Introduction

This manual provides information on implementing Fortran 77 programs using IRIX™ and the IRIS®−4D ™ series workstation. This implementation of Fortran 77 contains full American National Standard (ANSI) Programming Language Institute Fortran (X3.9-1978). Extensions provide full VMS Fortran compatibility to the extent possible without the VMS operating system or VAX data representation. This implementation of Fortran 77 also contains extensions that provide partial compatibility with programs written in SVS Fortran and Fortran 66.

Fortran 77 is referred to as "Fortran" throughout this manual except where distinctions between Fortran 77 and Fortran 66 are being specifically discussed.

Corequisite Publications

Refer to the *Fortran 77 Language Reference Manual* for a description of the Fortran language as implemented by the Silicon Graphics® IRIS−4D series workstation.

Refer to the *IRIS−4D Series Compiler Guide* for information on the following topics:

- an overview of the compiler system
- improving program performance by using the profiling and optimization facilities of the compiler system
- the dump utilities, archiver, and other tools used to maintain Fortran programs

Refer to the *dbx Reference Manual* for a detailed description of the debugger. For information on interfaces to programs written in assembly language, refer to the *Assembly Language Programmer's Guide.*

Organization of Information

This manual contains the following chapters and appendix:

- Chapter 1, "Compiling, Linking, and Running Programs," gives an overview of components of the compiler system, and describes how to compile, link edit, and execute a Fortran program. It also describes special considerations for programs running on IRIX systems, such as file format and error handling.
- Chapter 2, "Storage Mapping," describes how the Fortran compiler implements size and value ranges for various data types and how they are mapped to storage. It also describes how to access misaligned data.
- Chapter 3, "Fortran Program Interfaces," provides reference and guide information on writing programs in Fortran, C, and Pascal that can communicate with each other. It also describes the process of generating wrappers for C routines called by Fortran.
- Chapter 4, "System Functions and Subroutines," describes functions and subroutines that can be used with a program to communicate with the IRIX operating system.

- Chapter 5, "Fortran Enhancements for Multiprocessors," describes programming directives for running Fortran programs in a multiprocessor mode.
- Chapter 6, "Compiling and Debugging Parallel Fortran," describes and illustrates compilation and debugging techniques for running Fortran programs in a multiprocessor mode.
- Appendix A, "Run−Time Error Messages,"lists the error messages that can be generated during program execution.

Typographical Conventions

The following conventions and symbols are used in the text to describe the form of Fortran statements:

Bold	Indicates literal command line options, filenames, keywords, function/subroutine names, pathnames, and directory names.
Italics	Represents user−defined values. Replace the item in italics with a legal value. Italics are also used for command names, manual page names, and manual titles.
`Courier`	Indicates command syntax, program listings, computer output, and error messages.
`Courier bold`	Indicates user input.
[ ]	Enclose optional command arguments.
()	Surround arguments or are empty if the function has no arguments following function/subroutine names. Surround manual page section in which the command is described following IRIX commands.
|	Sseparates two or more optional items.
...	Indicates that the preceding optional items can appear more than once in succession.
#	IRIX shell prompt for the superuser.
%	IRIX shell prompt for users other than superuser.

Here are two examples illustrating the syntax conventions.

`DIMENSION` *a*`(`*d*`) [,`*a*`(`*d*`)] …`

indicates that the Fortran keyword **DIMENSION** must be written as shown, that the user−defined entity*a*(*d*) is required, and that one or more of *a*(*d*) can be optionally specified. Note that the pair of parentheses ( ) enclosing *d* is required.

`{STATIC | AUTOMATIC}` *v* `[,`*v*`] …`

indicates that either the **STATIC** or **AUTOMATIC** keyword must be written as shown, that the user−defined entity*v* is required, and that one or more of *v* items can be optionally specified.