About This Guide

This guide discusses a variety of issues and tools involved in programming under the IRIX™ operating system. It describes the components of MIPSpro™ compiler system, other programming tools and interfaces, and dynamic shared objects. It also explains ways to improve program performance.

The compiler system produces either 64-bit object code, *new* 32-bit (n32) object code, or old 32-bit object code. This guide describes the MIPSpro compilers that produce n32-bit and 64-bit objects. For additional information about n32, see the *MIPSpro N32 ABI Handbook*. For information about compilers that produce old 32-bit objects, refer to the *MIPS Compiling and Performance Tuning Guide*.

What This Guide Contains

This guide contains the following chapters:

- Chapter 1, "About the MIPSpro Compiler System," provides an overview of the MIPSpro compiler system.
- Chapter 2, "Using the MIPSpro Compiler System," describes the components and related tools of the MIPSpro compiler system and explains how to use them.
- Chapter 3, "Using Dynamic Shared Objects," explains how to build and use dynamic shared objects.
- Chapter 4, "Using the Performance Tools," describes how to use the *prof*, *pixie*, and *cord* commands.
- Chapter 5, "Optimizing Program Performance," explains how to reduce program execution time by using optimization options and techniques.
- Chapter 6, "Coding for 64-Bit Programs" describes how to write or update code that is portable to 64-bit systems.
- Chapter 7, "Porting Code to N32 and 64-Bit Silicon Graphics Systems" explains how to port code from the old 32-bit mode to the new 32-nit mode (n32).

For an overview of the IRIX programming environment and tools available for application programming, see *Programming on Silicon Graphics Computer Systems: An Overview*.

What You Should Know Before Reading This Guide

This guide is for anyone who wants to program effectively using the MIPSpro compilers. We assume you are familiar with the IRIX (or UNIX®) operating system and a programming language such as C or Fortran. This guide does not explain how to write and compile programs.

This guide does not cover the differences between n32-bit, 64-bit, and 32-bit compilation modes. Refer to *MIPSpro Application Porting and Transition Guide* and *MIPSpro N32 ABI Handbook* for information about the differences between these modes, language implementation differences, source code porting, compilation issues, and run-time execution.

Suggestions for Further Reading

This guide is part of the IRIS Developer Option (IDO), which provides the software and documentation that you can use to write applications for Silicon Graphics platforms. A few IDO online and printed manuals that may be of interest to you are listed in Table intro-1.

Table Intro-1 IDO Topics and Manuals

Topic	Document
IRIX programming	Programming on Silicon Graphics Systems: An Overview Topics in IRIX Programming
Debugging a program	dbx User's Guide
Assembly language	MIPSpro Assembly Language Programmer's Guide
C language	C Language Reference Manual
C++ language	C++ Programming Guide

Table Intro-1 IDO Topics and Manuals

Topic	Document
Fortran language	Fortran77 Programmer's Guide Fortran90 Programmer's Guide
Pascal language	Pascal Programming Guide
Real-time programming	REACT Real-Time Programmer's Guide

Silicon Graphics also provides manuals online. To read an online manual after installing it, type **`insight`** or double-click the InSight icon. It's easy to print sections and chapters of the online manuals from InSight. You can also order printed manuals from Silicon Graphics by calling SGI Direct at 1-800-800-7441. Outside the U.S. and Canada, contact your local sales office or distributor.

Silicon Graphics offers software options to assist in your software development. The compiler options include languages such as *Fortran77*, *Power Fortran*, and *C++*. *CASEVision/Workshop* provides the WorkShop toolset: the Debugger, Static Analyzer, Performance Analyzer, Tester, and Build Manager.

As a developer, you are eligible to become a member of the Silicon Graphics Developer Program at SGI. Call 1-800 -770-3033 for details. If you are developing a MIPS ABI-compliant application, you may want to consult the *mips ABI Frequently Asked Questions*.

You may also want to learn more about standard UNIX and ANSI C topics. For this information, consult a computer bookstore or manuals such as:

- AT&T. *UNIX System V Release 4 Programmer's Guide: ANSI C and Programming Support Tools*. Englewood Cliffs, NJ: Prentice Hall, 1990.
- Levine, Mason, and Brown. *lex & yacc*. Sebastopol. CA: O'Reilly & Associates, Inc., 1992.
- Oram and Talbott. *Managing Projects with make*. Sebastopol. CA: O'Reilly & Associates, Inc., 1991.
- American National Standards Institute, Inc. *American National Standard, Programming Language—C, ANSI C Standard*. ANSI X3.159-1989.
- International Standard ISO/IEC. *Programming languages—C, 9899*. 1990(E).

Conventions Used in This Guide

This guide uses these conventions and symbols:

Links	Links to other sections and chapters in this guide appear in blue. Links to applications, files, and reference pages appear in red.
`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 represent user input.
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 titles, manual page names, and commands. The section number, in parentheses, follows the name.
_ _	A double underline appears as __ in text.
[ ]	Brackets enclose optional items.
{ }	Braces enclose two or more items; you must specify at least one of the items.
|	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 typed.

The following two examples illustrate the syntax conventions:

```
DIMENSION a(d) [,a(d)] …
```

indicates that the Fortran keyword DIMENSION must be typed 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 typed as shown, that the user-defined entity *v* is required, and that one or more *v* items can be specified.