Tools and Compilers at CIRT/AHPCC

CS 471/Math 471, Fall 1997

Brian T. Smith and Andrew C. Pineda

Introduction

This is a short compilation of basic information, derived from various sources, on some of the programming tools that you will be using in this course. It is intended as a supplement to the materials found in the course text.

Text editors

There are two text editors that you will find available on all of the machines that you will be working on here at UNM. They are Vi and GNU Emacs. Vi is a compact, powerful, display oriented text editor that comes standard with every UNIX implementation, and GNU Emacs is a freely available and extremely powerful text editor from the Free Software Foundation. For those of you who may want to try them on your PC's at home, freely available DOS and Windows 95/NT implementations of each exist.

Vi is a modal editor which means that its responses to keystrokes depends upon what mode it is in. There are three modes the editor can be in; (1) command mode, (2) insert mode, and (3) line editor (ex) mode. When you start the editor, it starts in command mode. In command mode, you can move around the file with motion commands, delete, move text, etc., or enter insert or line editor modes. Insert mode is entered when a command that inserts text is given. (Among these are "i" for insert before the cursor, "a" for insert after the cursor, etc.) You return to command mode by pressing the ESC key. (Pressing the ESC key until a bell sounds is a sure way to find out that you are in command mode.) Line editor or ex mode is entered when you type a ":" in command mode. In this mode, you can perform more complex editing commands, save the file, or exit the editor using the commands of the Unix ex editor. Enclosed is a copy of the chapter on editing with Vi from Christopher C. Taylor's somewhat humorous "Unix is a Four Letter Word... and Vi is a Two Letter Abbreviation" which provides a more complete description of editing with Vi than the course text. Several free Windows 95/NT implementations can be found at http://www.itribe.net/virtunix/otherstuff.html.

GNU Emacs is a highly customizable editor built around a Lisp interpreter. It has several advantages over Vi when it comes to programming. First, when editing programming languages, it can be customized to automatically indent text in a manner appropriate to the language. (It does this typically by examining the name of the file you are editing.) Second, the screen can be divided into windows, and different files, different views of the same file, or even relevant Unix manual pages presented in other windows. Third, it can be configured to automatically save numbered backup copies of your work in case you need to go back to a previous version. It is modeless when it starts up, so that typing normal characters inserts them in the current buffer. Commands are performed by typing various sequences of control characters and/or escape sequences. Enclosed is a copy of the reference card for GNU Emacs version 19 which covers the basic editing commands. In the reference card

command keystrokes are abbreviated as C-n for Control-n and M-x for either ESC-x or Alt-x, etc. The online tutorial is available by typing C-h t (Control-h t). Full documentation on the editor is available from the internal documentation system called Info. It is accessed by typing C-h i. The Windows95/NT version of Emacs is available at
http://www.cs.washington.edu/homes/voelker/ntemacs.html. The Extended DOS/Windows 3.1 version of Emacs can be found at
ftp://ftp.coast.net/pub/Coast/disk2/msdos/editor/em19*.zip.

CIRT/AHPCC Compilers (Language Processors)

Each programming language (e.g. C or Fortran) has a compiler, loader/linker, and system procedure libraries (e.g. input/output libraries, intrinsic procedures, scientific subroutines) associated with it; this collection of 'tools' is sometimes referred to as the processor. The processor converts your program into an executable file, which when giving an optional input file, executes the instructions in your program and produces some output. Unfortunately, the commands to invoke these tools, the location of the libraries and the options associated with each tool are different (even the same ones are sometimes spelled differently) and often have different defaults when the options are not specified.

This document introduces you to the various tools for Fortran 90 on the CIRT and AHPCC systems. Two vendor products are briefly described, namely the IBM compiler(s) and the Numerical Algorithms Group portable Fortran 90 compiler (available on most workstations).

The common command structure for UNIX systems is as follows:

```
<tool_name> <options> <libraries> <source_file>
```

where the items surrounded by angle brackets (including the brackets) are replaced by specific characters for each compiler. For example, to compile and create an executable file from the traditional 'Hello world' program written in Fortran's free source form, the simplest command using the IBM compiler is:

```
xlf90 hello.f
```

and using the NAG compiler, the command is:

```
f90 hello.f90
```

Both programs create an executable file, named a.out, by default. To execute this program, type

```
a.out
```

and it produces the line 'Hello world, from Fortran 90' written to the standard output (the terminal screen, by default). For the above example, the source program (that is, the contents of the files hello.f or hello.f90) are:

```
program  hello
   print *, 'Hello world, from Fortran 90'
end program hello
```

You may have noticed that the source file name is different for each of these compilers. This is because each vendor has used a different default naming convention for files whose program source is written in free source form. Both compilers have options that specify that the file qualifier for free source form and specify that the given file is in a particular source form (free or fixed). For example, the command

```
f90 -free hello.f
```

compiles hello.f assuming the source text follows the rules of the free source form and the command

```
xlf90 -qfree=f90 -Fmyxlf.cfg hello.f90
```

compiles the same program with a f90 suffix, using the configuration file myxlf.cfg in which the
option fsuffix=f90 is specified for the xlf90 command.

Many other options are available for each of these compilers. You should consult the man
pages and/or the appropriate user guides. The man page for xlf90 can be viewed using the command

```
man xlf
```

and for the Nag compiler,

```
man f90
```

Note: please edit your .login file in your home directory on CIRT machines and add the line

```
setenv MANPATH $MANPATH\:/nfs/a/acpineda/CS471/man
```

Useful Compiler Options

There are several options that you will be find very useful throughout the lab periods. These include for the NAG compiler -C, -c, -o, and possibly -P. For the xlf compiler, -O, -O3, -c, -g, -gsigtrap and -C. You will find these used in Makefiles in the lab exercises. These options have the following meanings:

-C Insert code in the executables that detects subscript and substring range violations. The NAG compiler will point to the statement by line number where the violation occurred. For the xlf compiler, you need in addition the options -g and -qsigtrap.

-O Optimize the executable code. The NAG compiler does no Fortran optimization but passes the -O option to its code generator which is the cc compiler and thus it optimizes the generated C object code. The option -O3 for the xlf compiler does memory-intensive compile-time intensive optimizations, including in-lining and loop-unrolling that can sometimes greatly improve the run-time efficiency of the executable program.

-c Only compile the code to an object file whose name by default is prog.o when the program in the file prog.f (or prog.f90 is being compiled.

-g Compile the program so that profilers and/or debuggers can be used on the executable code. See various reference guides on xlf for the -p option for profiling code.

-P Only for the NAG compiler, it creates run-time code that checks for uninitialized (and out-of-range) pointers.

-o<name > Specify the name <name> for the executable file if -c is not used or specify the name of the object file if -c option is used. <name> is any UNIX name; for object files, it is a UNIX convention to use the .o name qualifier, e.g. hello.o, and is a good convention to follow.

These are by no means all of the useful compiler options. Others specify details of optimization, provide support for the IEEE standard, support levels of diagnostics and error checking, versions of Fortran (back to Fortran 77, the MIL-STD, XL Fortran extensions), source forms, compatibility with calling C programs, multi-byte characters, and so on.

The Linker

The linker is a separate phase of the xlf or f90 command (actually is the UNIX command ld) which allows you to specify link options such as the libraries to be used. Its main purpose is to link together object files created by the -c option with any user-created or other system libraries. Its form is:

```
xlf90 <options> <object_files>
```

or

```
f90 <options> <object_files>
```

where <object_files> is a list of object files (usually .o) files separated by blanks; the options are linker/loader options. (Actually, the list can contain .f or .f90 files as well; the xlf90 or f90 commands will start up the compiler, processing such named files.) Again, there are many options available, but the most useful ones are the -L and -l options; the -L<path> option specifies a directory where additional library archives (such as libc.a or liblinpack.a) can be found; the -l<name> specifies the particular archive to be searched to resolve external references (such as external functions and subroutines) where the archive file name is lib<name>.a. For example,

```
xlf90 -lc main.o sub1.o sub2.o
```

links the three specified object files, resolving unresolved external procedures from the archive library libc.a which is in the default directory /usr/lib

Machines On Which To Run These Compilers

On both the CIRT and AHPCC systems, the IBM xlf compiler will run only on RS/6000 based machines. At CIRT, these include all the machines accessed by the address aix.unm.edu when using telnet rsh, or rlogin to access the CIRT machines. At the CIRT facility, these include the following machines: argo, mirac, pegasus, and musca. At the AHPCC, these include: acoma, gallup, gila, taos, raton, and grants. At CIRT, the NAG f90 compiler is on all AIX machines and the SGI machines altair, neptune, europa, venus, pluto, castor, saiph, rigel, and spica.

As for the man pages, the IBM xlf man pages are accessible with your default path and are on the machines that have the compiler. The man pages for the NAG f90 compiler are not available directly; that is, it is not in the default manpath; either you modify the manpath as suggested above or access the man document as follows: on CIRT machines, type

```
man -M/nsf/user/a/acpineda/CS471/man f90
```

and on AHPCC machines, type

```
man -M/home/bsmith/man f90
```

References

Fortran 90 Handbook, Adams et al, McGraw-Hill, 1994

Fortran 95 Handbook, Adams et al, Springer-Verlag, 1997 (soon to be printed)

Fortran Top 90, Adams et al, Unicomp , Inc., 1994

Programming Guide to Fortran 95, Adams et al, Unicomp, Inc., 1996

Fortran 90 Programming, Ellis et al, Addison-Wesley, 1994

Optimization and Tuning Guide for Fortran, C, and C++, IBM, AIX Version 3.2 for RISC System/6000, SC09-1705-01

XL Fortran for AIX, Language Reference, Version 3 Release 2, SC09-1611-01

XL Fortran for AIX, User's Guide, Version 3 Release 2, SC091610-01