

FORTRAN

SYNOPSIS

This document describes the various implementations of the FORTRAN Language running under various implementations of the UNIX operating system. Topics include creation, compilation, and execution of a FORTRAN program. Also covered in a separate section is a discussion of user-created FORTRAN object libraries. It is assumed that the reader has a working knowledge of FORTRAN and a UNIX text editor, such as vi.

FORTRAN LANGUAGE INTRODUCTION There are many implementations of the FORTRAN language available on the various UNIX systems at Cal Poly. In this document we will be addressing the compilers which fall into the "standard" FORTRAN category.

The compilers involved boil down to the system supplied compiler "*f77*" (and some of its variants). The following table describes the platform, the name of the compiler, and any special considerations.

Hardware Platform	Command Name	Compiler Vendor
RISC/6000 AIX 3	<i>f77</i> , <i>xlf</i> , or <i>xlf90</i>	IBM XLF ANSI FORTRAN Compiler
SunOS 4.1.3	<i>f77</i>	SunOS FORTRAN Compiler

NOTE: Where multiple commands exist for a compiler, consult the man pages for that command to determine which options are defaulted with each command variation.

NAMING CONVENTIONS FORTRAN source programs must follow standard UNIX naming conventions. The FORTRAN source file must end with ".*f*" on the RISC Systems, ".*f*" or ".*for*" on the SunOS *f77*.

CREATE A SOURCE FILE Use any available text editor, such as vi, to create your source file. (See the Information Systems User Guide "vi Editor" which is included in the "Introduction to UNIX" User Guide bundle from El Corral Bookstore.) You may also bring the source in from another system using such facilities as Kermit, FTP, or xmodem.

FORTRAN FILE SPECIFICATIONS Depending on the system which you are programming for, FORTRAN is linked to its files in various ways.

When using IBM AIX XL FORTRAN, the following units are preconnected:

Unit 0 is connected to standard error when the program executes the first input/output statement on unit 0.

Unit 5 is connected to standard input when the program executes the first input/output statement on unit 5.

Unit 6 is connected to standard output when the program executes the first input/output statement on unit 6.

All other units are preconnected when run time begins. Unit *n* is connected to a file named "fort. *n*". These files need not exist, and they are not created unless you use their units. You may specify your own file name instead of using the "fort. *n*" file. To do so you may either specify your file for that unit through an **open** statement or you can create a symbolic link prior to executing the application. For example:

```
% ln -s myfile fort. 10<CR>
```

will create a symbolic link between "*myfile*" and "*fort. 10*". The symbolic link may be removed after execution has finish by entering

```
% rm fort. 10<CR>
```

which will not affect the file "*myfile*".

Unless otherwise specified, a unit is connected for sequential formatted input/output.

COMPILE THE FORTRAN SOURCE PROGRAM

Most of the FORTRAN compilers have many flags in common as well as the syntax of the command. In the following example, replace the word "*f77_cmd*" with the name of the compiler from the table on page 1. The general syntax of FORTRAN compilers is

```
% f77_cmd [-c] [-C] [-Dname[=value]] [-E] [-g] [-Ipathname] [-lkey]
          [-Lpathname] [-o executable_name] [-O] [-p] [-pg] [-P] [-Uname]
          [-w] sourcefile [sourcefile... sourcefile]<CR>
```

where the options are defined as

-c	Compile the source only, do not call the linker (<i>ld</i>).
-C	Write comments to output when doing preprocessing (used with -E and -P).
-Dname[=value]	Define <i>name</i> as in a #define directive. If =value is not specified, 1 is assumed.
-E	Preprocess, but do not compile; output to stdout (standard output).
-g	Produce debug information for use with debuggers such as " <i>dbx</i> ".
-Ipathname	Search the directory specified by " <i>pathname</i> " for include files which do not start with an absolute path.
-lkey	Selects the library <i>libkey</i> . a to be searched for unresolved references by <i>ld</i> .
-Lpathname	Search the directory specified by " <i>pathname</i> " for the libraries specified by -lkey.
-o name	Name the executable file <i>name</i> instead of a. out.
-O	Optimize code generation.
-p	Generate simple profiling support code.
-pg	Generate profiling support code more extensive than provided by -p.

- P Preprocess, but do not compile. Output to "*sourcefile.i*".
 - U*name* Undefine name as in #undef directive.
 - w Suppress informational, language-level, and warning messages.
- sourcefile [sourcefile ... sourcefile]*
One or more source filenames.

Additional parameters may be found by viewing the man page for the compiler on its system by entering

```
% man f77_cmd<CR>
```

where "*f77_cmd*" is the name of the compiler command you wish to use from the table on page 1.

NOTE: Throughout the rest of this document, "*f77_cmd*" is used to refer to the compiler command. Always replace that with the compiler you wish to use from those listed in the table on page 1.

EXECUTE A FORTRAN EXECUTABLE

To execute your compiled program, you simply type the name of the object module which was produced from the compile step. The default for both compilers is "a. out". Each of the different FORTRAN compilers does have the capability of producing an object module with a different name, but for the purposes of this example, we will assume that the object module name is "a. out". To execute it you enter

```
% a.out<CR>
```

This will cause the program to be loaded and begin execution.

PRINTING FILES WITH CARRIAGE CONTROL IN COLUMN 1

Using FORTRAN programs which were written for other operating systems can result in an output file which contains an old style carriage control in column one of the file. On the Sun and RISC systems, there is a utility named "*fpr*" which can take these output files and convert them to a format which meets UNIX line-printer conventions. The "*fpr*" command works as a filter in that it accepts input from standard input and places its output on standard output.

You may use "*fpr*" combined with your program and the "*lpr*" command to take output from your program and send it to the system line-printer as follows:

```
% a.out | fpr | lpr [options]<CR>
```

where "*a.out*" is the name of the program being executed and "*options*" are options that may be required for "*lpr*" on the system you are using.

You may also use "*fpr*" on a pre-existing file which was created by a program which generates FORTRAN type carriage control as follows:

```
% fpr <myoutputfile | lpr [options]<CR>
```

where "*myoutputfile*" is the name of the file containing the carriage control characters and "*options*" are options that may be required for "*lpr*" on the system you are using.

COMPILING IN MULTIPLE STEPS

Let us assume in this example we have a program that is composed of three different source files. These source files ("myprog1. f", "myprog2. f", and "myprog3. f") are all located in the current working directory. To compile them all, then produce a single object file, would be accomplished by executing the *f77_cmd* command. This is done by typing

```
% f77_cmd myprog1. f myprog2. f myprog3. f <CR>
```

This can also be accomplished in multiple steps by entering

```
% f77_cmd -c myprog1. f <CR>
% f77_cmd -c myprog2. f <CR>
% f77_cmd myprog3. f myprog1. o myprog2. o <CR>
```

This compiles the three separate source files. They are then linked to the standard FORTRAN library and produce an object module named "a. out".

To compile and link the routine "myprog. f" to additional system subroutine library files "libcurs. a" and "libcurses. a", you would type

```
% f77_cmd myprog. f -lcurs -lcurses <CR>
```

Note that only the portion "curs" and "curses" are used from the library file name. When this is done as part of the *f77_cmd* command, the additional subroutine libraries must be indicated as the last items on the command line.

USING FORTRAN WITH IMSL

FORTRAN may be used with the IMSL Math, Stat, SFUN libraries on some systems on campus. The following section describe how to perform this function on the IBM RISC systems supported by Information Technology Services and Information Systems.

IMSL on the AIX RISC System runs under a Software License Manager. In order to run a program that includes IMSL routines or run any of the IMSL support utilities, you must first set an environmental variable as shown below.

Bourne Shell Commands

```
$ LM_LICENSE_FILE=/usr/local/lib/imsl/bin/license.dat
$ export MYDATA
```

C Shell Command

```
% setenv LM_LICENSE_FILE /usr/local/lib/imsl/bin/license.dat
```

A FORTRAN program may be compiled and linked to routines contained in the IMSL subroutine library on AIX RISC Systems by entering

```
% xlf myprog. f -limsl <CR>
```

where "myprog. f" is the name of your source file containing the FORTRAN program. If the program compiles and links successfully, enter

```
% a. out <CR>
```

to execute it. This last command may be used as many times as are necessary without recompiling as long as no changes are made to your source code.

NOTE: Only one user may execute a program that uses the IMSL routines within the cluster at one time. If another user is using the subroutine library, your program will pause, then continue when they are done. If the program doesn't seem to run, you may wish to check the status of the License Manager by entering

`% l mstat<CR>`

This will display the status of the License Manager Server. If there appears to be a problem, contact the Help Desk at 756-7000 during normal working hours, or Computer Operations at 756-5512 after hours to report the problem.

For more information on using IMSL on the AIX RISC Systems, enter

`% i msl. i df<CR>`

which will take you into the IMSL Interactive Documentation Facility. For more information on using "`i msl. i df`" enter the command

`% man i msl. i df<CR>`

HOW TO USE THE INTERACTIVE DEBUGGER

There is an interactive debugger available for each of the FORTRAN compilers on most of the campus UNIX systems. Please refer to the section entitled "Debugging Tools" for more information on using these interactive debuggers.

COMMAND SUMMARY

`% f77_cmd filename<CR>`

Compile the FORTRAN source file specified by *filename* and place the object module in "a. out".

Compile the routines defined in "myprog1. f", "myprog2. f", and "myprog3. f". Once they are all compiled, link them into the object module "a. out".

`% f77_cmd -c myprog1. f<CR>`

`% f77_cmd -c myprog2. f<CR>`

`% f77_cmd -c myprog3. f myprog1. o myprog2. o<CR>`

`% dbx a. out<CR>`

Run a program under the interactive debugger on the current site where the *a. out* file is from the FORTRAN compiler.

`% f77_cmd myprog. f -l subs<CR>`

Compile the routine defined in "myprog. f" and link it to the additional library "libsubs. a".

DOCUMENT CODE: UNIX-41001E

DATE REVISED: September 7, 1995

NOTES