Get Started with the Intel® Fortran Compiler Classic and Intel® Fortran Compiler

Contents

Get Started with the Intel® Fortran Compiler Classic and Intel® Fortran Compiler

1

Attention
Intel® Fortran Compiler Classic (`ifort`) is now deprecated and will be discontinued in October 2024. Intel recommends that customers transition now to using the LLVM-based Intel® Fortran Compiler (`ifx`) for continued Windows* and Linux* support, new language support, new language features, and optimizations.

For the latest information on transitioning from `ifort` to `ifx`, see the Porting Guide for ifort Users to ifx.

The Intel® Fortran Compiler Classic and Intel® Fortran Compiler provide optimizations that help your applications to run faster on Intel® 64, and IA-32 architectures, with support for the latest Fortran language standards. This compiler produces optimized code that can run significantly faster by taking advantage of the ever-increasing core count and vector register width in Intel® Xeon® processors and compatible processors. The Intel® Compiler will help you boost application performance through superior optimizations and Single Instruction Multiple Data (SIMD) vectorization, integration with Intel® Performance Libraries, and by leveraging the OpenMP* 5.0/5.1 parallel programming model.

Start using the compiler from the command line or within Microsoft Visual Studio*.

NOTE

- macOS is no longer supported for Intel® Fortran Compiler Classic (`ifort`).
- Support for 32-bit targets is deprecated in `ifort` and may be removed in a future release. `ifx` does not support 32-bit targets.

Use of the Intel® Fortran Compiler

The Intel® Fortran Compiler (`ifx`) is a new compiler based on the Intel® Fortran Compiler Classic (`ifort`) frontend and runtime libraries, using LLVM backend technology. At this time, `ifx` supports features of the Fortran 95 language, and most OpenMP 5.0/5.1 directives and offloading features. `ifx` is binary (.o/.obj) and module (.mod) file compatible; binaries and libraries generated with `ifort` can be linked with binaries and libraries built with `ifx`, and .mod files generated with one compiler can be used by the other. Both compilers use the `ifort` runtime libraries.

The `-fp-model fast` and `-fp-model fast=2` options behave differently with `ifx` and `ifort`. With `ifort`, floating point compares happen as specified by the IEEE floating point standard, in that the code sequence generated for them assumes a compare can involve a NaN. `ifx` does not generate the check for NaN operands. If using `-fp-model fast` or `-fp-model fast=2` with `ifx` and NaN compares are desired to match `ifort`'s behavior, the -assume nan_compare option should be added to the command line.

Find More

NOTE
Explore the complete list of oneAPI code samples in the oneAPI Samples Catalog (GitHub*). These samples were designed to help you develop, offload, and optimize multiarchitecture applications targeting CPUs, GPUs, and FPGAs.

Document	Description
Intel® Fortran Compiler Classic and Intel® Fortran Compiler Developer Guide and Reference	The Developer Guide and Reference contains information on: • How to use the command line or Microsoft* Visual Studio*or Xcode*or the Eclipse* CDT • Support for the latest compiler technologies and architectures. • Compiler reference material, including options, program structures, class and math libraries, and much more
Intel® Fortran Compiler Documentation	Explore tutorials, training materials, and other Fortran documentation.
Intel® Fortran Compiler Classic and Intel® Fortran Compiler Release Notes	Information on product installation, new and changed features, and issues that are not described in the product documentation.
Intel® Fortran Compiler Forum	Ask questions and find answers in the Intel® Fortran Compiler forum.
Layers for Yocto* Project	Add oneAPI components to a Yocto project build using the meta-intel layers.

Notices and Disclaimers

Get Started on Linux*

Before You Begin

Set Environment Variables for CLI Development

> **NOTE**
> The Unified Directory Layout was implemented in 2024.0. If you have multiple toolkit versions installed, the Unified layout ensures that your development environment contains the correct component versions for each installed version of the toolkit.
>
> The directory layout used before 2024.0, the Component Directory Layout, is still supported on new and existing installations.
>
> For detailed information about the Unified layout, including how to initialize the environment and advantages with the Unified layout, refer to Use the setvars and oneapi-vars Scripts with Linux.

Compiler environment variables must first be configured if using the compiler from a Command Line Interface (CLI). Environment variables are set up with a script called setvars in the Component Directory Layout or oneapi-vars in the Unified Directory Layout. By default, changes to your environment made by sourcing the `setvars.sh` or `oneapi-vars.sh` script apply only to the terminal session in which the environment script was sourced. You must source the script for each new terminal session.

Detailed instructions on using the `setvars.sh` or `oneapi-vars.sh` script are found in Use the setvars and oneapi-vars Scripts with Linux.

Optionally use one-time setup for `setvars.sh` as described in Use Modulefiles with Linux*.

> **NOTE**
> The conda package for the Intel® Fortran Compiler runtime no longer has a runtime dependency on the Intel® MPI Library, which is needed to enable Coarrays. If you maintain a conda package that has a runtime dependency on the Intel Fortran Compiler runtime and your application uses the Intel MPI Library, you need to explicitly add the impi_rt conda package for the Intel MPI Library to the list of runtime dependencies in your project's `meta.yaml` file.

Invoke the Compiler From the Command Line

Invoke the compiler on the command line using the following syntax:

```
{ifx|ifort} [option] file1 [file2...] [/link link_options]
```

For example:

```
ifx hello.f90
```

Build a Program From the Command Line

Use the following steps to test your compiler installation and build a program.

1. Use a text editor to create a file called `hello.f90` with the following contents:

```
print *, "Hello, world!"
end
```

2. Open a terminal window.
3. If you are not using one-time setup for `setvars.sh`, set environment variables by sourcing setvars:

Component Directory Layout

```
source <install-dir>/setvars.sh
```

Unified Directory Layout

```
source <install-dir>/<toolkit-version>/oneapi-vars.sh
```

For information about the <install-dir> location for Component or Unified layout on system-wide or private installations, refer to Using the setvars and oneapi-vars Scripts with Linux*.

4. From the terminal window, issue the following command to compile `hello.f90`:

```
ifx -o hello hello.f90
```

5. Now you have an executable called `hello`, which can be run and will give immediate feedback.

```
hello
```

Which outputs:

```
Hello, world!
```

Next Steps

- Explore the latest oneAPI Fortran Code Samples.
- Explore the Intel® Fortran Compiler Classic and Intel® Fortran Compiler Developer Guide and Reference.

Get Started on macOS*

Before You Begin

NOTE Starting with the 2024.0 release, macOS is not supported by the `ifx` compiler.

Set Environment Variables for CLI Development

NOTE
The Unified Directory Layout was implemented in 2024.0. If you have multiple toolkit versions installed, the Unified layout ensures that your development environment contains the correct component versions for each installed version of the toolkit.

The directory layout used before 2024.0, the Component Directory Layout, is still supported on new and existing installations.

For detailed information about the Unified layout, including how to initialize the environment and advantages with the Unified layout, refer to Use the setvars and oneapi-vars Scripts with Linux.

Compiler environment variables must first be configured if using the compiler from a Command Line Interface (CLI). Environment variables are set up with a script called setvars in the Component Directory Layout or oneapi-vars in the Unified Directory Layout. By default, changes to your environment made by sourcing the `setvars.sh` or `oneapi-vars.sh` script apply only to the terminal session in which the environment script was sourced. You must source the script for each new terminal session.

Detailed instructions on using the `setvars.sh` or `oneapi-vars.sh` script are found in Use the setvars and oneapi-vars Scripts with Linux.

Optionally use one-time setup for `setvars.sh` as described in Use Modulefiles with Linux*.

Invoke the Compiler From the Command Line

Invoke the compiler using the following syntax:

```
ifort [option] file1 [file2...] [/link link_options]
```

For example: `ifort helloworld.f90`

Build a Program From the Command Line

Follow the steps below to test your compiler installation and build a program.

1. Use a text editor to create a file called `hello.f90` with the following contents:

```
print *, "Hello, world!"
end
```

2. Open a terminal window.
3. If you are not using one-time setup for `setvars.sh`, set environment variables by sourcing setvars:

   Component Directory Layout

```
source <install-dir>/setvars.sh
```

   Unified Directory Layout

```
source <install-dir>/<toolkit-version>/oneapi-vars.sh
```

   For information about the <install-dir> location for Component or Unified layout on system-wide or private installations, refer to Use the setvars and oneapi-vars Scripts with Linux*.
4. From the terminal window, issue the following command to compile `hello.f90`:

```
ifort -o hello hello.f90
```

5. Now you have an executable called `hello`, which can be run and will give immediate feedback.

```
hello
```

   Which outputs:

```
Hello, world!
```

Next Steps

- Explore the latest oneAPI Fortran Code Samples.
- Explore the Intel® Fortran Compiler Classic and Intel® Fortran Compiler Developer Guide and Reference.

Get Started on Windows*

Before You Begin

For building applications with full IDE functionality, including debugging and development, you must install a supported version of Microsoft Visual Studio*. To build applications using command-line tools only, you must have a supported version of Build Tools for Visual Studio installed. **Desktop development with C++** support must be selected for all versions as part of the Visual Studio or Build Tools installation.

- Refer to Installing Microsoft Visual Studio* for Use with Intel® Compilers for instructions on how to select **Desktop development with C++**.
- Refer to Intel® Compilers Compatibility with Microsoft Visual Studio* and Xcode* for detailed information about which version of Microsoft Visual Studio to use with the compiler.

If you open an Intel oneAPI command prompt from the Start menu, you do not need to set the environment variables as they are set automatically. If you open a standard Windows command prompt, you will need to set the environment variables as described in Use the setvars and oneapi-vars Scripts with Windows*.

NOTE
The conda package for the Intel® Fortran Compiler runtime no longer has a runtime dependency on the Intel® MPI Library, which is needed to enable Coarrays. If you maintain a conda package that has a runtime dependency on the Intel Fortran Compiler runtime and your application uses the Intel MPI Library, you need to explicitly add the impi_rt conda package for the Intel MPI Library to the list of runtime dependencies in your project's `meta.yaml` file.

Invoke the Compiler From the Command Line

The following steps show how to invoke the compiler on Windows. Exact steps may vary depending on which version of Windows is installed.

Step 1: Open a command prompt

1. Open the **Start** menu
2. Select **Intel oneAPI command prompt** under the installed version of Intel oneAPI, for example **Intel oneAPI 2024**

Step 2: Invoke the compiler

Invoke the compiler using the following syntax:

```
{ifx|ifort} [option] file1 [file2...] [/link link_options]
```

For example:

```
ifx hello.f90
```

To display all available compiler options, use the following command:

- `ifx /help`

Refer to Compiler Options for detailed information about available options.

Invoke the Compiler From Microsoft Visual Studio

The following steps show how to invoke the compiler from within Microsoft Visual Studio. Exact steps may vary depending on the version of Microsoft Visual Studio in use.

Step 1: Build a binary

1. Launch Microsoft Visual Studio
2. Select **File** > **New** > **Project**
3. In the **New Project** window, select a project type under **Fortran**

   (Set Fortran as the language in the **Language** dropdown)
4. Select a template and click **OK**
5. Select **Build** > **Build Solution**

The results of the compilation display in the **Output** window.

Step 2: Set build configurations

1. Right click on **Project** in **Solution Explorer > Properties**
2. Locate **Fortran** in the list and expand the heading
3. Walk through the available properties to select your configuration

NOTE To change the compiler version in Microsoft Visual Studio, navigate to **Tools > Options > Intel Compilers and Libraries> IFX Intel Fortran > Compilers**.

Build a Program From the Command Line

Use the following steps to test your compiler installation and build a program.

1. Use a text editor to create a file called `hello.f90` with the following contents:

```
print *, “Hello, world!”
end
```

2. From a command prompt, compile hello.f90:

```
ifx hello.f90
```

3. Now you have an executable called `hello`, which can be run and will give immediate feedback:

```
hello
```

   Which outputs:

```
Hello, world!
```

Next Steps

- Explore the latest oneAPI Fortran Code Samples.
- Explore the Intel® Fortran Compiler Classic and Intel® Fortran Compiler Developer Guide and Reference.