

IBM Open XL Fortran for AIX 17.1.2

Getting Started with IBM Open XL Fortran



Note

Before using this information and the product it supports, read the information in [“Notices” on page 23.](#)

First edition

This edition applies to IBM® Open XL Fortran for AIX® 17.1.2 (Program 5765-J19, 5725-C74) 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 International Business Machines Corporation 2023.**

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 |
| Conventions..... | v |
| Related information..... | ix |
| Available help information..... | ix |
| Standards and specifications..... | x |
| Technical support..... | xi |
| How to send your comments..... | xi |
| Inclusive language..... | xi |
| Chapter 1. Introducing IBM Open XL Fortran..... | 1 |
| Operating system and hardware support..... | 1 |
| A highly configurable compiler..... | 2 |
| Supported language levels..... | 2 |
| Source-code migration and conformance checking..... | 3 |
| Utilities and commands..... | 3 |
| Program optimization..... | 5 |
| 64-bit object capability..... | 5 |
| Optimization reports..... | 5 |
| Symbolic debugger support..... | 7 |
| Chapter 2. Setting up and customizing IBM Open XL Fortran..... | 9 |
| Using custom compiler configuration files..... | 9 |
| Chapter 3. Developing applications with IBM Open XL Fortran..... | 11 |
| Compiler phases..... | 11 |
| Editing Fortran source files..... | 11 |
| Compiling with IBM Open XL Fortran..... | 12 |
| Invoking the compiler..... | 12 |
| Compiling parallelized IBM Open XL Fortran applications..... | 14 |
| Specifying compiler options..... | 15 |
| IBM Open XL Fortran input and output files..... | 15 |
| Linking your compiled applications with IBM Open XL Fortran..... | 16 |
| Linking new objects with existing ones..... | 16 |
| Relinking an existing executable file..... | 16 |
| Dynamic and static linking..... | 17 |
| Diagnosing link-time problems..... | 17 |
| Running your compiled application..... | 18 |
| IBM Open XL Fortran compiler diagnostic aids..... | 19 |
| Debugging compiled applications..... | 19 |
| Determining which level of IBM Open XL Fortran is being used..... | 19 |
| Appendix. Accessibility features for IBM Open XL Fortran for AIX..... | 21 |
| Notices..... | 23 |
| Trademarks..... | 25 |
| Index..... | 27 |

About this document

This document contains overview and basic usage information for the IBM Open XL Fortran for AIX 17.1.2 compiler.

Who should read this document

This document is intended for Fortran developers who are looking for introductory overview and usage information for IBM Open 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 IBM Open XL Fortran can use this document to find information about the capabilities and features unique to IBM Open XL Fortran.

How to use this document

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

- An executive overview of new functions: see the *What's New for IBM Open XL Fortran*.
- Compiler installation: see the *IBM Open XL Fortran Installation Guide*.
- Migration considerations and guide: see the *IBM Open XL Fortran Migration Guide*.
- Compiler features including options, commands, and libraries: see the *IBM Open XL Fortran Compiler Reference* for detailed information about the usage of compiler features.
- The Fortran programming language: see the *IBM Open XL Fortran Language Reference* for information about the syntax, semantics, and IBM implementation of the Fortran programming language.

Conventions

Typographical conventions









The following table shows the typographical conventions used in the IBM Open XL Fortran for AIX 17.1.2 documentation.

| 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> |

| Table 1. Typographical conventions (continued) | | |
|--|---|--|
| Typeface | Indicates | Example |
| 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 -O3. |
| 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, this documentation 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 / Fortran 2008 ends | The text describes an IBM Open XL Fortran implementation of the Fortran 2008 standard. ¹ |
|  F2003  | Fortran 2003 begins / Fortran 2003 ends | The text describes an IBM Open XL Fortran implementation of the Fortran 2003 standard, and it applies to all later standards. ¹ |
|  TS 29113  | TS 29113 begins / TS 29113 ends | The text describes an IBM Open XL Fortran implementation of Technical Specification 29113, referred to as TS 29113. ¹ |
|  IBM  | IBM extension begins / IBM extension ends | The text describes a feature that is an IBM Open XL Fortran extension to the standard language specifications. |

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 IBM Open 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 Open 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):

►► keyword — *required_argument* ◄◄

- Optional items are shown below the main path:

►► keyword — *optional_argument* ◄◄

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.

►► keyword — *required_argument1*
required_argument2 ◄◄

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

►► keyword — *optional_argument1*
optional_argument2 ◄◄

- 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:

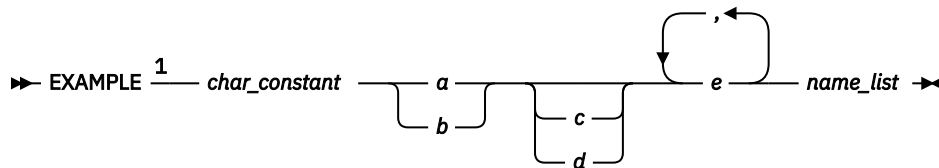
►► keyword — *repeatable_argument* ◄◄

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

►► keyword — *default_argument*
alternate_argument ◄◄

- 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.

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



Notes:

¹ IBM extension

Interpret the diagram as follows:

- Enter the keyword **EXAMPLE**.
- **EXAMPLE** is an IBM extension.
- 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.
- 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. (The *_list* syntax is equivalent to the previous syntax for *e*.)

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 documentation

The examples in this documentation, 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.

Terminology

Some of the terminology in this documentation 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 *IBM Open XL Fortran* often appears as *XLF*.

Related information

The following sections provide related information for IBM Open XL Fortran:

Available help information

IBM Open XL Fortran for AIX information

IBM Open XL Fortran for AIX provides product information in the following formats:

- Quick Start Guide

The Quick Start Guide ([quickstart.pdf](#)) is intended to get you started with IBM Open XL Fortran for AIX 17.1.2. It is located by default in the IBM Open XL Fortran for AIX directory.

- README files

README files contain late-breaking information, including changes and corrections to the product information. README files are located by default in the IBM Open XL Fortran for AIX directory.

- 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 Open XL Fortran for AIX 17.1.2 Installation Guide*.

- Online product documentation

The fully searchable HTML-based documentation is viewable in IBM Documentation at <https://www.ibm.com/docs/openxl-fortran-aix/17.1.2>.

- PDF documents

PDF documents are available online at <https://www.ibm.com/docs/openxl-fortran-aix/17.1.2?topic=pdf-format-documentation>.

The following files comprise the full set of IBM Open XL Fortran for AIX product information.

Note: To ensure that you can access cross-reference links to other IBM Open XL Fortran for AIX PDF documents, download and unzip the .zip file that contains all the product documentation files, or you can download each document into the same directory on your local machine.

| Table 3. IBM Open XL Fortran for AIX PDF files | | |
|--|---------------|--|
| Document title | PDF file name | Description |
| <i>What's New for IBM Open XL Fortran for AIX 17.1.2, GC28-3315-02</i> | whats_new.pdf | Provides an executive overview of new functions in the IBM Open XL Fortran for AIX 17.1.2 compiler, with new functions categorized according to user benefits. |
| <i>Getting Started with IBM Open XL Fortran for AIX 17.1.2, SC28-3313-02</i> | getstart.pdf | Contains an introduction to IBM Open XL Fortran for AIX, with information about setting up and configuring your environment, compiling and linking programs, and troubleshooting compilation errors. |

| Table 3. IBM Open XL Fortran for AIX PDF files (continued) | | |
|--|---------------|--|
| Document title | PDF file name | Description |
| <i>IBM Open XL Fortran for AIX 17.1.2 Installation Guide, GC28-3317-02</i> | install.pdf | Contains information for installing, upgrading, and uninstalling IBM Open XL Fortran for AIX. |
| <i>IBM Open XL Fortran for AIX 17.1.2 Migration Guide, GC28-3314-02</i> | migrate.pdf | Contains migration considerations for using IBM Open XL Fortran for AIX to compile programs that were previously compiled on different platforms, by previous IBM Open XL Fortran for AIX releases, or by other compilers. |
| <i>IBM Open XL Fortran for AIX 17.1.2 Compiler Reference, SC28-3316-02</i> | compiler.pdf | Contains information about compiler features including options, commands, and libraries. |
| <i>IBM Open XL Fortran for AIX 17.1.2 Language Reference, SC28-3318-02</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. |

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>.

For more information about the compiler, see C/C++ and Fortran compilers on the IBM Power® community at <http://ibm.biz/openxl-power-compilers>.

Other IBM information

- *Parallel Environment for AIX: Operation and Use*
- The IBM Systems documentation, at <https://www.ibm.com/docs/aix>, is a resource for AIX information.

You can find the following books for your specific AIX system:

- *AIX Commands Reference, Volumes 1 - 6*
- *Technical Reference: Base Operating System and Extensions, Volumes 1 & 2*
- *AIX National Language Support Guide and Reference*
- *AIX General Programming Concepts: Writing and Debugging Programs*
- *AIX Assembler Language Reference*

Standards and specifications

IBM Open 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 IBM Open 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)* available at <http://www.openmp.org>.

Technical support

Additional technical support is available from the IBM Open XL Fortran Support page at <https://www.ibm.com/mysupport/s/topic/OTO0z0000006v5WGAQ/xl-fortran?productId=01t0z000007g71hAAA>. This page provides a portal with search capabilities to a large selection of Technotes and other support information.

If you have any question on the product, raise it in the [IBM C/C++ and Fortran compilers on Power community](#) or open a case at <https://www.ibm.com/mysupport/s/topic/OTO0z0000006v5WGAQ/xl-fortran?productId=01t0z000007g71hAAA>.

For the latest information about IBM Open XL Fortran and IBM XL Fortran, visit the product information site at <https://www.ibm.com/products/open-xl-fortran-aix-compiler-power>.

How to send your comments

Your feedback is important for helping IBM to provide accurate and high-quality information. If you have any comments or questions about this document or any other IBM Open XL Fortran documentation, send an email to compinfo@cn.ibm.com.

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

Inclusive language

As other industry leaders join IBM in embracing the use of inclusive language, IBM will continue to update the documentation, product code, and user interfaces to reflect those changes. While IBM values the use of inclusive language, terms that are outside of IBM's direct influence are sometimes required for the sake of maintaining user understanding.

To learn more about this initiative, read the [Words matter blog](#) on ibm.com®

Chapter 1. Introducing IBM Open XL Fortran

IBM Open XL Fortran for AIX 17.1.2 fully incorporates the LLVM compiler infrastructure. With the new compiler infrastructure, you can enjoy the combination of IBM's strength in compiler optimization technology with LLVM open-source infrastructure.

IBM Open XL Fortran for AIX 17.1.2 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 IBM Open XL Fortran compiler. It is intended for people who are evaluating the compiler and for users who want to find out more about the product.

Operating system and hardware support

This section describes the operating systems and hardware that IBM Open XL Fortran for AIX 17.1.2 supports.

IBM Open XL Fortran for AIX 17.1.2 supports the following operating systems:

- IBM AIX 7.2: TL5 SP3 or later
- IBM AIX 7.3: TL0 or later

See "Prerequisites" in the *IBM Open XL Fortran Installation Guide* for a complete list of requirements.



Attention:

- The operating system has higher requirements when you use Profile Guided Optimization (PGO) and Link Time Optimization (LTO):

Table 4. System requirements of PGO and LTO

| Feature | Minimum requirement for AIX 7.2 | Minimum requirement for AIX 7.3 |
|---------------------------------------|---------------------------------|---------------------------------|
| PGO | AIX 7.2 TL5 SP5 ¹ | AIX 7.3 TL1 ¹ |
| LTO, which is enabled by -qlto | AIX 7.2 TL5 SP5 ² | AIX 7.3 TL1 ² |
| -qfat-lto-objects | AIX 7.2 TL5 SP5 | AIX 7.3 TL1 |
| -qlto=thin | AIX 7.2 TL5 SP7 | AIX 7.3 TL2 |

Notes:

1. If your OS version is lower than the required one, compiling with the **-qprofile-generate** option or using the **ibm-llvm-profddata** utility will fail, and linking with PGO enabled will result in link-time failures. Find details of the PGO feature in "Profile Guided Optimization (PGO)" in the *IBM Open XL Fortran Migration Guide*.
2. LTO is also supported on IBM AIX 7.2 TL5 SP4 or earlier, and IBM AIX 7.3 TL0 SP2 or earlier; however, note that non-default visibility symbols defined in the objects that are compiled with **-qlto** are not necessarily retained to satisfy references from the objects that are compiled without **-qlto** during linking. You can find details of the LTO feature in "Link Time Optimization (LTO)" in the *IBM Open XL Fortran Migration Guide*.

The compiler, its libraries, and its generated object programs run on Power7, Power7+, Power8, Power9, or Power10 processor-based systems 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

IBM Open XL Fortran provides several commands to invoke the compiler, for example, **xlf**, **xlf90**, **xlf95**, **xlf2003**, and **xlf2008**.

The compiler also provides corresponding **_r** versions of most invocation commands, for example, **xlf_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 IBM Open XL Fortran compiler invocation commands, see ["Compiling XL Fortran programs"](#) in the *IBM Open 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 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 IBM Open XL Fortran compiler options, see ["Summary of compiler options"](#) in the *IBM Open XL Fortran Compiler Reference*.

Custom compiler configuration files

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

If you frequently specify compiler option settings other than the default settings of IBM Open 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 9.

Supported language levels

This topic describes the Fortran programming language specifications that IBM Open XL Fortran for AIX 17.1.2 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, IBM Open 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

Source-code migration and conformance checking

IBM Open 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 **xl**f or **xl**f_r to automatically select the language level appropriate to the filename extension.

You can rebuild your FORTRAN 77, Fortran 90, Fortran 95, Fortran 2003, and Fortran 2008 source code with IBM Open XL Fortran for AIX 17.1.2 and link them all into the same application. Similarly, object code or libraries compiled with previous versions of IBM XL Fortran are still compatible with the newest IBM Open XL Fortran compiler and runtime environment, except for two cases.

Related information

[-qlanglvl](#)

Utilities and commands

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

Utilities

CreateExportList

The **CreateExportList** utility creates a file that contains a list of all the global symbols found in a given set of object files. For more information, see [Exporting symbols with the CreateExportList utility](#) in the *IBM Open XL Fortran Compiler Reference*.

ibm-bugpoint

The **ibm-bugpoint** utility enables you to create a minimal test case that is written in LLVM intermediate language so that you can reproduce the problems you encounter when using the compiler. For details on how to use **ibm-bugpoint**, run this utility with the **--help** option. For details of the open-source LLVM **bugpoint** utility, see <https://ibm.biz/openxl-1712-llvm-bugpoint>.

ibm-gen-list

This utility enables you to run annotated profiling with the **tprof** tool. **ibm-gen-list** generates listing files directly from binaries or object files and depends on the **objdump** tool to disassemble the binaries or object files. In addition, to add line numbers in listing files, use **ibm-gen-list** together with the **llvm-dwarfdump** tool. Find more information of this utility at [tprof listing annotation with IBM Open XL compilers for AIX](#).

Note: For the **objdump** tool, you can either use [llvm-objdump](#) or install the GNU **objdump** tool from Toolbox for GNU Open Source Software.

ibm-llvm-ar

The **ibm-llvm-ar** utility archives several files, such as objects and LLVM bitcode files, into a single archive library that can be linked into a program. **ibm-llvm-ar** has the same functionality as the open-source LLVM **llvm-ar** utility. For details of **llvm-ar**, see <https://ibm.biz/openxl-1712-llvm-ar>.

ibm-llvm-extract

The **ibm-llvm-extract** utility extracts the bitcode of specified functions from a given LLVM bitcode file. It is primarily used as a debugging tool to reduce test cases from large programs. Additionally, this utility removes unreachable global variables, prototypes, and unused types from the function bitcode. **ibm-llvm-extract** has the same functionality as the open-source LLVM **llvm-extract** utility. For details of **llvm-extract**, see <https://ibm.biz/openxl-1712-llvm-extract>.

Note: The **ibm-llvm-extract** utility is available starting from IBM Open XL Fortran for AIX 17.1.2.7.

ibm-llvm-profdata

The **ibm-llvm-profdata** utility is used for handling profile data files. It has several subcommands, described as follows:

- The **ibm-llvm-profdata show** utility takes a profile data file and displays Profile-Guided Feedback (PGO) information in this file.
- The **ibm-llvm-profdata merge** utility takes several profile data files generated by PGO instrumentation and merges them into a single indexed profile data file.
- The **ibm-llvm-profdata overlap** utility takes two profile data files and displays the overlap of counter distribution between the entire files and among any of the specified functions.
- The **ibm-llvm-profdata clean** utility takes the specified profile data files and deletes them. This utility deletes raw profile data files or instrumented profile data files created by the **ibm-llvm-profdata merge** utility. If a directory is specified, the utility removes all profile data files within that directory. However, it does not delete text-format profile data files or any profile files generated by IBM XL Fortran for AIX 16.1.0 or earlier releases.

Notes:

- The **ibm-llvm-profdata clean** utility is available starting from IBM Open XL Fortran for AIX 17.1.2.6.

For details on how to use **ibm-llvm-profdata**, run this utility with the **--help** option. For details of the open-source LLVM **llvm-profdata** utility, see <https://ibm.biz/openxl-1712-llvm-profdata>.

ibm-llvm-ranlib

The **ibm-llvm-ranlib** utility is an alias for **ibm-llvm-ar** that generates an index for one or more archives. **ibm-llvm-ranlib** has the same functionality as the open-source LLVM **llvm-ranlib** utility. For details of **llvm-ranlib**, see <https://ibm.biz/openxl-1712-llvm-ranlib>.

ibm-llvm-symbolizer

This utility reads object file names and addresses from the command line and prints corresponding source code locations to standard output. **ibm-llvm-symbolizer** has the same functionality as the open-source LLVM **llvm-symbolizer** utility. For details of **llvm-symbolizer**, see <https://ibm.biz/openxl-1712-llvm-symbolizer>.

Note: These compiler utilities are shipped in the /opt/IBM/openxlf/17.1.2/bin/ directory with the exception of the following ones, which are shipped in the /opt/IBM/openxlf/17.1.2/tools/ directory.

- **ibm-bugpoint**
- **ibm-llvm-ar**
- **ibm-llvm-ranlib**
- **ibm-llvm-symbolizer**

Commands

xlfndi

The **xlfndi** script installs IBM Open XL Fortran to a nondefault directory location. For more information, see [Updating a nondefault installation using xlfndi](#) in the *IBM Open XL Fortran Installation Guide*.

Program optimization

IBM Open XL Fortran incorporates the LLVM compiler infrastructure, so the compiler fully supports the optimization features from LLVM. The compiler 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 specific machine configuration, depending on where the program will run

Optimizing transformations can give your application better overall execution performance. IBM Open 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

For more details, see "Clang Compiler User's Manual" in the [Clang documentation](#).

Related information

[Optimization and tuning](#)

[Intrinsic procedures](#)

64-bit object capability

The 64-bit object capability of the IBM Open XL Fortran compiler addresses increasing demand for larger storage requirements and greater processing power.

The AIX operating system provides an environment that allows you to develop and execute programs that exploit 64-bit processors through the use of 64-bit address spaces.

To support larger executables that can fit within a 64-bit address space, a separate 64-bit object format is used. The binder binds these objects to create 64-bit executables. Objects that are bound together must all be of the same object format. The following scenarios are not permitted and will fail to link, load, or execute:

- A 64-bit object or executable that has references to symbols from a 32-bit library or shared library
- A 32-bit object or executable that has references to symbols from a 64-bit library or shared library
- A 64-bit executable that explicitly attempts to load a 32-bit module
- A 32-bit executable that explicitly attempts to load a 64-bit module
- Attempts to run 64-bit applications on 32-bit platforms

On both 64-bit and 32-bit platforms, 32-bit executables will continue to run as they currently do on a 32-bit platform.

IBM Open XL Fortran supports 64-bit mode mainly through the use of the **-q64** and **-qarch** compiler options. This combination determines the bit mode and instruction set for the target architecture.

For more information, see "[Using XL Fortran in a 64-bit environment](#)" in the *IBM Open XL Fortran Compiler Reference*.

Optimization reports

IBM Open XL Fortran provides access to optimization reports via the LLVM remarks framework.

The LLVM remarks framework and options are documented in <https://ibm.biz/openxl-1712-llvm-remarks>. Each option must be passed to one of `ibm-opt`, `ibm-llc`, or `LTO` via the `-X` or `-W` options. IBM Open

XL Fortran also provides the **-qreport** option, which generates a report in the listing file that includes LLVM remarks for function inlining and loop transformations. If you specify the **-qreport** option, do not also specify any other options from the LLVM remarks framework because they will interfere with report generation.

Please note the following when using the LLVM remarks framework options:

- If you use the **--pass-remarks-output=<filename>** option to save the optimization remarks to a file, make sure that you request remarks from only one of **ibm-opt**, **ibm-llc**, or **LTO**, or to save the remarks from each component in a separate file. For example, the following command saves the remarks from **ibm-opt** to **file.opt.yaml** and the remarks from **ibm-llc** to **file.llc.yaml**:

```
xlf95 file.f -O3 \  
-Xopt --pass-remarks-output=file.opt.yaml \  
-Xllc --pass-remarks-output=file.llc.yaml
```

The following command writes remarks from **ibm-opt** to **file.yaml** and then overwrites the file with the remarks from **ibm-llc**:

```
xlf95 file.f -O3 -mllvm --pass-remarks-output=file.yaml
```

- You must enable debug line number generation (**-g1** or **-g**) to get line numbers in the optimization remarks, or to use external visualization tools like **opt-viewer.py**.
- Many LLVM optimizations produce remarks. You can use the filtering options to control the amount of information presented.

The following example demonstrates generating optimization reports using the **-qreport** option as well as the LLVM remarks framework:

```
$ cat loop.f  
module m  
contains  
subroutine sub(a, b, c)  
  real a(160), b(160), c(160)  
  integer i  
  
  do i = 1, 160  
    a(i) = b(i) + c(i)  
  end do  
  
  do i = 1, 8  
    a(i) = a(i) * 3  
  end do  
end  
  
subroutine sub2(a, b, c)  
  real a(160), b(160), c(160)  
  call sub(a, b, c)  
end subroutine  
end module  
  
program main  
  use m  
  real a(160), b(160), c(160)  
  read(*, *) b, c  
  call sub2(a, b, c)  
  print *, a  
end program  
$
```

Optimization report using -qreport

This command generates **loop.lst**:

```
xlf95 loop.f -c -O3 -qarch=pwr9 -qreport
```

loop.lst contains the following optimization report:

```
>>>> LOOP TRANSFORMATION SECTION <<<<  
< loop.f  
1      | module m
```

```

2      | contains
3      | subroutine sub(a, b, c)
4      |     real a(160), b(160), c(160)
5      |     integer i
6      |
7  U5V4,8 |     do i = 1, 160
8      |         a(i) = b(i) + c(i)
9      |     end do
10     |
11  U8    |     do i = 1, 8
12     |         a(i) = a(i) * 3
13     |     end do
14     | end
15     |
16     | subroutine sub2(a, b, c)
17     |     real a(160), b(160), c(160)
18  I    |     call sub(a, b, c)
19     | end subroutine
20     | end module
21     |
22     | program main
23     |     use m
24     |     real a(160), b(160), c(160)
25     |     read(*, *) b, c
26     |     call sub2(a, b, c)
27     |     print *, a
28     | end program

```

The letters in the left margin indicate the transformation performed. U<iterations> means the loop was unrolled with unroll factor <iterations>. V<width, count> means the loop was auto-vectorized with vector width <width> and interleaved count <count>. I means the function call was inlined.

Optimization report using LLVM remarks

Compile using the following command:

```

xlf95 loop.f -c -O3 -qarch=pwr9 -g1 \
  -Xopt --pass-remarks-output=loop.opt.yaml \
  -Xl1c --pass-remarks-output=loop.llc.yaml

```

Install python3. Download the opt-viewer tool from <https://llvm.org>.

Set up the python3 environment for opt-viewer.py:

```

python3 -m virtualenv optviewer_env
. optviewer_env/bin/activate
python3 -m pip install pyyaml pygments

```

Use opt-viewer.py to generate the HTML report:

```

opt-viewer/opt-viewer.py --demangler cat -source-dir $PWD *.yaml -o report

```

The commands above generate a report directory containing optimization report loop.f.html. The report includes annotated source with transformation reports from ibm-opt and ibm-llc. Transformations that were performed are highlighted in green. Transformations that were missed are highlighted in red.

Related information

- [-qreport](#) in the *IBM Open XL Fortran Compiler Reference*

Symbolic debugger support

You can instruct IBM Open XL Fortran to include debugging information in your compiled objects.

The debugging information can be examined by symbolic debuggers that support the AIX XCOFF executable format and the DWARF debug format.

Chapter 2. Setting up and customizing IBM Open 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 IBM Open XL Fortran, see [Prerequisites](#) in the *IBM Open 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 IBM Open 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 3. Developing applications with IBM Open XL Fortran

By default, when you invoke the IBM Open XL Fortran compiler, all of the following phases of translation are performed. Usually, compiling and linking are combined into a single step

- Preprocessing of program source
- Compiling and assembling into object files
- Linking into an executable

You can use compiler options to perform only certain phases, such as preprocessing, or assembling. You can then re-invoke the compiler to resume processing of the intermediate output to a final executable.

The following sections describe how to invoke the IBM Open XL Fortran compiler to preprocess, compile, and link source files and libraries:

Notes:

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

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

Some separation of these phases is observable with the **-v** compiler option. To see the amount of time the compiler spends in various phases, specify **-qphsinfo**.

Editing Fortran source files

To create Fortran source programs, you can use any text editor available on your system, such as **vi** or **emacs**.

Source programs must be saved using a recognized file name suffix. See “[IBM Open XL Fortran input and output files](#)” on page 15 for a list of suffixes recognized by IBM Open XL Fortran.

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

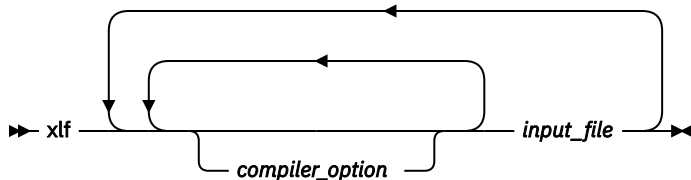
Compiling with IBM Open XL Fortran

IBM Open 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.
- 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 *IBM Open 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 .f08, .f03, .f95, .f90, or .f77.

| Table 5. Invocation commands and corresponding Fortran language standards | | | |
|---|---|--|---|
| Language level | Invocation commands | Notes | |
| Fortran 2008 | <ul style="list-style-type: none">• f2008• xlf2008 | 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. | The Fortran 2008 language standard is partially supported in this release. |
| Fortran 2003 | <ul style="list-style-type: none">• f2003• xlf2003 | | |
| Fortran 95 | <ul style="list-style-type: none">• f95• xlf95 | | 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. |
| Fortran 90 | <ul style="list-style-type: none">• f90• xlf90 | | |

Table 5. Invocation commands and corresponding Fortran language standards (continued)

| Language level | Invocation commands | Notes |
|----------------|---|---|
| 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 IBM Open 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 *IBM Open XL Fortran Compiler Reference* for more information about compiler invocation commands available to you.

Compiling parallelized IBM Open XL Fortran applications

IBM Open XL Fortran provides threadsafe compiler invocation commands to compile parallelized applications for use in multiprocessor environments.

These invocations are similar to their corresponding base compiler invocations, except that they link and bind compiled objects to threadsafe components and libraries. The generic IBM Open XL Fortran threadsafe compiler invocation is `xlf_r`.

IBM Open XL Fortran provides additional threadsafe invocations to meet specific compilation requirements. For more information, see ["Compiling XL Fortran programs"](#) in the *IBM Open XL Fortran Compiler Reference*.

Note: Using any of these commands alone does not imply parallelization.

POSIX Pthreads API support

IBM Open XL Fortran supports 64-bit thread programming with the 1003.1-1996 (POSIX) standard Pthreads API. It also supports 32-bit programming with the 1003.1-1996 standard API.

You can use invocation commands (which use corresponding stanzas in the `xlf.cfg` configuration file) to compile and then link your programs with the 1003.1-1996 standard interface libraries.

- To compile and then link your program with the 1003.1-1996 standard interface libraries, use the **_x** variants of the compiler invocation commands. For example, you could specify:

Specifying compiler options

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

Priority sequence of compiler options

When specifying compiler options, it is possible for option conflicts and incompatibilities to occur. The IBM Open XL Fortran compiler resolves most of these conflicts and incompatibilities in a consistent fashion, as follows:

In most cases, the compiler uses the following order in resolving conflicting or incompatible options:

1. Pragma statements in source code override compiler options specified on the command line.
2. Compiler options specified on the command line override compiler options specified as environment variables or in a configuration file. If conflicting or incompatible compiler options are specified in the same command line compiler invocation, the subsequent option in the invocation takes precedence.
3. Compiler options specified as environment variables override compiler options specified in a configuration file.
4. Compiler options specified in a configuration file, command line or source program override compiler 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 xlf.cfg file before it searches the directories specified with **-I** on the command line. The **-I** option is cumulative rather than preemptive.

Related information

[Specifying options on the command line](#)

IBM Open XL Fortran input and output files

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

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

| Table 6. Input file types | |
|--|--------------------------|
| Filename extension | Description |
| .a | Archive or library files |
| .f, .F, .f77, .F77, .f90, .F90, .f95, .F95, .f03, .F03, .f08, .F08 | Fortran source files |
| .mod | Module symbol files |
| .smod 1 | Submodule symbol files |
| .o | Object files |
| .s | Assembler files |
| .so | Shared object files |

| Table 6. Input file types (continued) | |
|---------------------------------------|-------------|
| Filename extension | Description |
| Note: 1 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 |
| .o | Object files |
| .s | Assembler files |
| .so | Shared object files |
| Note: 1 Fortran 2008 | |

Linking your compiled applications with IBM Open XL Fortran

By default, you do not need to do anything special to link an IBM Open 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
```

Linking new objects with existing ones

If you have `.o` or other object files that you compiled with an earlier versions of the compiler, you can link them with object files that you compile with the current level of IBM Open XL Fortran.

See ["Linking new objects with existing ones"](#) in the *IBM Open XL Fortran Compiler Reference* for more information.

Relinking an existing executable file

The linker accepts executable files as input, so you can link an existing executable file with updated object files.

You cannot, however, relink executable files that were previously linked using the `-qlto` option.

If you have a program consisting of several source files and only make localized changes to some of the source files, you do not necessarily have to compile each file again. Instead, you can include the executable file as the last input file when compiling the changed files:

```

xlf95 -o mansion front_door.f entry_hall.f parlor.f sitting_room.f \
    main_bath.f kitchen.f dining_room.f pantry.f utility_room.f

vi kitchen.f # Fix problem in OVEN subroutine

xlf95 -o newmansion kitchen.f mansion

```

Limiting the number of files to compile and link the second time reduces the compilation time, disk activity, and memory use.

Dynamic and static linking

You can use IBM Open 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 IBM Open 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.

Diagnosing link-time problems

You can use AIX commands and linker options to address unresolved and duplicate symbols that you might encounter.

Unresolved symbols

When an external symbol cannot be resolved, a linker error is issued. The example messages are as follows:

```

ld: 0711-317 ERROR: Undefined symbol: .function_name
ld: 0711-345 Use the -bloadmap or -bnoquiet option to obtain more information.

```

The most common reason of unresolved symbol errors is missing input files. You can take one of the following actions to locate the symbols and include the target input files in which the symbols are defined during linking:

- Include all the supplied libraries during linking so that the linker can find where the symbols are.
- Use the **nm** command to list symbols in libraries or object files so that you can find out the target input files.
- Use the **dump -Tv** command to examine the deferred imported symbols, which have [noIMid] value for the IMPid field in the output. You must resolve those deferred symbols to avoid unresolved symbols.
- Use one of the following linker options to generate linker log files and analyze the log files to determine the libraries or object files that reference unresolved symbols. These options examine interdependent or redundant libraries that are used in error.
 - The **-bnoquiet** option writes each binder subcommand and its results to standard output. It lists unresolved symbol references and the symbols that are imported from the specified library modules.
 - The **-bmap:filename** option generates an address map. Unresolved symbols are listed at the top of the map, followed by imported symbols.

- The **-bloadmap:filename** option generates a linker log file. The linker log file includes information about all of the arguments passed to the linker, the names of the shared objects, and the number of imported symbols. If any unresolved symbol is found, the linker log file lists the object file or shared object that references the symbol.

Duplicate symbols

When two or more functions in the linked object files have the same name, a warning message is issued. The example messages are as follows:

```
ld: 0711-224 WARNING: Duplicate symbol: .function_name
ld: 0711-345 Use the -bloadmap or -bnoquiet option to obtain more information.
```

By default, the linker uses the first definition that it encounters, which might not produce expected results. To fix this problem, you can rename the function.

To find the symbols that are defined in multiple files, you can use the **nm** command.

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 IBM Open XL Fortran compiler is `a.out`. You can select a different name with the **-o** compiler option.

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, 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 IBM Open 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 *IBM Open XL Fortran Installation Guide*.

Running compiled applications on other systems

In general, applications linked on a system using an earlier version of AIX can run with more recent versions of AIX. However, applications linked on a system using a newer version of AIX might not necessarily run with earlier versions of AIX.

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

You can obtain the latest IBM Open XL Fortran Runtime Environment images, together with licensing and usage information, from the [Open XL Fortran for AIX support page](#).

libc++ load compatibility

When you run an application, load it against the `libc++.a`, `libc++abi.a`, `libunwind.a` runtime filesets at the same or a higher level than those that the application was linked against. Otherwise, if the application is loaded against an older version of these filesets, some objects or symbols might not be found and an error message similar to the following one will be issued:

```
exec(): 0509-036 Cannot load program ${PROGRAM_NAME} because of the following errors:
0509-130 Symbol resolution failed for ${PROGRAM_NAME} because:
0509-136 Symbol _ZNSt3_122_libcpp_verbose_abortEPKcz (${NUMBER}) is not exported from
dependent module ${RUNTIME_LOCATION}/libc++.a(shr.o).
```

IBM Open XL Fortran compiler diagnostic aids

IBM Open XL Fortran issues diagnostic messages when it encounters problems during compilation of your application.

You can use messages and other information provided in optimization reports to help identify and correct such problems.

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

Debugging compiled applications

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

At compile time, you can use the **-g** or **-qlinedebug** option to instruct the IBM Open 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 *IBM Open XL Fortran Compiler Reference*.

You can then use **dbx** or any symbolic debugger that supports the AIX XCOFF executable format to step through and inspect the behavior of your compiled application.

Optimized applications pose special challenges when you debug your applications. If you need to debug an optimized application, you can consider using the **-gN** form of the **-g** option along with any optimization options. This form of the **-g** option provides different levels of tradeoff between full optimization and full debugging support, depending on the value of N.

Determining which level of IBM Open XL Fortran is being used

To display the version and release level of IBM Open 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 Open XL Fortran for AIX

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

Accessibility features

IBM Open XL Fortran for AIX uses the latest W3C Standard, [WAI-ARIA 1.0](http://www.w3.org/TR/wai-aria/) (<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) (<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/) (<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 Open XL Fortran for AIX online product documentation in IBM Documentation is enabled for 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 Open XL Fortran for AIX 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 Open XL Fortran for AIX, see the following information:

- [IBM AIX](https://www.ibm.com/support/knowledgecenter/en/ssw_aix) (https://www.ibm.com/support/knowledgecenter/en/ssw_aix)

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](http://www.ibm.com/able) (www.ibm.com/able).

Notices

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

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.

Licensees 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, 2023.

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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

Special Characters

-qversion [19](#)

Numerics

64-bit object [5](#)

A

AIX commands [17](#)

AIX XCOFF [7](#)

C

command [3](#)

compiler option [2](#)

configuration file

 customization [2](#)

 default file [2](#)

CreateExportList utility [3](#)

customization

 compiler [9](#)

 configuration file [9](#)

D

debug

 DWARF [7](#)

diagnosis [17](#)

diagnostic message [19](#)

diagnostic messages [3](#)

DWARF [7](#)

dynamic linking [17](#)

E

emacs [11](#)

environment variable [2](#)

execution

 applications [18](#)

F

file type

 input [15](#)

 output [15](#)

filename extension [3](#)

H

hardware [1](#)

I

ibm-bugpoint [3](#)

ibm-gen-list [3](#)

ibm-llc [5](#)

ibm-llvm-ar [3](#)

ibm-llvm-extract [4](#)

ibm-llvm-profdata [4](#)

ibm-llvm-ranlib [4](#)

ibm-opt [5](#)

input file type [15](#)

installation [9](#)

invocation

 applications [18](#)

 default [11](#)

 phases [11](#)

 threadsafe [14](#)

invocation command

 language standard [13](#)

L

language extension [3](#)

language level

 selection [2](#)

 supported [2](#)

linker error [17](#)

linker option [17](#)

linking

 applications [16](#)

 dynamic [17](#)

 objects [16](#)

 relinking [16](#)

 static [17](#)

LLVM [1](#)

LLVM optimization [5](#)

LLVM remarks

 optimization reports [5](#)

LTO [5](#)

M

migration

 source code [3](#)

multiprocessor environment [14](#)

O

operating system [1](#)

optimization [5](#)

optimization reports [5](#), [19](#)

optimize [2](#)

option

 incompatibility

 resolution [15](#)

 priority sequence [15](#)

output file type [15](#)

P

parallelized application [14](#)

POSIX Pthreads API [14](#)

programs

running [18](#)

S

set up [9](#)

source file

edit [11](#)

standard language level [2](#)

static linking [17](#)

support

hardware [1](#)

operating system [1](#)

symbol

duplicate [18](#)

unresolved [17](#)

symbolic debuggers [7](#)

T

threadsafe [12](#), [14](#)

tool [3](#)

U

utility

CreateExportList [3](#)

ibm-bugpoint [3](#)

ibm-gen-list [3](#)

ibm-llvm-ar [3](#)

ibm-llvm-extract [4](#)

ibm-llvm-profdata [4](#)

ibm-llvm-ranlib [4](#)

V

version level [19](#)

vi [11](#)



Product Number: 5765-J19, 5725-
C74

SC28-3313-02

