Chapter 1

The IRIX Operating System

This chapter provides a brief overview of the IRIX operating system, Silicon Graphics' implementation of the UNIX operating system. It describes IRIX compliance with standards, summarizes the features unique to IRIX (such as support for graphics hardware), briefly discusses writing device drivers (which run as part of the IRIX kernel), and lists the hierarchy of tools and libraries you can use when you write an IRIX application.

About the IRIX Operating System

IRIX, Silicon Graphics' implementation of the UNIX operating system, is based on UNIX System V, Release 4 (SVR4).

Standards Compliance

IRIX provides standard SVR4 programming interfaces and BSD networking, and complies with the following standards:

- System V Interface Definition, Issue 3 (SVID3), which is the defining document for SVR4. IRIX provides the SVR4 Applications Programming Interface (API) and the Applications Binary Interface (ABI) as defined in SVID3.
- *X/Open Portability Guide, Issue 3 (XPG3)*, which specifies a set of programming interfaces to be provided by operating systems in order to facilitate the writing of portable programs.
- POSIX P1003.1, another standard for portable programming.
- X Window System, Version 11, Release 6 (X11R6). IRIX provides the industry-standard X Window System™.
- OSF/Motif™ Release 1.2. IRIX includes IRIS IM™, the Silicon Graphics port of the industry-standard OSF/Motif user-interface toolkit.
- OpenGL™. IRIX provides a full implementation of this standard for 3D graphics.

Internationalization

Internationalization is the process of making a program capable of running in more than one spoken-language environment without recompiling. Internationalized software can be made to produce output in a user's native language, to format data (such as dates and currency values) according to local custom, and in other ways make the software more comprehensible for people whose culture is different than that of the original software developer.

Table 1-1 lists the internationalization features supported by IRIX and the standards with which this implementation complies.

Table 1-1 IRIX-Supported Internationalization Features

Feature	Standard
Locales	ANSI C and POSIX (ISO0045-1)
XPG/3 message catalogs and interpretation of locale strings	*X/OPEN Portability Guide, Issue 3* (XPG/3)
Multi-National Language Support (MNLS) message catalogs	UNIX System V Release 4
Input methods Text rendering Resource files	X11R6

The chapter titled "Internationalization" in *Topics in IRIX Programming* describes functions you can use and guidelines you can follow to create an application that runs in more than one spoken-language environment.

Networking

The network programming facilities available with the IRIX operating system include the following:

- The Transport Layer Interface (TLI) defined in ISO-OSI, using System V Release 4 STREAMS modules.

  The International Standards Organization (ISO) has developed a standard known as the Reference Model of Open Systems Interconnection (ISO-OSI). This model uses a layered view of networking. The TLI, which defines an interface between two of these layers, provides a set of functions that applications can call to perform various network operations. TLI conforms to the MIPS® ABI (Application Binary Interface).

- Network interfaces (sockets) defined by the 4.3 release of the Berkeley Software Distribution (BSD).

  The 4.3BSD Inter-Process Communication (IPC) facility provides a socket interface that enables low-level access to network addressing and data transfer. IRIX supports *libsocket*, the standard SVR4 method of accessing the BSD4.3 networking interface. The *libsocket* library provides a socket interface that is MIPS ABI compliant.

- An implementation of the Sun Microsystems® Remote Procedure Call (RPC) library.

  RPC implements a remote procedure call model, in which a procedure executing on a remote system can be treated as a local procedure call by the calling application. RPC enables synchronous execution of procedure calls on remote hosts and provides transparent access to network facilities.

The IRIX operating system implements the Internet Protocol (IP) suite. The IP suite is a collection of layered protocols developed by the Department of Defence Advanced Research Project Agency (DARPA). The two most widely used IP protocols are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). The IRIX operating system implements the Internet Protocol suite and UNIX domain sockets using the 4.3BSD UNIX socket mechanism.

X Window System

The Silicon Graphics native X Window System is based on the X Version 11 Release 6 (X11R6) standard that supports multiple visuals, overlay windows, the X Input Extension, the Display PostScript™ Extension, the Shape Extension, and the Shared Memory Extension. Read "X Window System" for more information about the X Window System on Silicon Graphics platforms.

Silicon Graphics Extras

In addition to the standard UNIX features described in "About the IRIX Operating System", IRIX provides software that supports the graphics and multiprocessing capabilities of Silicon Graphics platforms. These added features include:

- Fast 3D graphics
- Parallel programming
- Real-time enhancements
- High-performance I/O
- An icon-oriented interface to IRIX

Fast 3D Graphics

Most Silicon Graphics systems are equipped with specialized graphics hardware and software. Programs can take advantage of this graphics support by using:

- OpenGL interfaces—the industry standard interfaces for 3D rendering across multiple platforms
- Open Inventor™—a graphics library, built on OpenGL, for creating interactive graphics applications
- IRIS Graphics Library™ (IRIS GL)—a library of subroutines for creating color graphics and animation

- IRIS Performer™—a graphics library, built on IRIS GL, for creating real-time graphics and visual simulation applications
- ImageVision Library™—an image processing library built on IRIS GL

"Graphics Libraries" and "ImageVision Library" describe these libraries in greater detail.

Parallel Programming

IRIX is designed to take advantage of multiple processors and provides interfaces to support parallel programming. These interfaces include:

- Low-overhead inter-process communication routines—The IRIX IPC mechanisms use a shared arena (shared memory) for communication between processes. This arena is mapped into a process's user space, which means most of the shared arena IPC functions don't have to make system calls. This keeps overhead lower than for standard System V IPC. Read the chapter titled "Inter-Process Communication" in *Topics in IRIX Programming* for more information about shared arenas.
- Multiprocessor control routines—IRIX supports the sysmp(2) commands MP_RESTRICT and MP_MUSTRUN, and the runon(1) and mpadmin(1) commands. These allow you to control the distribution of processes among the processors in a multiprocessor system. Read the *REACT/Pro Release Notes* for more information about these commands.
- Shared address space (sproc)—This system call creates a new process that is a clone of the calling process and that shares the virtual address space of the calling process.

Real-Time Enhancements

IRIX includes a number of extensions to enable programs to achieve real-time response. You can use these features to

- accurately time events
- control allocation of memory to the process
- provide for priority scheduling

The *REACT/Pro Release Notes* tell you where you can find more information about the mechanisms available to help you achieve real-time performance and how to use these features together to create the best real-time environment.

High-Performance I/O

IRIX includes several special programming interfaces for high-performance I/O. These include:

- Memory-mapped files. You can use the mpin(2) and munpin(2) commands to lock data into memory, thus reducing I/O activity.
- Asynchronous I/O—IRIX has added support for asynchronous I/O in accordance with the specification in POSIX 1003.4a Draft 12.
- Direct I/O—This bypasses the system buffer cache.

These interfaces are described in the *REACT/Pro Release Note*s.

Indigo Magic Environment

The Indigo Magic™ desktop is an end-user environment that provides an icon-oriented interface to the IRIX system. Users can launch applications and select files using icons instead of pathnames. You can integrate your application into the desktop environment to give it the Indigo Magic look and feel.

"Indigo Magic Desktop Integration" provides an overview of desktop integration. The *Indigo Magic Desktop Integration Guide* covers the topic in detail.

Device Drivers

A device driver enables communication between a user process and a peripheral device. In addition to kernel-level drivers, there are user-level drivers and STREAMS modules. Device drivers perform functions such as taking the device online and offline, transmitting data from the kernel to the device, receiving data from the single device and passing it to the kernel, and handling and reporting I/O errors.

Silicon Graphics provides drivers for many popular devices. However, you still may find that you need to write a device driver to support a device for which no driver is available.

The IRIX filesystem provides a device-independent interface that allows device drivers to be opened, read, written, and closed as though they were files. The program issues standard system I/O calls, and the system then calls the driver you've written to handle the device.

Figure 1-1 illustrates the relationship between the various software modules that connect the user application and the device with which it needs to communicate.

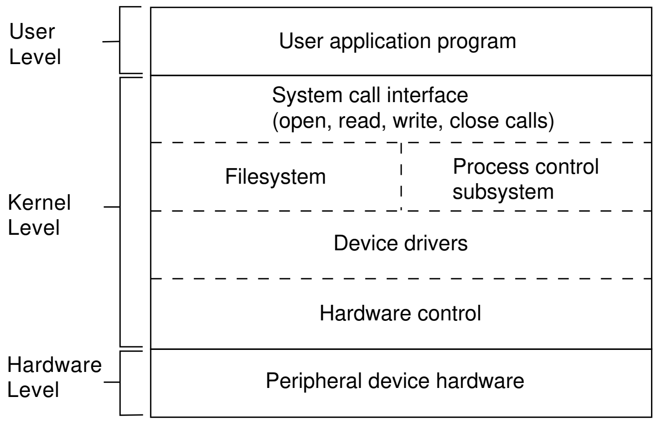

Figure 1-1 Driver Position in the Kernel

The kernel interfaces for IRIX device drivers are

- SVR4 Device Driver interface/Driver Kernel Interface (SVR4 DDI/DKI). IRIX uses the multiprocessor version of DDI/DDK.
- SVR4 STREAMS Interface, documented in the *UNIX System V Release 4 Streams Programming Guide*.
- IRIX 5.x Data Link Provider Interface (DLPI).
- User-level device drivers for VME, EISA, and SCSI.

The following list describes some Silicon Graphics-specific features you may need to include in a device driver:

- Memory mapping extensions (map and unmap)—the Silicon Graphics-specific memory mapping functions used by a device driver.
- Utility extensions—the Silicon Graphics-specific utility functions a developer needs to include in a device driver.
- Data structure extensions—data structure extensions supported by Silicon Graphics include *eisa_dma_buf* and *eisa_dma_cb*.
- Four new kernel definitions.

Writing an Application Program

The IRIS Developer's Option (IDO) is required for writing applications that run on Silicon Graphics platforms. This option provides the basic software and documentation for the development environment. "The IRIS Developer's Option" tells you more about the IDO.

When you write an application for a Silicon Graphics platform, you can choose from a number of languages, toolkits, and libraries supplied by Silicon Graphics for developers. Your choice depends on the nature of your application and, to some degree, on your personal preference. Here are some choices you can make and the options available under IRIX:

- A programming language. Choose a language that is best suited for your application. Silicon Graphics provides compilers for C, C++, Fortran, Pascal, and Ada®. If your application runs on a multiprocessing platform and performance is a key consideration, you may want to choose either Power C or POWER Fortran, the versions of C and Fortran with special support for multiprocessing.
- A graphical user interface. If you plan to incorporate a graphical user interface in your application, you can use one of the user interface toolkits provided by Silicon Graphics rather than writing your own. These toolkits are IRIS ViewKit™ and IRIS IM, which is Silicon Graphics' port of the industry-standard OSF/Motif. They allow you to build an interface for your application that looks like the interfaces of other applications running in the Indigo Magic desktop environment.
- Software development tools. When you order the IDO, you receive tools for compiling, debugging, and tuning your application. These tools include the compiler driver, the *dbx* debugger, and several performance tuning tools. You can also order the CASEVision WorkShop, which provides several interactive, graphical tools for debugging and tuning your application. If your project is large, you may need configuration management and version control. You can use the CASEVision™/Tracker and CASEVision/ClearCase tools for these tasks.
- An application library. You may want to use one of the application libraries developed by Silicon Graphics. These libraries provide tools for developers of 3D graphics, image processing, digital media, and printer/scanner management applications.

Figure 1-2 shows, in a hierarchical arrangement, the components you can select.

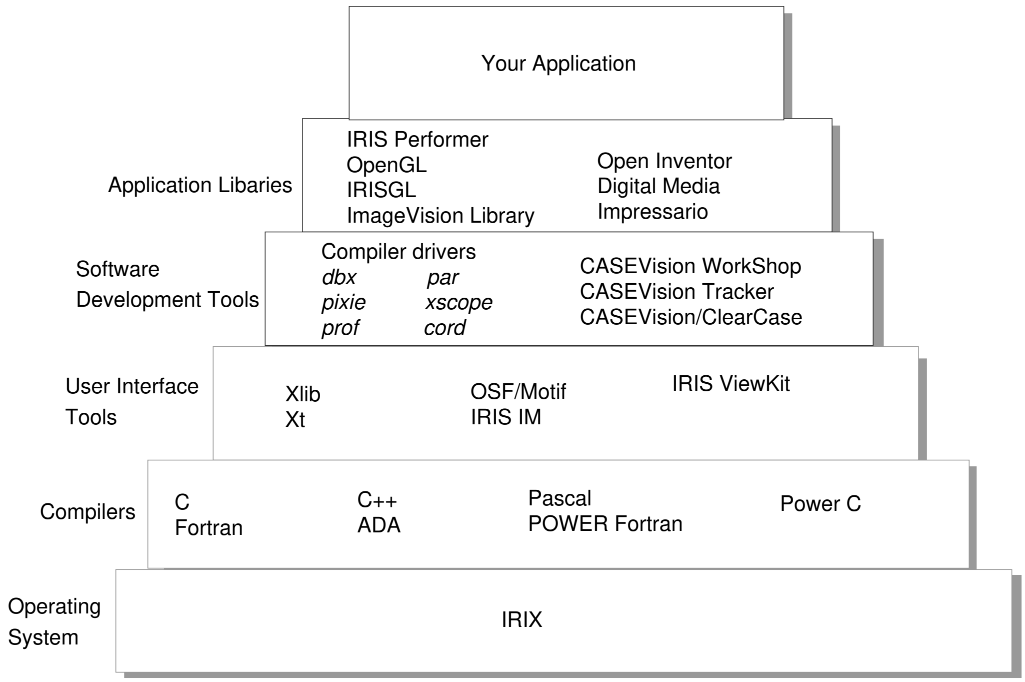

Figure 1-2 Component Hierarchy of an IRIX Application Program

Now that you have an overview of the IRIX programming environment, you're ready to learn more about the tools and libraries available and the documentation Silicon Graphics provides for reading about them. Chapter 2 lists the IRIX programming documentation. Chapters 3 through 6 describe IRIX compilers, tools, and application libraries in greater detail.