LA-UR-17-30373

Title: How to Build MCNP 6.2

Author(s): Bull, Jeffrey S.
MCNP Development Team, XCP Division

Intended for: Report

Issued: 2017-11-13

How to Build MCNP 6.2

MCNP6 Development Team

Los Alamos National Laboratory, PO Box 1663, MS A143, Los Alamos NM 87545

1. Introduction

MCNP®[*] 6.2 can be compiled on Macs, PCs, and most Linux systems. It can also be built for parallel execution using both OpenMP and Messing Passing Interface (MPI) methods. MCNP6 requires Fortran, C, and C++ compilers to build the code. The Fortran compiler must be compliant with the 2003 Fortran Standard [ISO04]. On a Linux type platform, the build system is based upon GNU *make* [STA98], which is widely used to build and execute code [STA02a, STA02b]. GNU *make* may be installed on a system with the name *gmake* or *make*, depending on choices of the system administrator at installation. In this document, use of the name *make* is assumed to refer to GNU *make*. New in MCNP 6.2 is the ability to compile using Microsoft Visual Studio on Windows systems.

With *make*, building MCNP6 for a variety of hardware platforms is made relatively easy for end-users. The end-user simply types a *make* command, optionally specifying the desired target names and configuration features.

For the rest of this document, directory names will be subdirectories of the `MCNP_CODE/MCNP620` directory in the MCNP6 installation directory.

As of the date of this document, these Fortran compilers are known to build MCNP 6.2:

Compiler	Versions[‡]
gfortran	4.8.5 – 6.4.0
Intel	15.0.5 – 18.0
PGI (Linux)	14.10 – 16.10

Versions 7.1 and higher of gfortran do not compile MCNP 6.2

New compiler versions are often released a few times a year, and some of them may not compile MCNP6. Check the MCNP FAQ (https://mcnp.lanl.gov/mcnp_faq.shtml) web page for the latest information on which compilers do not build MCNP6, and if any changes can be made to the source code or configuration files so that MCNP6 can be built with these troublesome compilers.

[*] MCNP® and Monte Carlo N-Particle® are registered trademarks owned by Los Alamos National Security, LLC, manager and operator of Los Alamos National Laboratory. Any third party use of such registered marks should be properly attributed to Los Alamos National Security, LLC, including the use of the ® designation as appropriate. Any questions regarding licensing, proper use, and/or proper attribution of Los Alamos National Security, LLC marks should be directed to trademarks@lanl.gov.

[†] Only a sampling of compiler versions were tested in these ranges.

2. CONFIGURING AND BUILDING MCNP6

Before building MCNP6, check that the compilers to be used are installed and can be executed directly from the command line. This can be checked with the *which* command (e.g., `which ifort`; `which gcc`). Compilers and libraries are referenced in the `Source/config/$(OS).gcf` files, where `$(OS)` is a particular operating system: Linux, Darwin (Mac OS X), or Windows (Cygwin). Rules associated with file name extensions for the compiler used on each platform also appear in these files. Although the `Source/config` directory contains several other `$(OS).gcf` files, they have not been updated in several years, and are unlikely to build MCNP6.

2.1 THE CONFIG MAKE VARIABLE

The `CONFIG` variable, usually included on the *make* command line, controls which compilers are used to build the code and which features are included in the build. Table 2-1 contains a partial list of `CONFIG` options.

Table 2-1. MCNP6 `CONFIG` variable options

CONFIG option	Description
FORTRAN COMPILERS[*]	
`intel`	Use the Intel Fortran compiler (default)
`gfortran`	Use the GNU *gfortran* compiler
C/C++ compilers[*]	
`gcc`	Use the GNU gcc/g++ compiler (default)
Other options (Note: these options must be included to activate these features)	
`debug`	Compile the code for debugging. Optimizations are turned off and Fortran post-processed files are kept in the `Source/TMP_FPP` directory
`ints_8byte`	Default integer sizes are set to 8-bytes.
`jav`	Do not delete the post processed files in the `Source/TMP_FPP` directory.
`mpi`[*]	Build the code that uses Message Passing Interface (MPI) parallelization
`omp`	Build the code with OpenMP parallelization.

[*]Additional compiler and MPI options are also available. See Section 2.4 for more information

CONFIG option	Description
`plot`	Include tally and geometry plotting capability

For all systems, the default `CONFIG` variable is `"intel plot omp"`.

2.2 BUILDING MCNP6 WITH MAKE[*]

From the Source directory, build the executable by issuing the following commands:

```
make realclean
make build
```

The first command will delete all the files created from previous builds, including test results. It will also delete most of the MCNP6 and MCNP6 Utility executable files in the `/bin` directory. The second command will build the default configuration of MCNP6 using to the settings found in the `<OS>.gcf` file for the computer system in use.

To build a configuration other than the default, add the desired options on the make command line using the `CONFIG` make variable:

```
make build CONFIG="new config options"
```

The new MCNP6 executable, named `mcnp6`, will be placed into the `/bin` directory. If compiling the code with the `mpi` option, the new executable is named `mcnp6.mpi`.

The build can be executed in parallel if more than one processor is available. Parallel build is controlled through the *make* variable `GNUJ`. For example, the *make* command "`make build GNUJ=4`" will invoke a four-way build. That is, *make* will attempt to compile four source files simultaneously.

Additional make targets are available in the MCNP6 build system. These are listed in Table 2-2.

Table 2-2. Commonly Used MCNP6 Build System Make Command Targets

Make Target	Function
`build`	Compiles MCNP6. If the CONFIG options are not the same as the contents of the CONFIG_SAVE file, this will also perform a "`realclean`"
`clean`	Delete compiled object and module files
`depends`	Build the mcnp object dependency files

[*]The commonly used *make* targets such as "`clean`" and "`install`" are redefined in the MCNP6 build system.

Make Target	Function
`install`	Perform a realclean Build *mcnp6* Build the mcnp6 utilities Run the REGRSSION test set with both type 1and type 2 data sets. Includes running *makxsf* [BRO06] to create the type 2 data sets
`mcnp_env`	Create the source file `Source/src/mcnp_env.F90` which contains information about the build
`realclean`	Same as clean, but also delete the object dependency files, `Source/src/mcnp_env.F90`, and the `CONFIG_SAVE` file
`test`	Run the Regression test set
`utils`	Build the MCNP6 utilities

2.3 Parallelism Support Software

MCNP6 can be built with either or both of shared-memory and distributed-memory parallelism. OpenMP is used for shared memory parallelism (threading). OpenMP is supported directly by the compilers and is built into the executable during compilation. No additional software is needed. The prebuilt executables in the MCNP6 distribution are compiled for OpenMP threading.

For distributed memory parallelism, MCNP6 exclusively uses the *Message Passing Interface* (MPI) communications protocol.

2.3.1 Installing MPI Software

To build and run MCNP6 with MPI, a properly implemented version of MPI with Fortran bindings must be installed on the system. Many MPI implementations are available, with most common versions used with MCNP6 are OpenMPI, Intel-MPI, and MPICH.

Most systems do not include MPI software by default, requiring the user to install it themselves. For Linux systems, MPI can be installed using the distribution’s package manager. Mac's also have package mangers (e.g., MacPorts (`https://www.macports.org/`) or Homebrew (`https://brew.sh/`) which can be used to install MPI software. There are two supported Windows versions of MPI that can be used with the Intel compiler. Microsoft MPI can be downloaded for free from the Microsoft Developer Network web site (`https://msdn.microsoft.com/`). Intel includes their version of MPI software with the Intel Parallel Studio XE Cluster Edition for Windows.

Most MPI implementations also provide source code which can be used by the user to compile and install the software. If compiling from source, be sure to set the `MPI_ROOT` environment variable to the MPI installation directory. In addition, add the MPI executables (usually in `$MPI_ROOT/bin`) to your `$PATH.`

2.3.2 Building MCNP6 with MPI

To build MCNP6 with MPI, just add the option `mpi` to the `CONFIG` variable. For example.

```
make build CONFIG="plot mpi omp intel"
```

If using Intel-MPI, replace "`mpi`" with "`intelmpi`".

2.4 Building MCNP6 on Specific Platforms

2.4.1 Linux[*]

Default `CONFIG` variable: `"intel plot omp"`

Additional Fortran complier options:

`gfortran`:	compile with the GNU Fortran and C/C++ compilers
`portland`:	compile with the Portland Group Fortran compiler

Additional C/C++ compiler options:

`icc`:	compile with the Intel C/C++

Notes:

1. To compile MCNP6 with plotting support the X11 header files must be installed. On many Linux distributions the X11 libraries may be installed without the header files. The header files will commonly be available through the Linux distribution’s package manager in a "development," "dev," or "devel" package.

2. On many Linux systems, the GNU suite of compilers may not be installed by default or only parts of the suite may be installed (e.g., *gcc* and *g++* but not *gfortran*). The GNU suite of compilers can be installed from source (`https://gcc.gnu.org/`) or by using the Linux distribution’s package manager.

3. For the Portland Group Fortran compiler, the "`omp`" option is disabled.

4. If using the *gfortran* compiler with later versions of OpenMPI (1.7 or higher), segmentation faults may occur if compiled together with OpenMP. In that case, recompile MCNP6 without the `omp` `CONFIG` option.

2.4.2 Mac OsX

Default `CONFIG` variable: `"intel plot omp"`

Additional Fortran complier options:

[*] Including Linux on Windows. **Note:** Linux on Windows is not an OS supported by the MCNP Development Team.

`gfortran`: compile with the GNU Fortran compiler

Additional C/C++ compiler options:

none

Notes:

1. To compile MCNP6 with the plotting capability, X11 libraries must be installed. The XQuartz project (`https://www.xquartz.org`) provides the necessary X11 libraries for Mac OS X.

2. The GNU Fortran compiler is not installed by default. It can be installed via one the Mac OS X package managers, e.g., MacPorts (`https://www.macports.org/`) or Homebrew (`https://brew.sh/`), or by obtaining the sources and compiling it yourself (`https://gcc.gnu.org/`).

2.4.3 Cygwin (Windows)

Default `CONFIG` variable: `"intel plot omp"`

Additional Fortran compiler options:

`gfortran`:	compile with the GNU Fortran and C/C++ compilers
`mingw64`:	compile using the MingGW-64 Fortran and C/C++ compilers

C/C++ compiler options (Intel compiler only):

`icl`:	compile using the Intel C/C++ compiler
`msvc`:	compile using the Microsoft C/C++ compiler (default)

MPI options:

`intelmpi`:	compile using the Intel MPI package (Intel compiler only)
`msmpi`:	compile using the Microsoft MPI package (Intel compiler only)
`openmpi`:	compile using OpenMPI MPI package (*gfortran* compiler only)

Notes:

1. To setup the environment to use the Intel compiler, it is recommended that the user first open the Intel compiler command prompt, and then start Cygwin in the same terminal window.

2. When compiled with the Intel compiler, *mcnp6* needs access to two dynamically linked libraries, (*libiomp5md.dll* and *X11_64.dll*) to be able to run. If these libraries are not in the `$PATH` environment variable, they must be in the same directory as the *mcnp6* executable.[*]

3. The *libiomp5md.dll* library provided with the MCNP6 installation is for the 17.0.1 version of the Intel compiler, and might not be compatible with newer or older versions of the compiler.

[*]If MCNP6 was installed using Cygwin, the `$PATH` variable already includes these libraries.

Be sure that the newly compiled *mcnp6* executable is using the correct version of *libiomp5md.dll*.

4. To compile with either the *gfortran* or the *mingw64* compilers, the Cygwin packages (gcc-fortran for *gfortran*, mingw64-x86_64-gcc-fortran for *mingw64*) must to be installed.

5. There are no additional C/C++ compiler options for gfortran and mingw64.

6. For MPI, *gfortran* must use the OpenMPI package provided by Cygwin. The OpenMPI package is not installed by default.

7. The `mingw64` compiler option will not build with the Cygwin OpenMPI package.

2.4.4 Visual Studio (Windows)

Four configurations are provided for the MCNP6 Visual Studio solution:

Release	same as `make CONFIG="intel omp plot"`
Debug	same as `make CONFIG="debug intel omp plot"`
MS_MPI_Release	compile using Microsoft MPI same as `make CONFIG="intel msmpi omp plot"`
Intel_MPI_Release	compile using Intel MPI same as `make CONFIG="intel intelmpi omp plot"`

Notes:

1. The Visual Studio solution file is located in the `Source\Visual_Studio` folder.

2. Either the Microsoft or Intel C/C++ compilers can be used with Visual_Studio.

3. When compiled within Visual Studio, *mcnp6* needs access to two dynamically linked libraries, (*libiomp5md.dll* and *X11_64.dll*) to be able to run. If these libraries are not in the `%PATH%` environment variable, they must be in the same folder as the *mcnp6* executable. These libraries are only in `%PATH%` if running *mcnp6* using the MCNP6.2 CMD shortcut.

4. The *libiomp5md.dll* library provided with the MCNP6 installation is for the 17.0.1 version of the Intel compiler, and might not be compatible with newer or older versions of the compiler. Be sure that the newly compiled *mcnp6* executable is using the correct version of *libiomp5md.dll*.

2.5 THE SOURCE DIRECTORY

This directory contains all the source code files needed for MCNP6 and the top-level `Makefile` for building the code. The `Source/src` directory contains the main source code for MCNP6. The directories cgm, dedx, hexs, import, incl_abla, lcs, and llnlfiss contain the source code for specific code packages that have been incorporated into MCNP6.

The Makefile in this directory is used to invoke the main functions required for building and testing the code including the cleaning of all directories (removing files from previous builds and testing), building the MCNP6 executable(s), building the MCNP6 utilities in the Utilities folder, and regression testing of the code.

2.6 SOURCE/CONFIG DIRECTORY

This directory contains files needed to setup the build configuration. The files in this directory include the following:

<OS>.gcf	(with <OS>=Linux, Darwin, Windows)
make_depends.pl	
make_setup.mk	
mcnp_env.pl	
VC_info.gcf	

The system-dependent configuration files have names of the form <OS>.gcf. These files contain system-dependent defaults, values of compiler flags, and links to specific files and libraries which are used for certain options such as plotting.

The file make_setup.mk contains Makefile elements common to all operating systems.

The script mcnp_env.pl, together with the VC_info.gcf file, generate a Fortran source code file, Source/src/mcnp_env.F90, which includes information about the build including, but not limited to,

- The date of the build,
- The compiler versions used,
- The configuration options, and
- The code version information.

The script make_depends.pl is used to generate the list of files (FILE_list) and their dependencies (Depends). These files are located in the directory module_lib/, which is created by the build system, if not present. This directory and all of its contents are deleted by the command "make realclean"

2.7 MCNP620/UTILITIES DIRECTORY

The Makefile in this directory generates many of the MCNP6 utilities and places them in the bin directory. These utilities include:

- The event-log analyzer (*ela.pl*)
- FIT_OTF*
- *gridconv**
- *makxsf**
- *mcnp_pstudy*

* At present, the utilities FIT_OTF, *gridconv*, *makxsf*, and *um_convert* can only be built with the Intel Fortran compiler

- *merge_mctal*
- *merge_meshtal*
- *simple_ace.pl*
- the unstructured mesh utilities (*um_convert*[*], *um_post_op*, and *um_pre_op*)

The utilities can be built by entering "`make utils`" from the `MCNP620` directory, or just "`make`" in the `MCNP620/Utilities` directory. If not using the Intel Fortran compiler, add `CONFIG="compiler"` to the make line

2.8 TESTING/REGRESSION DIRECTORY

The `Makefile` in this directory is used to execute the regression test suite and present a summary of the test results in comparison against the standard templates (expected answers) included in the MCNP6 distribution. The list of numbers after each test input name are the length, in bytes, of the difference between the calculation and the appropriate template. The summary table also includes information about parallelism used in the calculations, if used. All templates in the distribution were generated by a sequentially run, default-built executable.

Don't be surprised if the summary table lists a few test differences. Differences are possible if *mcnp6* is compiled or executed with different compilers, operating systems, and/or hardware. The report "MCNP6.2.0 Release Testing" [MCN17] includes testing results from several different computer environments.

An additional option is to run the tests with type-2 data. With this option, the regression tests are run with type-1 data and then MAKXSF is executed to generate type-2 data files. Then the regression tests are repeated with the type-2 data. The summary table indicates when type-2 data results are being shown.

It is important to note that most options can be overridden on the *make* command line. For example, if a user wants to execute the tests with OpenMP threading, say with three threads, the following command will do so:

```
make test1 NTRD=3
```

This assumes that the executable has been built with threading enabled.

Note: *Sequential or OpenMP executables will have the same name,* `mcnp6`*, while MPI executables, with or without OpenMP capability, will be named* `mcnp6.mpi`.

The regression tests can also be executed with MPI distributed memory parallelism. The command form for MPI test execution [with optional threading] is

```
make CONFIG=mpi* NMPI=<n> [NTRD=<m>]
```

where `<n>` is the number of MPI processes (master plus workers) and `<m>` is the number of OpenMP threads *per MPI process*. (Leave off the square and angle brackets). The command-line option, `CONFIG=mpi`, causes *make* to look for an executable named `mcnp6.mpi` and issues appropriate MPI commands for execution, e.g., `mpirun`. This execution form will use (1+(*n*-1)**m*) processes: 1 master MPI task; and (*n*-1) MPI workers—each consisting of (*m*) threads.

[*] If *mcnp6* was compiled with `intelmpi` or `msmpi` configuration options,

If using the Windows terminal, the batch file `RunRegression.bat` is used to run the REGRESSION tests. This script does not support testing mcnp in parallel.

3. Additional Software

3.1 Installing a Linux Emulator for Windows—Cygwin

To use the *make*-based MCNP6 build system, a Linux-like command environment must be available. Cygwin, a freeware port of a Linux command shell for Windows, is one such emulator and provides all necessary and desirable Linux commands.

To install Cygwin, follow the instructions on its website, http://www.cygwin.org. MCNP6 requires the 64-bit installation of Cygwin. In addition to the default Cygwin packages, MCNP6 needs the `perl`, `diffutils,` and `make` packages to use the *make* utility for building and testing. Other packages that can be used by MCNP6 are `gcc-fortran, mingw64-x86-gcc-fortran,` and `openmpi`

3.2 Installing Plotting Software—X-Windows Client

In order to use the MCNP6 plotter, an X11 windows client/server software package is needed. The MCNP6 team does not recommend or endorse any of these products, free or commercial. The listed X-Window servers have been tested with MCNP6.

For Windows, Xming is a popular X-Windows software package. Cygwin also provides an X-Windows package, X11. There are also several commercial X-Windows products available for Windows operating systems

For Mac OS X, the XQuartz project (https://www.xquartz.org/) provides X11 libraries and server support.

For Linux, X11 libraries are standard.

4. Testing

4.1 Test Execution

After building an MCNP6 executable, we strongly recommend that you verify its functionality by running the `REGRESSION/` test suite located in the `Testing/` directory. The `Testing/` directory contains cross-section files and libraries used with the set of tests in the same directory.

The `Source/` and `Testing/` directories are at the same directory level in the MCNP6 distribution. You can run the `REGRESSION/` tests from within the `Source/` directory.

If you wish to run the tests with only type-1 (ASCII) cross-section data, use the command

```
make test1
```

If you wish to run the tests with only type-2 (binary) cross-section data, use the command

```
make test2
```

Although the `REGRESSION/` test suite runs a variety of differently contrived tests, it does not test the plotting utilities. To run a few simple tests of the plotting utilities, work from the `Testing/REGRESSION/` directory:

```
cd ../Testing/REGRESSION
```

4.2 Testing the Plotter

MCNP6 supports three types of plotting. Testing of each is discussed in the following section. The instructions for each type of test assume the following:

1. The `DISPLAY` environment variable has been properly set,
2. An X11 server has been started, and
3. The `REGRESSION/` test suite has been run as described above.
   - This leaves the directory suitably set up for plotter testing.
   - If this is not the case, issue to following commands to set up for the plotter tests:

```
make type1            (to link in cross-section files)
cp Inputs/inp01 .     (to get inp01 into the working directory)
```

4.2.1 Geometry Plotting—PLOT Utility

Run the executable (`mcnp6`) and specify the input file (`i=inp01`), the cross-section file (`xsdir=testdir1`), and options to process input and enable geometric plotting (`ip`) with the command shown below:

```
../../bin/mcnp6 i=inp01 xsdir=testdir1 ip
```

Interact with the plot window that appears according to the instructions in the MCNP6 User's Manual. To end the session, click *End* in the bottom plot window menu. If a plot window appears, the plotter is working.

4.2.2 Tally Plotting—MCPLOT Utility

Run the executable (`mcnp6`) and specify the input file (`i=inp01`), the output file (`o=out01`), the runtpe file (`r=rtpe01`), and the cross-section file (`xsdir=testdir1`) with the command shown below:

```
../../bin/mcnp6 i=inp01 o=out01 r=rtpe01 xsdir=testdir1
```

Run the executable (`mcnp6`) and specify the runtpe file where results of the prior run were collected (`r=rtpe01`), and enable tally plotting (`z`) with the command shown below:

```
../../bin/mcnp6 r=rtpe01 z
```

When the `mcplot>` prompt shows up, specify the tally (`tally 1`) and a tally fluctuation chart (`tfc=m`) with the command shown below:

```
tally 1 tfc=m
```

To end the session, give the command shown below:

```
exit
```

4.2.3 Cross-Section Plotting—MCPLOT Utility

In the example that follows, a simple neutron transport problem is used and the default particle type in use is neutrons (`par=n`).

Run the executable (`mcnp6`) and specify the input file (`i=inp01`), the cross-section file (`xsdir=testdir1`), and the options to process input (`i`), process cross sections (`x`), and enable cross-section plotting (`z`) with the command shown below:

```
../../bin/mcnp6 i=inp01 xsdir=testdir1 ixz
```

When the `mcplot>` prompt shows up, specify appropriate commands and values from the input file inp01 (e.g., material 1 (`xs=m1`) and reaction identifier 1 (`mt=1`)) with the command shown below:

```
xs=m1 mt=1
```

Try another plot at the `mcplot>` prompt (e.g., material 2 (`xs=m2`) and reaction identifier 2 (`mt=2`)) with the command shown below:

```
xs=m2 mt=2
```

To end the session, give the command as shown below:

```
exit
```

5. MODIFYING THE SOURCE CODE

To make local changes to a copy of MCNP6, simply edit the appropriate source files. A partial directory source code structure for MCNP6 follows:

```
Source/
    Makefile          (the controlling Makefile)
    Visual_Studio/    (Microsoft Visual Studio files)
    X11R6/            (X11 header and library files)
    config/           (build configuration files)
    cgm/
    dedx/
    hexs/
    import/
    incl_abla/
    lcs/
    llnlfiss/
    Makefile
    regl/
    src/
```

Except for `config/`, `Visual_Studio/`, and `X11R6/`, these directories contain source files used to build MCNP6. All of the Fortran source files are in the free-form (F90) style. The `cgm/`, and `llnlfiss/` directories contain mostly C++ source files.

If you need to make changes to existing files, edit them and leave them in their current locations.

Note: *Before making any changes to the MCNP6 distribution, backup the original files for recovery and for comparison. It is necessary to have an unmodified copy of the entire distribution to generate and/or apply patches.*

If you add new files, they will be automatically included in builds if you follow these steps:

1. Create the new file conforming to the Fortran free-style format and place it in the `Source/src/` directory. The name of new Fortran files should be of the form `<name>.F90`. The uppercase F90 suffix indicates the file should be preprocessed before (or during) compilation.
2. In `Source/`, type '`make realclean`' to prepare for a full rebuild. This cleans the entire distribution tree.

Note: *All source code files* (`.F90`, `.f90`, `.F`, `.f`, `.c`, `.cpp`, *etc.*) *are found and included in the build. So do not leave files in the listed source directories you do not need, such as copies of unmodified files.*

When your changes are ready, change into the `Source/` directory and type "`make build`". Added files (or modifications that affect dependencies) will be automatically found and included. Be sure to clean the directory before the first build to ensure your files are included. You can include additional build options, as needed.

6. References

BRO06 F.B. Brown, "The makxsf Code with Doppler Broadening", LA-UR-06-7002 (2006). Included in the MCNP Reference Collection (https://laws.lanl.gov/vhosts/mcnp.lanl.gov/references.shtml).

MCN17 MCNP Team,"MCNP6.2.0 Release Testing", LA-UR-17-25000 (2017). Included in the MCNP Reference Collection.

STA98 R. M. Stallman and R. McGrath, *GNU MAKE*, Free Software Foundation, Boston, MA (JULY 1998). See `http://www.gnu.org`.

STA02a Richard Stallman, Paul Visscher, Bret Smith, and Luis M. Arteaga, "GNU's not Unix!" October 22, 2002. See `http://www.gnu.org/software`.

STA02b Richard Stallman, Bob Chassell, Melissa Weishaus, Roland McGrath, and Paul D. Smith, *GNU make*, Version 3.80 (stable), Free Software Foundation, Inc., Boston, MA (October 2002).

IOS04 "Information technology - Programming languages - Fortran - Part 1: Base Language", International Organization for Standardization, ISO/IEC FCD 1539-1:2004(E), (November, 2004)