoffload** to take effect, you must also specify the **-qsmp** option to enable support for OpenMP target regions. You can also use the -qtgtarch option to embed PTX code for the targeted virtual GPU architectures and compiled code images for the targeted real GPU architectures into the object files or executables.

You can control the computations offloaded to target devices by using the **XLSMPOPTS=target={mandatory | default | disabled}** and **XLSMPOPTS=cudamemcheckfriendly={off | on}** environment variables.

Programming with supported CUDA Fortran features

Starting from V15.1.4, IBM XL Fortran for Linux supports the CUDA Fortran programming model to exploit the NVIDIA GPUs. 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.

You must specify the **-qcuda** option to enable the compiler support for CUDA Fortran.

For more information about CUDA Fortran programming using IBM XL Fortran for Linux, including useful compiler options and a list of supported CUDA Fortran features, see the *Getting Started with CUDA Fortran programming using XL Fortran*.

System prerequisites

To compile and link programs that contain code to be offloaded to the NVIDIA GPUs with IBM XL Fortran for Linux, you must ensure the following operating system, hardware, and software requirements are met.

- Use any IBM Power Systems™ server that has one or more NVIDIA GPUs installed and is supported by your Linux operating system distribution and the NVIDIA CUDA Toolkit.
- Use either of the following supported little endian operating systems:
  - Ubuntu 16.04.3

  - Red Hat Enterprise Linux 7.3 (RHEL 7.3) or above
- Install CUDA Toolkit 9.0 or 9.1.

For more information, see the *XL Fortran Installation Guide*.

CUDA Fortran support

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

Starting from V15.1.4, IBM XL Fortran for Linux supports a commonly used subset of CUDA Fortran. You can use the **xlcuf** invocation command or specify the **-qcuda** option to enable CUDA Fortran support. For more information on the language extensions introduced by CUDA Fortran, see the *CUDA Fortran Programming Guide and Reference* manual downloadable from http://www.pgroup.com/doc/pgicudaforug.pdf.

Some CUDA Fortran features are not supported in IBM XL Fortran for Linux. For detailed information, see *Getting Started with CUDA Fortran programming using XL Fortran*.

Chapter 5. What's new in V15.1.6

IBM XL Fortran for Linux, V15.1.6 for little endian distributions includes all fixes from all previous release PTFs, resolutions to other known issues, and the following updates.

OpenMP support

IBM XL Fortran for Linux, V15.1.6 for little endian distributions partially supports the OpenMP Application Program Interface Version 4.5 specification. In addition to the existing OpenMP directives, IBM XL Fortran for Linux, V15.1.6 adds support for the following directives and their clauses.

- `SIMD`
- `DO SIMD`
- `DISTRIBUTE SIMD`
- `DISTRIBUTE PARALLEL DO SIMD`
- Combined constructs

For more information, see the *XL Fortran Optimization and Programming Guide*.

CUDA Fortran support

These CUDA Fortran enhancements apply to IBM XL Fortran for Linux, V15.1.6.

- Support for shuffle intrinsics
- Support for the cudaMemAdvise, cudaMemPrefetchAsync, cudaDeviceGetStreamPriorityRange, cudaStreamCreateWithPriority, and cudaStreamGetPriority runtime API functions
- Pass LLVM IR bitcode libraries specified on the command line to llvm2ptx
- Improved performance for kernels that have assumed-shape, allocatable, or pointer dummy arguments
- The **-qtgtarch** option to specify the real or virtual GPU architectures where the code can run. It overrides the default GPU architecture and allows the compiler to take maximum advantage of the capabilities and machine instructions which are specific to a GPU architecture, or common to a virtual architecture. The default GPU architecture is detected at compiler configuration time and encoded into the compiler configuration file.
- Configuration of GPU stack and data limits by using the **XLCUF_GPU_STACK_LIMIT** and **XLCUF_GPU_DATA_LIMIT** environment variables
- Display of GPU stack and data limits by using the **XLCUF_DISPLAY_LIMITS** environment variable
- Display of GPU stack and data limits by using the **XLCUF_DISPLAY_LIMITS** environment variable

For more information about CUDA Fortran support provided by IBM XL Fortran for Linux, see the *Getting Started with CUDA Fortran programming using XL Fortran*.

Offloading computations to the NVIDIA GPUs

IBM XL Fortran for Linux, V15.1.6 provides the following enhancements that you can benefit from when offloading computations to the NVIDIA GPUs.

Support for CUDA Toolkit 9.0 and 9.1
IBM XL Fortran for Linux, V15.1.6 requires CUDA Toolkit 9.0 and 9.1. The sm_70 and compute_70 GPU architectures are supported as defined by the CUDA Toolkit.

Specification of GPU architectures for the generated code
You can use the **-qtgtarch** option to specify the real or virtual GPU architectures where the code can run, overriding the default GPU architecture.

Support for the cuda-memcheck tool
You can use the **XLSMPOPTS=cudamemcheckfriendly={off | on}** environment variable to control whether to disable the check for pinned memory in the runtime and allow the program to be executed under the cuda-memcheck tool from the NVIDIA CUDA Toolkit.

Pass LLVM IR bitcode libraries to llvm2ptx
You can specify LLVM IR bitcode libraries, which have a suffix of .bc, on the command line, to pass the LLVM IR bitcode libraries to llvm2ptx, the NVVM-IR to PTX translator.

GPU runtime inlining support for inlining calls made to the OpenMP GPU runtime libraries
This enhancement reduces overhead and significantly improves performance of OpenMP target regions that are offloaded to the accelerator.

Improved GPU code generation
This enhancement applies to several OpenMP directives when contained in an OpenMP target region, most notably parallel loops and reductions.

Pay attention to the following change when you migrate to IBM XL Fortran for Linux, V15.1.6 from earlier releases.

Changed environment variables
XLSMPOPTS=target=optional is renamed to **XLSMPOPTS=target=default** with the identical functionality.

XLSMPOPTS=target=disable is renamed to **XLSMPOPTS=target=disabled** with the identical functionality.

OpenMP interoperability with CUDA C/C++ and CUDA Fortran

- You can call kernels written in CUDA C/C++ or CUDA Fortran in your OpenMP programs from the host.
- You can use the OpenMP **USE_DEVICE_PTR** clause to pass OpenMP mapped variables to CUDA kernels that are launched from the host.
- You can use the OpenMP **IS_DEVICE_PTR** clause to access CUDA device attribute variables or to pass device addresses directly to target regions.

Other new or changed compiler options

New compiler options

-gsplit-dwarf
The **-gsplit-dwarf** option is added to enable the DWARF debugging information to be generated in one or more separate .dwo files.

-qtgtarch

The **-qtgtarch** option is added to embed the PTX code for virtual GPU architectures and the compiled code images for real GPU architectures into the object files or executables.

-qpreprocess

The **-qpreprocess** option is added to invoke cpp to preprocess files that have valid Fortran file suffixes, such as .f and .f90.

Changed compiler options

-D The **-D** option is updated to define a macro as in a **#define** preprocessor directive.

-U The **-U** option is updated to undefine a preprocessor macro defined by the compiler or by the **-D** compiler option.

Operating system support

IBM XL Fortran for Linux

Starting from V15.1.1, IBM XL Fortran for Linux includes two Fortran compilers: IBM XL Fortran for Linux for big endian distributions and IBM XL Fortran for Linux for little endian distributions. XL Fortran packages two Linux compilers in a single product.

Operating system support

IBM XL Fortran for Linux, V15.1.6 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 3 (SLES 12 SP3)
- Red Hat Enterprise Linux 7.3 (RHEL 7.3)
- Red Hat Enterprise Linux 7.4 (RHEL 7.4)
- Red Hat Enterprise Linux 7.4 for Power Little Endian (POWER9)
- Community Enterprise Operating System 7 (CentOS 7)

Note: To compile CUDA Fortran programs or programs that contain code to be offloaded to the NVIDIA GPUs, you must use either of the following operating systems:

- Ubuntu Server 16.04.3
- Red Hat Enterprise Linux 7.3 (RHEL 7.3) or above

IBM XL Fortran for Linux, V15.1 for big endian distributions supports the following operating systems:

- Red Hat Enterprise Linux 6.4 (RHEL 6.4)
- Red Hat Enterprise Linux 6.5 (RHEL 6.5)
- Red Hat Enterprise Linux 6.6 (RHEL 6.6)
- Red Hat Enterprise Linux 7.0 (RHEL 7.0)
- SUSE Linux Enterprise Server 11 Service Pack 2 (SLES 11 SP2)

- SUSE Linux Enterprise Server 11 Service Pack 3 (SLES 11 SP3)

Migrating from big endian Linux to little endian Linux

IBM XL Fortran for Linux for little endian distributions is compatible with versions of the compiler running on the POWER8 big endian systems. To help migrate programs from big-endian systems, you can use the **-qaltivec** option to toggle the vector element sequence in registers to big-endian or little-endian element order.

For more information, see "Program migration from big-endian systems" in the *XL Fortran for Linux Optimization and Programming Guide*.

Chapter 6. Support for POWER8 architecture

Starting from XL Fortran for AIX, V15.1 and XL Fortran for Linux, V15.1, support for the POWER8 processor is introduced.

The features and enhancements introduced in support of the POWER8 processors fall under the following categories:

- MASS libraries for POWER8 processors
- Compiler options for POWER8 processors
- Built-in functions for POWER8 processors

Mathematical Acceleration Subsystem (MASS) libraries for POWER8 processors

- **Vector libraries**. The vector MASS library **libmassvp8.a** contains vector procedures that have been tuned for the POWER8 architecture. The library supports both single precision and double precision vector arguments.
- **SIMD libraries**. The MASS SIMD library **libmass_simdp8.a** contains an accelerated set of frequently used math intrinsic procedures that provide improved performance over the corresponding standard system library procedures.

Compiler options for POWER8 architecture

- The **-qarch** compiler option specifies the processor architecture for which code is generated. The **-qtune** compiler option tunes instruction selection, scheduling, and other architecture-dependent performance enhancements to run best on a specific hardware architecture.
- The **-qarch=pwr8** suboption produces object code containing instructions that utilize the POWER8 technology. With the **-qtune=pwr8** suboption, optimizations are tuned for the POWER8 architecture.

POWER8 hardware directives and intrinsics

The hardware directives and intrinsics are added to support the following POWER8 processor features:

- POWER8 intrinsics for vector processing
- POWER8 cryptography intrinsics
- POWER8 transactional memory intrinsics
- POWER8 prefetch directives
- POWER8 prefetch intrinsic procedures

Chapter 7. Support for POWER9 architecture

Starting from XL Fortran for Linux, V15.1.5 for little endian distributions, the compiler introduces the support for the POWER9 architecture, which falls into the following categories:

- MASS libraries that are tuned for the POWER9 architecture

  The newly added vector library **libmassvp9.a** and the SIMD library **libmass_simdp9.a** contain functions that are tuned for the POWER9 architecture.
- Compiler options for the POWER9 architecture

  The **-qarch=pwr9** suboption enables the compiler to generate code that exploits new POWER9 instructions, which improve the program performance on the POWER9 architecture. With the **-qtune=pwr9** suboption, optimizations are tuned for the POWER9 architecture.
- Built-in functions for the POWER9 architecture

  The new built-in functions fall into the following categories:
  - Fixed-point built-in functions
  - Binary floating-point built-in functions
  - Binary-coded decimal built-in functions
  - Vector built-in functions

Chapter 8. Porting code to IBM XL Fortran

IBM XL Fortran has several options to assist you in porting Fortran code from other systems. The **-qport** option includes suboptions that allow you to toggle compiler behaviors that can assist in porting non-standard code. There are many other compiler options that can assist with code porting or reproducing results from other platforms, including such functionality as:

- Altering default data type sizes and data type interaction rules
- Controlling certain program flow behaviors
- Influencing what types of comment-form and conditional compilation directives are accepted
- Altering the naming scheme for global names seen by the linker
- Specifying floating-point manipulation rules such as rounding mode
- Altering the behavior of the IBM XL Fortran runtime through environment variables

Chapter 9. Utilization tracking and reporting

On Linux for big endian distributions and AIX, you can use the utilization reporting tool to help determine whether the use of the compiler by your organization matches your compiler license entitlements. When enabled, each invocation of the compiler is recorded in a compiler utilization file. The utilization reporting tool can then be used to generate a report of the overall usage of the compiler within your organization. In particular, the report indicates whether the compiler usage complies with the number of Concurrent User licenses that you have acquired.

On Linux for little endian distributions and AIX, you can also enable IBM Software License Metric (SLM) Tags logging in the compiler so that IBM License Metric Tool (ILMT) can track compiler license usage.

Chapter 10. Programming with IBM XL Fortran

IBM XL Fortran supports a traditional command-line development toolset common on many platforms and works well with standard build tools such as *make*.

The binary code IBM XL Fortran produces is compatible with IBM XL C/C++ for AIX and IBM XL C/C++ for Linux. This gives you the flexibility of coding your application with interlanguage calls where appropriate. IBM XL Fortran provides several compilation options to assist application development in a mixed-language environment. The **BIND(C)** Fortran 2003 standard feature supported by IBM XL Fortran allows you to program interlanguage calls in a standards-conforming portable manner.

Debugging IBM XL Fortran applications

You can debug applications built with IBM XL Fortran using debuggers such as the standard AIX dbx debugger and Linux gdb debugger. Various third parties have marketed software products for debugging IBM XL Fortran applications, including the TotalView debugger available from Rogue Wave Software and the DDT debugger available from Allinea.

Note: References to these third party products are provided for your convenience only. Check with the vendor for more details.

The IBM XL Fortran runtime library supplies traceback capabilities through procedures you can call from your application. The runtime library contains default signal and exception-handling support, but is flexible enough to allow you to override these with your own handlers. The compiler also supplies options to assist in debugging applications.

Program analysis

IBM XL Fortran produces binary code that can be used with system standard performance analysis tools such as *gprof* or, on AIX, *tprof*. Using the **-p** compiler option allows your application to emit the information that some profilers require.

The compiler can assist you in analyzing your program in several ways. A listing facility can show you your source code with embedded diagnostic messages in addition to seeing them on the screen. Other options allow you to select the severity of the messages reported, or filter out specific messages completely.

IBM XL Fortran runtime library

The IBM XL Fortran runtime library is an integral part of the product. The runtime is available in both threadsafe and non-threadsafe variations, allowing you to choose the library best suited to your application's performance requirements. Many environment variables are recognized by the runtime and allow you to direct the runtime behavior. These include variables for language conformance and porting issues, input-output characteristics, error reporting, and multithreading behaviors. The XL Fortran runtime library can be redistributed with your XL Fortran applications to your customers. Alternatively, the XL Fortran runtime library can be downloaded by users of your applications.

The IBM XL Fortran runtime library is available for download at:

www.ibm.com/support/docview.wss?rs=43&uid=swg21156900

Chapter 11. Optimization capabilities

One of the key strengths of IBM XL Fortran is optimization. The compiler offers the benefits of optimization technology that has been evolving at IBM since the late 1980s, combining extensive hardware knowledge with a comprehensive understanding of compiler technology and what users look for in a compiler when building end-user applications. The optimization can decrease execution time and make your applications run faster, producing code that is highly tuned for execution on PowerPC and Power platforms. Improving optimization is a key goal of the IBM compiler team, and one that will continue to be a major focus with each iteration of the IBM XL Fortran compiler.

The optimizer includes five base optimization levels; **-O0**, **-O2**, **-O3**, **-O4**, and **-O5**. These levels allow you to choose from minimal optimization to intense program analysis that provides benefits even across programming languages. Optimization analyses range from local basic block to subprogram to file-level to whole-program analysis. The higher the optimization level, the more intense the program analysis becomes as increasingly sophisticated optimization techniques are applied to your code.

At any optimization level, the compiler performs transformations that result in performance improvements, while still executing your code the way it was written. At higher levels, the compiler can trade numeric precision for execution speed. If this effect is not desired, you can specify compiler options such as **-qstrict** to prevent such trade-offs. Other options such as **-qsmallstack** or **-qcompact** allow you to bias optimization decisions in favor of smaller stack space or program size.

The IBM XL Fortran compiler does not limit your optimization choices unnecessarily. All of the optimization capabilities, including those discussed above, can be combined. You choose the levels and types of optimizations best suited to your application and build constraints, putting ultimate control of how your application builds and runs firmly in your hands.

For more information on optimization, see the *Code optimization with the IBM XL compilers on Power architectures* whitepaper and the IBM XL Fortran *Optimization and Programming Guide*.

Chapter 12. Documentation

IBM XL Fortran supplies comprehensive documentation describing the functionality and capabilities of the compiler.

- The *Compiler Reference* describes all aspects of the compilation process including: compiling, linking, and executing your application. This reference also details the behavior of all command line options available for compilation, and environment variables that you can use to control program execution.
- The *Language Reference* documents the entire Fortran language, including all supported standards. This reference also details the many useful extensions IBM has added to Fortran. It also identifies certain language constructs as being from a particular language revision, such as Fortran 2008, or as an IBM extension.
- The *Optimization and Programming Guide* discusses common programming tasks with IBM XL Fortran, such as OpenMP SMP programming. This guide also details the optimization process to ensure you take advantage of all the optimization capabilities the compiler has to offer, whether you are a beginner, or already have some experience with IBM XL compiler optimization capabilities.
- The *Getting Started with CUDA Fortran programming using XL Fortran* 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. This documentation is applicable to XL Fortran for Linux, V15.1.4, V15.1.5, and V15.1.6 for little endian distributions.
- The *Installation Guide* contains information for installing XL Fortran and configuring your environment for basic compilation and program execution.
- README files and the *Quick Start Guide* are shipped with IBM XL Fortran.
- Man pages for all utilities shipped with the compiler, as well as compiler invocation commands are also included.

IBM XL Fortran for Linux, V15.1.6 also provides the following manuals:

- The *What's New* provides an executive overview of new functions in the latest compiler product, with new functions categorized according to user benefits.
- The *Migration Guide* contains migration considerations for using XL Fortran to compile programs that were previously compiled on different platforms, by previous releases of XL Fortran, or by other compilers.

IBM XL Fortran compilers product documentation is available online. You can access it from the following library page links:

- IBM XL Fortran for AIX: http://www.ibm.com/support/docview.wss?uid=swg27036673
- IBM XL Fortran for Linux: http://www.ibm.com/support/docview.wss?uid=swg27036672

An extensive collection of technical materials, demos, support information, and features and benefits of IBM XL Fortran can be found at the following URL:

http://www.ibm.com/software/products/en/fortcompfami

Chapter 13. Premier customer service

IBM XL Fortran comes with IBM's premier service and support. The IBM Service and Support team is dedicated to providing you with responsive platform and cross-platform software support. For complex or code-related problems, IBM employs specialized service teams with access to compiler development experts. The vision of IBM Service and Support is to achieve a level of support excellence that exceeds customer expectations and differentiates IBM in the marketplace. You will always have access to the right level of IBM expertise when you need it. For more information about the latest updates about supported IBM XL Fortran compilers, see the following resources:.

- Latest updates for supported IBM XL Fortran compilers (http://www.ibm.com/support/docview.wss?uid=swg21156900)
- XL Fortran for Linux support page (https://www.ibm.com/support/home/product/U128148Q26691I65/XL_Fortran_for_Linux)
- XL Fortran for AIX support page (https://www.ibm.com/support/home/product/C844576N76531G06/XL_Fortran_for_AIX)

Chapter 14. Summary

The IBM XL Fortran compilers conform to Fortran standards and include many common industry language extensions and features. They give you the flexibility to easily port your code to the IBM XL Fortran family of compilers on supported platforms. The compilation and optimization technology in IBM XL Fortran is designed to deliver the best performing applications on the Power Architecture®. You can also use the CUDA Fortran or OpenMP 4.5 features to offload computations to the NVIDIA GPUs to increase the computing performance for your programs. IBM XL Fortran is IBM's premier Fortran compiler, and has become synonymous with performance and quality for small and large customers around the world.

Chapter 15. Community Edition and purchasing

IBM XL Fortran for Linux, V15.1.6 Community Edition for little endian distributions is available for download and deployment in December 2017. This compiler product is a no-charge, fully functional product for developers who do not require official IBM support.

The following link contains the XL Fortran compiler product web pages where you can find download information about full versions and the Community Edition if available:

http://www.ibm.com/software/products/en/fortcompfami

Purchasing information of IBM XL Fortran is also available at the above website.

Chapter 16. Contacting IBM

IBM welcomes your comments. You can send them to compinfo@cn.ibm.com.

Printed in USA