

Table of Contents

Compilers	1
<u>Intel Compiler</u>	1
<u>GNU Compiler Collection</u>	3
<u>PGI Compilers and Tools</u>	4

Compilers

Intel Compiler

Intel compilers are recommended for building your applications on either Pleiades or Columbia.

On Columbia, a system default version has been loaded automatically. On Pleiades, there is no system default--you must load a specific module. Use the `module avail` command on Pleiades to see what versions are available and load an Intel compiler module before compiling. For example:

```
% module load comp-intel/11.1.072
```

Notice that when a compiler module is loaded, some environment variables, such as **FPATH**, **INCLUDE**, **LD_LIBRARY_PATH**, etc., are set or modified to add the paths to certain commands, include files, or libraries, to your environment. This helps to simplify the way you do your work.

To check what environment variables will be modified for a module, for example:

```
% module show comp-intel/11.1.072
```

Intel Compilers for Columbia and Pleiades

On Columbia and Pleiades, there are Intel compilers for both Fortran and C/C++:

Intel Fortran Compiler: ifort (version 8 and above)

The **ifort** command invokes the Intel Fortran Compiler to preprocess, compile, assemble, and link Fortran programs.

```
% ifort [options] file1 [file2 ...]
```

Read **man ifort** for all available compiler options. To see the compiler options by categories, go to:

```
% ifort -help
```

fileN is a Fortran source (.f .for .ftn .f90 .fpp .F .FOR .F90 .i .i90), assembly (.s .S), object (.o), static library (.a), or other linkable file.

Source Files Suffix Interpretation:

- .f, .for, or .ftn: fixed-form source files
- .f90: free-form F95/F90 source files

- .fpp, .F, .FOR, .FTN, or .FPP: fixed-form source files which must be preprocessed by the fpp preprocessor before being compiled
- .F90: free-form Fortran source files which must be preprocessed by the fpp preprocessor before being compiled

Intel C/C++ compiler: **icc** and **icpc** (version 8 and above)

The Intel C++ Compiler is designed to process C and C++ programs on Intel-architecture-based systems. You can preprocess, compile, assemble, and link these programs.

```
% icc [options] file1 [file2 ...]
% icpc [options] file1 [file2 ...]
```

Read **man icc** for all available compiler options. To see the compiler options by categories, go to:

```
% icc -help
```

The **icpc** command uses the same compiler options as the **icc** command. Invoking the compiler using **icpc** compiles .c, and .i files as C++. Invoking the compiler using **icc** compiles .c and .i files as C. Using **icpc** always links in C++ libraries. Using **icc** only links in C++ libraries if C++ source is provided on the command line.

fileN represents a C/C++ source (.C .c .cc .cp .cpp .cxx .c++ .i), assembly (.s), object (.o), static library (.a), or other linkable file.

GNU Compiler Collection

GCC stands for "GNU Compiler Collection". GCC is an integrated distribution of compilers for several major programming languages. These languages currently include C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada.

The GNU C and C++ compiler (**gcc** and **g++**) and Fortran compiler (**gfortran**) through the Linux OS distribution are available on Pleiades and Columbia. The current version installed (under `/usr/bin`) can be found with the command `gcc -v`, for example:

```
% gcc -v
... gcc version 4.3.4 [...] (SUSE Linux)
```

Newer versions of GNU compilers can be requested and installed as modules. Currently, there are `gcc/4.4.4`, `gcc/4.4.5`, and `gcc/4.7.0` modules, which includes **gcc**, **g++**, and **gfortran**, available on Pleiades.

Read **man gcc** and **man gfortran** for more information.

PGI Compilers and Tools

Summary: PGI compilers and tools are installed as modules under the system directory /nasa on Pleiades. By default, the PGI compilers generate code that is optimized for the compilation host.

PGI Compiling

As an option to using the Intel compilers, you can use the compilers and program development tools from PGI ([The Portland Group, Inc.](#)) for Fortran, C, and C++. Several versions of the PGI compilers and tools are installed under the system directory /nasa on Pleiades as modules. To see what is available, do:

```
pfe20% module avail comp-pgi
comp-pgi/10.6 comp-pgi/11.0 comp-pgi/11.6
comp-pgi/12.3 comp-pgi/12.4 comp-pgi/12.5
```

The following command will load version 12.5:

```
pfe20% module load comp-pgi/12.5
```

Newer versions may be installed under `/u/scicon/tools/modulefiles`. Use the following command to see what is available.

```
pfe20% cd /u/scicon/tools/modulefiles
pfe20% ls -l /u/scicon/tools/modulefiles/pgi*
pgi_11.10
pgi_12.4
pgi_12.6
pgi_12.8
```

Follow the example below to load version 12.8:

```
pfe20% module load /u/scicon/tools/modulefiles/pgi_12.8
```

Several MVAPICH2 modules built with the PGI compilers are also available under `/u/scicon/tools/modulesfiles`; some have MPI CUDA calls enabled and some do not. Both will work with MPI codes with sections compiled to run on the [GPU](#), but the modules with `_cuda_` in their names also allow some MPI calls to use the GPU memory. See [this MVAPICH web page](#) for more information. At the time of publication the two latest modules are:

```
mvapich2_1.8_pgi_12.8
mvapich2_1.8_pgi_12.8_cuda_4.1
```

Future versions will probably be installed under the system modules tree /nasa, but not under the /u/scicon tree.

Using the PGI Compilers

PGI provides various commands for different languages or functions, as shown in this table.

Command	Language or Function
pgfortran	PGI Fortran
pgf77	Fortran 77
pgf90 or pgf95	Fortran 90/95/F2003
pghpf	High Performance Fortran
pgcc	ANSI C99 and K&R (Kernighan and Ritchie) C
pgCC or pgcpp	ANSI C++ with cfront features
pgdbg	Source code debugger (supports OpenMP and MPI)
pgprof	Performance profiler (supports OpenMP and MPI)

NOTE: By default, the PGI compilers generate code that is optimized for the type of processor on which compilation is performed (the compilation host). Be aware that the processors on Pleiades are forward-compatible, but not backward-compatible. Thus a code compiled and optimized on a newer-generation processor, such as Sandy Bridge, will not necessarily execute correctly on previous-generation processors, such as Harpertown, Nehalem-EP, or Westmere. This could be an issue if you compile and optimize your code on the new Pleiades Sandy Bridge front-end nodes (pfe[20-27]) or the Sandy Bridge compute nodes (through a PBS session) and later want to use the same executable to run on the Harpertown, Nehalem-EP, or Westmere nodes.

If you want to build an executable that targets a specific processor type on Pleiades, use the `-tp` flag:

<code>-tp= sandybridge-64</code>	Intel SandyBridge architecture Core processor, 64-bit mode
<code>-tp= nehalem-64</code>	Intel Nehalem architecture Core processor, 64-bit mode (including Nehalem-EP and Westmere)
<code>-tp= penryn-64</code>	Intel Penryn architecture Pentium processor, 64-bit mode (including Harpertown)

TIP: Using the `-tp=penryn-64`, `nehalem-64`, `sandybridge-64` option will generate a single executable where the code is optimized for the Intel Penryn (Harpertown), Nehalem (Nehalem-EP and Westmere), and Sandy Bridge architectures. The choice of which optimized copy to execute is made at run time depending on the machine executing the code.

PGI recommends that for best performance on processors which support SSE instructions (including all Pleiades processor types), use **pgfortran**, even for FORTRAN 77 code, use the **-fastsse** option.

For more information about the PGI compilers, see the **pgfortran**, **pgcc**, **pgCC man pages** or use the command **pgfortran -help**, **pgcc -help**, or **pgCC -help**. Information about the PGI debugger and performance analysis tool can be found in the **pgdbg** and **pgprof man pages**.