Introduction

This guide discusses a variety of issues and tools involved in programming under the IRIX™ operating system. It describes the components of the IRIS®-4D™ compiler system, and other programming tools and interfaces.

Audience

This guide is intended for anyone who wants to program effectively under IRIX. The material is intended as an introduction to the topics covered; more advanced features are discussed in more detail in the relevant manual pages. However, this guide is not an introduction to programming; there are a wide variety of good texts available for anyone wanting to learn C, for instance, and we do not cover such topics here. Most of our examples are written in C; we assume a basic familiarity with that language.

Organization

This guide contains the following chapters:

- Chapter 1, "Using the Compiler System," describes the components of the IRIS-4D Series compiler system and explains how to use them.
- Chapter 2, "Dynamic Shared Objects," explains how to build and use dynamic shared objects, which replace the static shared libraries used by previous versions of IRIX.
- Chapter 3, "Improving Program Performance," explains how to reduce program execution time using profiling and optimization tools.
- Chapter 4, "lex," describes how to use the *lex* program generator to develop a lexical analyzer.
- Chapter 5, "yacc," describes the *yacc* program generator, a tool that simplifies the task of developing parsers.

- Chapter 6, “Inter-Process Communication,” describes System V and IRIX inter-process communication mechanisms.
- Chapter 7, “File and Record Locking,” describes how to lock and unlock files and parts of files from within a program.
- Chapter 8, “Using Real-Time Programming Features,” describes features available for real-time programming.
- Chapter 9, “Working with Fonts,” discusses typography and font use on Silicon Graphics computers, and describes the Font Manager library,
- Chapter 10, “Internationalization,” explains how to create an application that can be adapted for use in different countries.
- Chapter 11, “Localization and Creating New Locales,” describes the process of creating a databases of country-specific information for use with internationalized programs.
- Appendix A, “Position-Independent Coding in Assembly Language,” describes assembly language coding techniques required by this version of IRIX.
- Appendix B, “ISO 3166 Country Names and Abbreviations,” lists country codes for use with internationalization and localization.
- Appendix C, “Changing Default Xsgi Settings,” explains how to direct the X Window System server to use alternate keyboard mappings and fonts.
- Appendix D, “Internationalized Commands,” lists the IRIX commands which have been internationalized.

Further Reading

The following references provide more information on internationalization:

- Nye, Adrian: *Xlib Programming Manual for Version 11 of the X Window System, Volume One*, third edition (covers Release 4 and Release 5), O’Reilly & Associates, ISBN 1-56592-002-3
- Scheifler, Robert and Gettys, Jim: *X Window System, Third Edition*, Digital Press, ISBN 1-55558-088-2
- X/Open Company, Ltd.: *X/Open Portability Guide*, Prentice Hall, ISBN 0-13-685819-8 (Set of 7 Volumes)

Conventions

This guide uses the following conventions and symbols:

`Courier`	In text the Courier font represents function names, file names, and keywords. It is also used for command syntax, output, and program listings.
bold	Boldface is used along with Courier font to distinguish user input from system output.
italics	Words in italics represent characters or numerical values that you define. Replace the abbreviation with the defined value. Also, italics are used for manual page names and commands. The section number, in parentheses, follows the name.
[ ]	Brackets enclose optional items.
{ }	Braces enclose two or more items of which you must specify at least one.
|	The OR symbol separates two or more optional items.
…	A horizontal ellipsis in a syntax statement indicates that the preceding optional items can appear more than once in succession.
( )	Parentheses enclose entities and must be written as shown.

The following two examples illustrate 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 specified. The parentheses ( ) enclosing *d* are 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 *v* items can be specified.