Solaris 1.x to 2.x Transition Guide

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043-1100
U.S.A.

Part No: 802-5758–10
August 1997

Contents

Preface

The Solaris™ 2.6 operating environment, SunSoft's™ distributed computing solution, comprises SunOS™ release 5.6 software with ONC™, OpenWindows™, ToolTalk™, and DeskSet™ utilities as well as other features. *Solaris 1.x to Solaris 2.x Transition Guide* focuses on the differences between the SunOS release 4.x and SunOS release 5.6 operating systems for people already familiar with the SunOS release 4.x software. This guide also handles other aspects of the Solaris 2.6 operating environment that can help you through the transition.

If you are looking for more information about features now available with the Solaris 2.6 operating environment, see *Solaris User's Guide.*

The system administration tool covered in this book, Admintool, is part of the Solaris 2.6 product and can be used only for local system administration. System administration tools used to manage a network of systems are provided with the Solstice family of products.

Note - The term "x86" refers to the Intel 8086 family of microprocessor chips, including the Pentium and Pentium Pro processors and compatible microprocessor chips made by AMD and Cyrix. In this document the term "x86" refers to the overall platform architecture, whereas "*Intel Platform Edition*" appears in the product name.

Who Should Use This Guide

This guide can help users, system administrators, and software developers make the transition from a SunOS release 4.x computing environment to the Solaris 2.6 operating environment.

What to Expect From This Guide

The purpose of this guide is to give you an overview-level understanding of the differences between SunOS release 4.x and SunOS release 5.6 operating environments to make your transition to the Solaris 2.6 operating environment a smooth one. As a result, *Solaris 1.x to Solaris 2.x Transition Guide* covers a wide range of topics. Because it is not practical to list detailed procedures for tasks here, you will find references throughout this guide to publications in the Solaris 2.6 documentation set where detailed information is available.

How This Guide Is Organized

This guide is divided into 2 parts with 19 chapters and 6 appendixes as outlined here.

Part 1: Transition Information for Users and System Administrators

You can use this part of the guide to help install Solaris 2.6 software, to understand changes to the local computing environment, and to understand changes to routine tasks.

This part of the guide contains the following chapters:

- Chapter 1, discusses the benefits of migrating to the Solaris operating environment and summarizes the main differences between SVR4 and the Solaris operating environment.
- Chapter 2, is an overview of some of the principal changes between the SunOS release 4.x and SunOS release 5.6 environments. It provides background for topics in subsequent chapters, focusing on procedures, tools, and concepts that have changed between releases.
- Chapter 3, suggests what to consider to facilitate a smooth transition through software installation and post-installation so that SunOS release

4.x data can most easily be restored in the Solaris 2.6 operating environment.

- Chapter 4, discusses the SunOS/BSD Source Compatibility Package and the Binary Compatibility Package. These packages make the transition easier by enabling you to use SunOS release 4.x commands and applications during migration to the Solaris 2.6 operating environment.
- Chapter 5, describes the major differences between SunOS release 4.x and Solaris 2.6 security, and points out how those changes might affect system administration procedures.
- Chapter 6, describes differences in tasks used to set up a local user environment after installing the Solaris software. It includes discussions on setting up a default shell, customizing the user environment, the window system, and user and group administration. It also discusses changes regarding man pages.
- Chapter 7, explains SunOS release 5.6 device naming conventions and discusses changes to device-related tasks such as getting information about disks, adding devices to a system, and using volume management.
- Chapter 8, describes changes to procedures for booting and shutting down a system.
- Chapter 9, familiarizes you with changes to file-system layout and the changes to file systems, virtual file systems, directories, and files. It also describes changes to file-system administration.
- Chapter 10, discusses setting up servers for clients. It describes three programs–`discover4x`, `install4x`, and `convert4x`–that work together to help prepare a Solaris 2.6 server to serve SunOS release 4.x clients.
- Chapter 11, describes how to set up and administer printers after you install Solaris 2.6 software and changes to printer commands. It also describes terminal and modem management using Admintool and the Service Access Facility (SAP).
- Chapter 12, outlines changes to the network facilities, TCP/IP and UUCP.
- Chapter 13, discusses NIS+ and the domain name system (DNS), and compares NIS+ to NIS and DNS.
- Chapter 14, describes the Common Desktop Environment (CDE) and how to make the transition from the OpenWindows environment to CDE.

Part 2: Transition Information for Developers

This part of the guide concentrates on the changes that most affect developers. It describes these differences, points out similarities, and explains the implications for your programming environment.

This part contains the following chapters:

- Chapter 15, discusses which capabilities have been added to or removed from compilers, linkers, and debuggers.
- Chapter 16, discusses changes to tools and resources for the development environment including changes to `ioctl`() requests, `ptrace`() request values, libraries, and the make and SCCS facilities. This chapter also describes how to determine application compatibility, how to use Solaris 2.6 packaging capabilities, and how to find SunOS release 4.x tools.
- Chapter 17, discusses Solaris 2.6 networking features as they relate to the programming environment. It also describes improved internationalization features.
- Chapter 18, describes aspects of system and device configuration that have changed, including the dynamically loaded kernel and kernel layout, `config` and `boot` commands, and the `/etc/system` file.
- Chapter 19, discusses device drivers issues such as changes to device driver interfaces, the `devinfo` command, porting considerations, STREAMS, and the Solaris 2.6 driver architecture.

Reference Appendixes

The following appendixes comprises reference tables showing SunOS 4.1 interfaces and their status in several operating systems. This information is useful to users, system administrators, and developers. The appendixes are:

- Appendix A, compares SunOS release 4.x and SunOS release 5.6 commands.
- Appendix B, compares SunOS release 4.x and SunOS release 5.6 system calls.
- Appendix C, compares SunOS release 4.x and SunOS release 5.6 library routines.
- Appendix D, compares SunOS release 4.x and SunOS release 5.6 system files.
- Appendix E, compares SunOS release 4.x and SunOS release 5.6 system files.
- Appendix F, is a quick reference for changes in common commands, files and directories, and daemons and standard processes.

Ordering Sun Documents

The SunDocs™ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of the SunExpress™ Internet site at http://www.sun.com/sunexpress.

What Typographic Changes and Symbols Mean

Table P-1 describes the type changes and symbols used in this guide.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `system% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`system% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename.*
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root to do this.

Code samples are included in boxes and may display the following:

TABLE P–1 Typographic Conventions *(continued)*

Typeface or Symbol	Meaning	Example
%	UNIX C shell prompt	system%
$	UNIX Bourne and Korn shell prompt	system$
#	Superuser prompt, all shells	system#

man Page References

When commands, system files, or library routine names are first mentioned in the text, the number of the manual page section where the term is fully described is appended; for instance: mv(1). The manual pages are in the *man Pages(1): User Commands* section.

Related Books

For more information the Solaris 2.6 operating environment, see the following documentation:

- *Solaris User's Guide*
- *Solaris Advanced User's Guide*
- *Installation Instructions for Solaris 2.6 (SPARC Platform Edition)*
- *System Administration Guide*
- *NIS+ Transition Guide*
- *NFS Administration Guide*
- *Solaris Naming Administration Guide*
- *Solaris Naming Setup and Configuration Guide*
- *TCP/IP and Data Communications Administration Guide*
- *SunOS Reference Manual*

- *Binary Compatibility Guide*
- *Source Compatibility Guide*
- *Developer's Guide to Internationalization*
- *Multithreaded Programming Guide*
- *Linker and Libraries Guide*
- *Programming Utilities Guide*

Getting Help from Sun Microsystems' WWW Site

You can get additional Solaris transition information by accessing the following URL:

`http://www.sun.com/smcc/solaris-migration/index.html`

The Solaris Migration Initiative home page is a central point for the distribution of tools, documentation, and information to aid you in your migration to Solaris 2.x. You can always find the most current resources and pointers here.

PART I Transition Information for Users and System Administrators

You can use this part of the guide to help install Solaris 2.6 software, and to understand changes to the local computing environment and to routine tasks.

CHAPTER 1

Introduction

The Solaris operating environment enhances your system's capabilities with powerful tools and features. This introduction discusses the benefits of migrating to the Solaris operating environment and summarizes the principal differences between SVR4 and the Solaris operating environment.

Advantages of Migrating to the Solaris Operating Environment

The UNIX® standard, SVR4, accommodates the leading UNIX variants (System V, BSD, SunOS, and XENIX), uniting the majority of the installed base of UNIX users. The Solaris operating environment, based on SVR4, gives software developers, system administrators, and end users the benefits of a standard operating system including broad compatibility, a growth path, and reduced time to market. It also delivers a functional and powerful product reflecting years of refinement. Among the many advantages the Solaris operating environment provides are portability, scalability, interoperability, and compatibility.

Although the foundation of the Solaris operating environment is based on SVR4, SunSoft has added extensive functionality in areas such as symmetric multiprocessing with multithreads, real-time functionality, increased security, and improved system administration.

The Solaris operating environment offers the following features:

- SunOS release 5.6, a full 32-bit operating system based on UNIX System V Release 4 (SVR4)
- Cross-functional compatibility, which enables the SunOS release 5.6 software to run on SPARC™ as well as Intel 386, 486, Pentium and other DOS-compatible CPUs
- Industry standards including SVR4 and the ONC family of networking protocols
- Graphical user interface (GUI) in the OPEN LOOK® Window Manager
- Common Desktop Environment, a desktop graphical interface. This window environment helps you organize and manage your work. The desktop provides windows, workspaces, controls, menus, and a Front Panel allowing simple access to Mail, File Manager, Printers, Image Tool, Calendar Manager, and others
- Calendar Manager, a time management application that displays appointments and ToDo items at a glance and offers a multibrowse feature that makes scheduling among a group easy
- File Manager, a graphical and intuitive way to navigate to local and remote file systems
- Image Tool, which enables you to load, view and save images of over 40 different formats
- Audio, a new, Motif-based audio application for playing and recording AU, WAV, and AIFF files.
- OPEN LOOK Admintool, the base for local system administration
- Installation GUI for easing install and update
- Log-based file systems on servers
- Advanced architecture that includes fully symmetric multiprocessing and sophisticated multithreading
- Real-time priority scheduling and a fully pre-emptible kernel, providing the benefits of open systems while meeting the requirements of control applications
- Network Information Services Plus (NIS+), an upward-compatible version of the NIS name service with simpler hierarchical administration, improved security, and faster updates.
- Standards conformance for application developers interested in the benefits of application portability
- Multimedia Mail, for sending messages that incorporate audio, graphics, and embedded files
- Java Virtual Machine™, provides access to the Java platform for the Solaris operating environment

- WebNFS, makes it possible to make a file system accessible through a Web browser.
- AnswerBook2™ Viewer, Sun's premier online documentation system that uses a web-browser-based interface

Portability, Scalability, Interoperability, and Compatibility

The Solaris operating environment is portable, scalable, interoperable, and compatible.

Portability

The SunSoft SunOS product is portable across multiple vendor platforms. Software conforming to an application binary interface (ABI) runs as shrink-wrapped software on all vendor systems with the same microprocessor architecture. This enables application developers to reduce software development costs and bring products to market quickly, and enables users to upgrade hardware while retaining their software applications and minimizing conversion costs.

Scalability

Over time, applications become more widely used, and require more powerful systems to support them. To operate in a growing environment, software must be able to run in a wide power range and must be able to take advantage of the additional processing power. SunSoft's operating system runs on machines of all sizes, from laptops to supercomputers.

Interoperability

Heterogenous computing environments are a reality today. Users purchase systems from many vendors to implement the solutions they need. Standardization and clear interfaces are critical to a heterogeneous environment, enabling users to develop strategies for communicating throughout their network. Solaris systems can interoperate with every popular system on the market today, and applications running on UNIX can communicate easily.

Compatibility

Computing technology continues to advance rapidly, but the need to remain competitive requires vendors to minimize their costs and to maximize their investments. SunSoft will ensure that as new technology is introduced, the existing software investment is preserved. Users can take advantage of today's solutions and still be compatible with tomorrow's technologies.

Advantages for Large Organizations

The Solaris operating environment provides a number of sound business reasons for transitioning to an industry-standard-based UNIX operating system. Application development and maintenance costs are lower, and application portability is enhanced.

Comparison of SVR4 and the Solaris Operating Environment

This section describes the main differences between SVR4 and the Solaris operating environment. It points out features that the Solaris operating environment includes that are not available in SVR4 and a few SVR4 features that are not available in the Solaris operating environment.

Additional Features in the Solaris Operating Environment

The Solaris operating environment offers value-added components in addition to the SVR4-based operating system. These make computing easier and create new opportunities for users, system administrators, and developers.

In general, the merge of established UNIX variants into SVR4 and the Solaris operating environment was done by consolidating the existing functionality while maintaining compatibility for existing applications. As a result, features and commands were added to the product with few features being withdrawn.

Features for the User

For users, the Solaris operating environment incorporates a suite of powerful DeskSet applications to enhance personal productivity. All DeskSet applications rely on the drag-and-drop metaphor, enabling users to carry out complex UNIX commands with a mouse. Specifically, some of the features are:

- *A workspace manager*. Provides basic window management services (open, close, move, and so on), as well as tools that enables user to customize their workspace.
- *Desktop integration services*. These include ToolTalk, drag and drop, and cut and paste, providing the foundation that enables applications to seamlessly integrate with one another.
- *Graphics libraries*. These include XGL™, Xlib, PEX™, and XIL™, providing support for 2D and 3D graphics applications.
- *Calendar Manager*. A time management application that displays appointments and ToDo items for a day, week, or a month at a glance. It also contains a multibrowse feature that makes scheduling meetings among a group of users easy. Multiple calendars can be overlaid simultaneously to determine convenient meeting time slots at a glance.
- *Image Tool*. Enables you to load, view and save images of over 40 different formats including PICT, PostScript™, TIFF, GIF, JFIF, and many more.
- Other tools include a print tool, audio tool, shell tool, clock, and text editor.

Features for the System Administrator

For system administrators, the Solaris operating environment offers a variety of new tools to simplify the administration of a distributed computing environment. These include:

- *Device information*. Administrators can use these optional utilities to obtain information about installed devices including device names, attributes, and accessibility. Administration can be simplified by creating device allocation pools, a feature not previously found in UNIX systems.
- *File system administration*. These utilities enable administrators to create, copy, mount, debug, repair, and unmount file systems; create and remove hard file links and named pipes; and manage volumes.
- *Interprocess communication*. Two interprocess communication utilities create, remove, and report on the status of the system's interprocess communication facilities (message queues, semaphores, and shared memory IDs). They provide information helpful in tuning the system.
- *Process management*. The process management utilities help you control system scheduling. Using these utilities, administrators can generate

reports on performance, logins, disk access locations, and seek distances to better tune system performance. In addition, you can change the system run level, kill active processes, time the execution of commands, and change the default scheduling priorities of kernel, timesharing, and real-time processes.

- *System accounting*. The accounting utilities enable system administrators to track system usage by CPU, user, and process for better resource allocation.
- *System information*. These utilities report system memory and system configuration. The system administrator can use the utilities to change the names of the systems and the network node.
- *User and group management*. With these utilities, a system administrator can create and delete entries in group and password databases, specify default home directories and environments, maintain user and system logins, and assign group and user IDs. The utilities support both primary and supplementary user groups.
- *Admintool*. Admintool, which runs under the OpenWindows environment, provides system management facilities to help add hosts, manage the network, and perform many other routine tasks on local systems.
- *Auto configuration*. The Solaris operating environment has a dynamic kernel, which means that it loads drivers and other modules into memory when the devices are accessed. You no longer need to rebuild the kernel after installation, nor must you add or remove drivers.
- *Network Information Services Plus (NIS+)*. An upward-compatible version of the NIS name service with simpler hierarchical administration, improved security, and faster updates.
- *Installation*. The Solaris operating environment has an install GUI to ease installation or upgrades. Automatic installations and upgrades are also possible over the network.
- *Security*. The automated security enhancement tool (ASET) is a utility that improves security by allowing system administrators to check system file settings including permissions, ownership, and file contents. ASET warns users about potential security problems and, where appropriate, sets the system file permissions autonomically according to the specified security level.

Features for the Developer

For application developers, the Solaris operating environment includes a variety of toolkits and features to simplify the development of complex applications with graphical user interfaces.

- *Multithreaded (MT) kernel*. MT provides for a symmetric multiprocessing kernel where multiple processors can execute the kernel at the same time.

Applications can be structured as several independent computations rather than as one thread of control. Independent computations execute more efficiently because the operating system handles the interleaving of the independent operations. This benefit of multithreading is known as *application concurrency*.

- *STREAMS.* STREAMS is a flexible framework for character input and output (I/O) that has been implemented throughout SVR4. It is easily customized for applications.
- *Expanded fundamental types.* ID data types (`uid`, `pid`, device IDs, and the like) and certain other data types are expanded to 32 bits. This improves the scalability of the operating system in large systems and for use in large organizations.
- *Device driver interfaces.* There are three types of interfaces for Solaris device drivers: Device Kernel Interface (DKI) Device Driver Interface/Device Kernel Interface (DDI/DKI), and Sun Device Driver Interface (Sun DDI). The DDI/DKI conformance means that device drivers have better source and binary compatibility across SPARC platforms so developers can write one driver to support a peripheral on all SPARC platforms.
- *Automatic device driver loading.* This makes drivers easier to install and devices easier to access.
- *Dynamic linking.* The Solaris application environment supports static and dynamic linking of libraries. The linker uses the version numbers of the libraries and executables to link applications with the proper libraries, routines, and interfaces.

SVR4 Features Excluded From the Solaris Environment

In a few instances, features in SVR4 were not include in the Solaris operating environment. These features are specific to AT&T hardware, or features included primarily for backward compatibility with SVR3 features and therefore, are of little value to SunOS users.

The Solaris operating environment does not include the System V file system and associated utilities because of their limitations compared to the UNIX file system. The SVR4 boot file system was not included because of its maintenance burden when compared to the SunOS traditional boot model.

The generic AT&T SVR4 model for device auto-configuration and for rebuilding kernels was replaced with a fully dynamically configurable kernel better suited to the needs of present and future users of SPARC systems.

Because there is no installed base of SPARC XENIX programs, the SPARC release of the Solaris operating environment does not include compatibility for XENIX applications.

The Solaris operating environment does not include the AT&T SVR4 `sysadm` utility. Because the `sysadm` menu utility was designed primarily for use with terminal devices on freestanding systems, SunSoft chose to concentrate its efforts on tools with graphical user interfaces that simplify the administration of distributed systems across a network. The Solaris operating environment provides the utilities and configuration directories that underlie the SVR4 `sysadm` utility but not the `sysadm` utility itself.

CHAPTER 2

Overview of Major Changes

As you use the Solaris 2.6 operating environment, you will find similarities to the SunOS release 4.x operating environment; however, you will also notice some differences. The rest of this guide focuses on the procedures, tools, commands, and concepts that have changed between releases.

This chapter is an overview of some of the principal changes. It provides background information for topics in subsequent chapters. Some topics receive sufficient coverage here, while others require more in-depth technical background. In the latter case, the text directs you to a chapter that more fully describes the changes.

Software Packages and Clusters

Solaris 2.6 system software is delivered in units known as *packages.* A package is a collection of files and directories required for a software product. A *cluster* is a collection of packages.

The list below describes four clusters. Note that as you progress through the list, each cluster contains the software of the preceding cluster plus additional software.

- *Core System Support* is the minimum software configuration; it contains only the software necessary to boot and run the Solaris 2.6 operating environment.
- *End User System Support* contains Core System Support plus end user support such as the OpenWindows windowing system and the related DeskSet application files; this cluster includes the recommended software for an end user.
- *Developer System Support* contains End User System Support plus the libraries, include files, and tools needed to develop software in the Solaris 2.6 operating environment. Compilers and debuggers are not included in the Solaris 2.6 operating environment.
- *Entire Distribution* contains the entire Solaris 2.6 environment.

For more information about this section's topics, see *System Administration Guide.*

Package Administration

Software package management simplifies installing and updating software. Administration is simplified because the method for managing system software and third party applications is now consistent. The tools for creating software packages are in an application packaging tools library.

There are two tools you can use to install and remove packages:

- A graphical user interface program (see the `admintool`(1M) man page)
- The command-line utilities (see the `pkgadd`(1M) and `pkgrm`(1M) man pages)

Graphical User Interface (admintool)

You can install software on your local system or on a remote system with Admintool (started with the admintool command). The default location for the installation is the local system.

Use Admintool to:

- Look at the software installed on the local system
- Install or remove software on a local system

If you want to install or remove the software, you must run Admintool as superuser or as a user in the sysadmin group (group 14). You do not need to be superuser to look at the software packages that are already installed on a system.

Command-Line Utilities

You can use command-line utilities to install, remove, and check the installation of software packages. The commands are:

- pkgadd(1M) for installing a package
- pkgrm(1M) for removing a package
- pkgchk(1M) for checking the installation of a package
- pkginfo(1M) for listing the packages installed on a system

Patch Administration

The patchadd(1M) and patchrm(1M) commands are used to install and remove patches from a Solaris 2.x system. You can add one or more patches to a system, client, service or net install image.

See patchadd(1M) and patchrm(1M) for more information.

Disk Slices (or Partitions)

A single range of contiguous blocks or a physical subset of a disk is known as a disk *partition* in the SunOS release 4.x software. In the SunOS release 5.x software, a physical subset of a disk is known as a disk *slice*. Before you can create a file system on a disk, you must format and divide it into slices. This is usually done when the Solaris release is installed using the Solaris 2.x installation program. See *System Administration Guide* if you need to install and format a disk after installation.

Note - In some Solaris documentation, Solaris slices are still referred to as "partitions". The Solaris 2.x documentation distinguishes between `fdisk` partitions (for Intel systems) and the divisions within an `fdisk` partition, referred to interchangeably as slices or partitions.

See *System Administration Guide* for information about Solaris `fdisk` partitions.

A slice can be used as a raw device for swap space or to hold one and only one UFS file system, unless you are using a product like Solstice™ DiskSuite™. Table 2–1 describes how disk slices can be set up on each Solaris 2.x platform.

TABLE 2–1 Slice Differences on Platforms

SPARC	Intel-based
The whole disk is devoted to the Solaris operating environment.	The disk is divided into four `fdisk` partitions, one per operating environment.
The disk is divided into eight slices, numbered 0-7.	The Solaris `fdisk` partition is divided into 10 slices, numbered 0-9. Only 0-7 can be used to store user data.

See *System Administration Guide* for a description of customary disk slice assignments for each platform.

Cylinder Groups

You create a UFS file system on a disk slice, which is divided into one or more areas called *cylinder groups*. A cylinder group is composed of one or more consecutive disk cylinders (the set of tracks on a group of platters that have the same radial distance from the center of the platter). See *System Administration Guide* for a complete description of disk geometry.

A *cylinder group map* is created for each cylinder group. The cylinder group map records the block usage and available blocks.

Figure 2–1 shows the relationship between disk slices and cylinder groups.

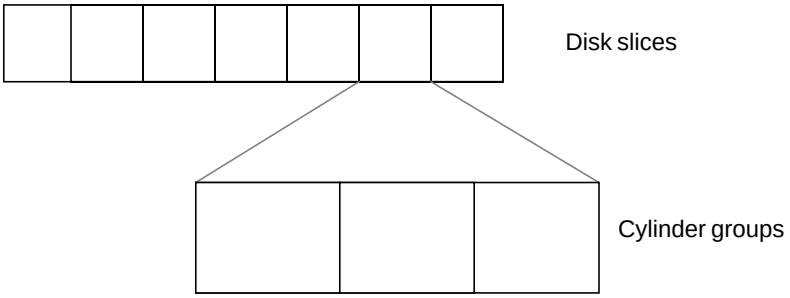


Figure 2–1 Disk Slices and Cylinder Groups

Device Naming

SunOS release 5.6 device names make it easier to infer certain device characteristics from a device name. SunOS release 4.x systems convey type rather than device attributes, which makes it difficult for programs and scripts to derive necessary information about devices. SunOS release 5.6 conventions are slightly different from AT&T SVR4 device names because SunOS release 5.6 allows only eight user-configurable slices on a disk.

In addition, special device files are now stored in the hierarchical `/devices` directory, with symbolic links to the hierarchical `/dev` directory, which is used by administrators and users to access devices. The `/dev` directory contains subdirectories, such as `/dev/dsk/*`, used to access disk devices, and `/dev/rdsk/*`, used to access raw disk devices. For more information, see “Device Naming Conventions” on page 57. For discussions on device naming conventions, see “Device Naming From a Developer’s Perspective” on page 187.

File Systems

SunOS release 5.6 and SunOS release 4.x file systems are similar, but there are changes in the locations and names of system directories and files. There are also new file systems, new pseudo file systems, and one directory is not used.

“File System Changes” on page 73, describes file system changes. *System Administration Guide* describes file system concepts and administration in detail.

Changes to File System Locations and Names

Some of the changes to file system locations and names are:

- The `/dev` directory has changed from a flat directory to a hierarchical one.
- The `/etc` directory contains system configuration information. Several files and subdirectories have been added, removed, or changed.
- The `/etc/vfstab` tab file replaces `/etc/fstab`.
- The `/etc/lp` directory replaces `/etc/printcap`.
- The SunOS release 5.6 `/sbin` directory contains the `rc` scripts used to alter system run levels as well as the `rcs` script used to initialize the system prior to mounting file systems.
- The SunOS release 5.6 `/usr` directory contains sharable files and executables provided by the system.
- The `/var` directory contains files that change sizes during normal operation. Several files and subdirectories in the `/var` directory have been added, removed, or changed.
- The `/var/mail` directory replaces `/var/spool/mail`.
- The `/sys` directory is no longer needed because the kernel is dynamically loaded.
- The `terminfo` database replaces `termcap`.
- The SunOS release 5.6 core kernel is called `genunix`, and the kernel modules are stored in the `/kernel`, `/usr/kernel`, `/platform`, and `/usr/platform` directories.

Pseudo File Systems

Pseudo file system types are logical groupings of files that reside in disk-based systems. The TFS pseudo file system is not included in the SunOS release 5.6 software.

The SunOS release 5.6 pseudo file systems are:

- CACHEFS pseudo file system – can be used to improve performance of slow devices such as a CD-ROM drive.
- PROCFS pseudo file system – resides in memory and contains a list of active processes, by process number, in the `/proc` directory. See the `proc`(4) man page.
- FDFS pseudo file system – provides explicit names for opening files using file descriptors.
- FIFOFS pseudo file system – contains pipe files that give processes common access to data.

- NAMEFS pseudo file system – used mostly by STREAMS for dynamic mounts of file descriptors on top of files.
- SWAPFS pseudo file system – the default swap device when the system boots, or you create additional swap space.

Added File Systems

The following file systems are included in the SunOS release 5.6 directory structure:

- The optional `/opt` file system, which can be used to store third-party or unbundled software. If `/opt` is not a separate file system, it may be a symbolic link to `/usr/opt`.
- The `/vol` file system, which provides the default file system for the Volume Management daemon, `vold`(1M). See the `volfs`(7) man page.

Removed File Systems

Support for the RFS file system type has been removed.

Kernel Configuration

Unlike in the SunOS release 4.x software, the SunOS release 5.6 kernel is dynamically configured. This means that you no longer need to rebuild it manually when you make changes to the system configuration.

Starting with release 5.5 of the SunOS software, the kernel and its modules were separated into platform-independent and platform-dependent objects. The platform-independent kernel consists of a small static core, called `/kernel/genunix`, and its dynamically loadable kernel modules are stored in the `/kernel` and `/usr/kernel` directories if they are platform independent, and `/platform` and `/usr/platform` if they are platform independent. See *System Administration Guide* for a description of the platform-dependent directories and their contents.

Drivers, file systems, STREAMS module, and other modules are loaded automatically as needed, either at boot time or at run time. These modules are unloaded when they are no longer in use. The `modinfo`(1M) command provides information about the modules currently loaded on a system.

The `modload`(1M) and `modunload`(1M) commands are still available in this release but they perform differently. These commands have more limited

usage, and are no longer sufficient to correctly install a loadable driver onto the system. The `modunload`(1M) command is similar to the SunOS release 4.x command, but it includes the capability to unload all unloadable (and not busy) modules, as the following example illustrates.

```
# modunload -i 0
```

Chapter 18, discusses these topics in more detail.

Kernel Layout

The contents of the kernel, which were formerly in a single file, `/vmunix`, are now contained in modules in a platform-independent and platform-dependent directory hierarchy. By default, the directory hierarchy is:

- `/kernel`
- `/usr/kernel`
- `/platform`
- `/usr/platform`

The directory search path for modules can be set by the *moddir* variable in the `/etc/system` file. Typically, `/kernel/genunix` is the first portion of the kernel to be loaded. See `system`(4) and `kernel`(1M) for more information.

Automounting

A new version of the automounter, called AutoFS, has been included. In the SunOS 4.X releases, the automounter mounted everything under `/tmp_mnt` and used symbolic links to redirect the lookups. AutoFS allows for file systems to be mounted in place (for instance, `/home`).

In SunOS 4.X, the maps for the automounter were named `auto.master` and `auto.home`. For Solaris 2.6, these maps have been renamed to `auto_master`, `auto_home`, and so on. The NIS+ name service, which is included with the release, requires this change. A default copy of these maps is included in the release, so that the AutoFS service is started when the system is booted. The SunOS 4.X releases did not include the maps, so additional installation steps were required.

The Solaris 2.6 release provides the ability to select the name service that is being used through `/etc/nsswitch.conf`. The automount entry can be changed to select local files, NIS+, NIS or some combination of these.

Earlier releases supported a home directory naming convention like: `/home/server/login`. With the AutoFS maps it is much easier to use `/home/login` for each entry. This new naming convention also provides for location independence. The old convention can still be used, but once a transition to using the AutoFS maps has been made, it will be easier to administer the shorter paths.

The following paths were reserved for use by AutoFS:

- `/net` - for mounting file systems from a known host
- `/home` - for mounting the home directory of a known user
- `/xfn` - for mounting file systems which support the X/Open XFN standard

On home directory servers, the actual home directories should be moved to `/export/home` rather than `/home`, so that they do not conflict with the automounter directory structure. This also means that you cannot mount file systems on `/home` while the automounter is running.

The AutoFS software now has two programs. The first program is `automount` that runs at boot time to establish AutoFS mount points. This command can also be run anytime by superuser to change the mount points. The second command is `automountd` which is a stateless daemon which answers AutoFS file system mount and unmount requests. These two programs replace the 4.1.X `automount` daemon.

The automount daemon is now fully multi-threaded. Multiple automatic mount requests can be serviced concurrently, which makes AutoFS more reliable. In short, one mount request could block connecting to a slow server, while a second request is processed without waiting.

The Solaris 2.6 release supports browsability of indirect AutoFS maps. All mountable entries under an AutoFS mount point (for example, `/home`) are now visible without the overhead of mounting them first.

Also provided is improved on-demand automounting of hierarchically related file systems. Previous releases would automount an entire set of file systems if they were hierarchically related (for example, `/net/server`) even if only one of the file systems was referenced. The file system that is referenced will be dynamically mounted without mounting all of the other file systems in the hierarchy. Other file systems will be mounted when they are individually referenced.

See “Mounting File Systems and `autofs`” on page 90for more detailed information. Also, *NFS Administration Guide* describes how to use AutoFS.

Mail Administration

The version of `sendmail` that is included on the release is Version 8 compatible. The new version fixes some security holes and includes several improvements to Version 5. Several extensions to the standard BSD release have been added, including name service switch and NIS+ support.

To further support NIS+, a new command, aliasadm, has been included. The command will aid in the administration of NIS+ alias lists.

The mailbox spooling directory has been moved from `/var/spool/mail` to `/var/mail`. A new directory, `/var/mail/:saved`, is used for creating locks and temporary files by the `mailx` program. Also, the mail configuration files are now all located in `/etc/mail`. The new directory includes the `aliases` and the `sendmail.cf` files.

The mailbox locking mechanism has been enhanced so that Solaris 2.6 clients can safely mount mailboxes from both Solaris 2.X and SunOS 4.X mail servers. This enhancement eases administration of mail, especially in large sites.

In the Solaris 2.6 release, `/usr/bin/mailx` supersedes `/usr/ucb/mail`. The `mailx` program has been enhanced to behave the same way as the SunOS 4.x version of `/usr/ucb/mail`. The `/usr/ucb/mail` file is now a symbolic link to `/usr/bin/mailx`.

In SunOS 4.X releases, a program called `sendmail.mx` was used in DNS sites to access mail exchange records. The new version of `sendmail` includes the needed functionality and can be configured through `/etc/nsswitch.conf`.

Mail Administration Guide describes the administration of `sendmail`.

Admintool

One of the major changes between SunOS release 4.x and SunOS release 5.6 that affects system administration is the availability of Admintool to perform basic system administration tasks. This tool employs a graphical user interface to simplify tasks, such as managing users, hosts, printers, and serial devices, on local desktop systems.

Admintool applications enable you to manage the following tasks on a local system:

- System database files such as `aliases` and `netmasks`

- User account and group information, including tasks such as adding users and groups, modifying password aging features, and removing user account information
- Printer setup for local and remote printers
- Terminal and modem setup
- Package management

Using a graphical user interface (GUI) like Admintool to perform administration tasks has the following benefits:

- It is faster than using numerous SunOS commands to perform the same tasks
- System files are updated automatically without the risk of making editing errors
- The application programs interact with appropriate system daemons and notify you when the two are out of sync

---

Note - You do not need to be root to start Admintool but you do need to be a member in the `sysadmin` group (GID=14). Use the `groups`(1) command to display your group membership.

To display Admintool, type the following command in any window.

```
$ admintool &
```

Network Information Service Plus (NIS+)

NIS+ is the preferred network information service for Solaris networks. Solaris networks can also use NIS either as an alternative to NIS+ or as a supplement to NIS+.

NIS+ is a name service built on top of the ONC transport-independent remote procedure call (RPC) interface. NIS+ has significant benefits compared to NIS in the areas of security, performance, scalability, and administration. Some of the advantages of using NIS+ are:

- NIS+ shares data with the NIS environments, allowing a smooth migration.
- Domains are hierarchical; you can create subdomains.

- You can use the name service switch (`/etc/nsswitch.conf`) to set which name service a system will try to use first: NIS+, NIS, `/etc`, or DNS.
- You can use AdminSuite to make changes to NIS+ tables for adding, modifying, deleting, and searching for information.
- NIS+ enables you to create and maintain an enterprise-wide name service, even across geographically separated sites connected by WAN links.
- You can use the NIS+ backup and restore feature to quickly and easily preserve your name space data set. This feature can also be used to quickly bring additional replica servers on line.

See Chapter 13, in this guide and *NIS+ Transition Guide* and *NFS Administration Guide* for more information.

Print Subsystem

The print management commands have changed between the SunOS release 4.x and Solaris 2.6 operating environments. In the Solaris 2.6 operating environment, you can use command-line procedures or Admintool to set up printers, and you can use administrative commands or the PrintTool to control print jobs.

See Chapter 11, and *System Administration Guide* for more information.

Users can accomplish the same basic tasks using PrintTool or commands in a shell.

PrintTool

PrintTool is a software tool available in the Solaris 2.6 user environment. It provides a graphical user interface within OpenWindows or CDE through which a user can monitor printers and monitor and cancel print jobs.

Command Changes

The following list summarizes command changes:

- `lp`(1) replaces `lpr`
- `lpstat`(1) replaces `lpq`
- `cancel`(1) replaces `lprm`

- `troff`(1) requires a printer name
- `TEX`, `pscat` (`C/A/T`), and raster image filters are not available in the Solaris 2.6 environment

The `lp` service consists of several *daemons*, or processes, that monitor system work, a hierarchy of configuration files in the `/etc/lp` directory, and a set of administrative commands.

Service Access Facility

The Service Access Facility (SAF) is the tool used for administering terminals, modems, and other network devices. In particular, the SAF enables you to:

- Add and administer `ttymon` and `listen` port monitors (using the `sacadm` command)
- Add and administer `ttymon` port monitor services (using the `pmadm` and `ttyadm` commands)
- Add and administer `listen` port monitor services (using the `pmadm` and `nlsadmin` commands)
- Administer and troubleshoot TTY devices
- Administer and troubleshoot incoming network requests for printing service
- Administer and troubleshoot the Service Access Controller (using the `sacadm` command)

The SAF is an open systems solution that controls access to system and network resources through TTY devices and local-area networks (LANs). The SAF offers well-defined interfaces so you can easily add new features and configure existing ones.

The SAF is not a program. It is a hierarchy of background processes and administrative commands. The top-level SAF program is the SAC. The SAC controls port monitors that you administer through the `sacadm` command. Each port monitor can manage one or more ports.

You administer the services associated with ports through the `pmadm` command. While services provided through SAC may differ from network to network, SAC and the administrative programs `sacadm` and `pmadm` are not tailored to network types.

Table 2–2 illustrates the SAF control hierarchy. The `sacadm` command is used to administer the SAC, which controls the `ttymon` and `listen` port monitors.

TABLE 2–2 SAF Functions and Associated Programs

Function	Program	Description
Overall administration	`sacadm`	Command for adding and removing port monitors
Service Access Controller	`sac`	SAF's master program
Port monitors	`ttymon`	Monitors serial port login requests
	`listen`	Monitors requests for network services
Port monitor service administrator	`pmadm`	Controls port monitor services
Services	logins, remote procedure calls, and so on	Services to which SAF provides access

The services of `ttymon` and `listen` are in turn controlled by `pmadm`. One instance of `ttymon` can service multiple ports and one instance of `listen` can provide multiple services on a network interface.

See Chapter 11 for more information.

Volume Management

Beginning with the Solaris 2.2 software, a new layer of software manages CD-ROM and diskette devices: Volume Management. This software automates the interaction between you and your CDs and diskettes.

CDE OpenWindows File Manager has been modified to use Volume Management to provide immediate user access to CDs and diskettes with file systems on them. See *Solaris User's Guide* for more information on File Manager's new features.

There are also several new commands to help you administer Volume Management on your system.

For more information, see "Using Volume Management " on page 61 in Chapter 7.

CHAPTER 3

Converting a SunOS 4.x System to the Solaris 2.6 Environment

Converting a SunOS 4.x system to the Solaris 2.6 environment is a three-phase process that includes pre-installation (backing up data), installing the Solaris environment, and post-installation (restoring data).

This chapter provides information about the pre-installation and post-installation phases for a single system or an entire network. See Chapter 10, for information about creating an environment that serves both Solaris 2.6 and SunOS release 4.x clients.

What's New About Installing

The Solaris 2.6 environment introduces a number of changes in the way you install software on systems, which makes it different than installing the SunOS 4.x software. These include:

- The Solaris 2.6 software is distributed on compact disc (CD) only. This means you must have access to a CD-ROM drive before you can install the software. For systems without local CD-ROM drives, you can set up a system that has a CD-ROM drive to act as an install server on the

network. For more information about network installations, see *Solaris Advanced Installation Guide.*

- The Solaris 2.6 software is bundled into modules called *packages.* You can select which packages to install on your system and control the amount of space each installation requires.

  Also, related packages are grouped into *clusters.* This means that you can select a cluster to install without having to select each package separately.

- Solaris 2.6 installation also provides a set of software groups, which are groups of packages and clusters for typical users (for example, there is an end-user software group). You can select a software group to get systems running without selecting individual packages and clusters. This can be useful when you are first installing the Solaris 2.6 software in a limited environment for testing. You can add or remove packages later as you gain more experience with the system.
- The Solaris 2.6 environment includes architecture-specific kernels rather than the generic kernel configuration provided in earlier SunOS software releases. You will find the installed kernel in `/kernel` instead of `/vmunix`.
- The Solaris 2.6 installation program guides you step-by-step through the installation process.
- The Solaris 2.6 environment provides custom JumpStart™ technology to automate installations. This can save you time when you need to install many systems. For more information, see *Solaris Advanced Installation Guide.*

What to Do Before You Install Solaris Software

Converting a SunOS 4.x system to the Solaris 2.6 software involves more than just running the Solaris installation program and loading the software. Usually, there is data on the SunOS 4.x system that needs to be transferred to a Solaris 2.6 system. This data may be full file systems, such as `/home`, or locally customized system files, such as `/etc/hosts` or `/etc/passwd`.

No matter how you plan to handle the data transfer, you should back up all disk partitions by doing full dumps before you begin the installation process. Because the device naming conventions are different in the Solaris 2.6 operating environment, you might inadvertently choose the wrong disk when you install the Solaris 2.6 software. Backing up the file systems before you begin the installation procedure offers some protection should this occur.

For information about device naming conventions, see “Device Naming Conventions” on page 57.

Note about file system formats:

- If the Solaris 2.6 Extended Fundamental Types (EFT) are not used, the SunOS release 4.x file system format is upwardly compatible with, and in some cases identical to that used in, the software.
- If you are running SunOS 4.1.1 software with QuickCheck or Backup Copilot™ utilities installed or the SunOS 4.1.2 software, the file system formats are identical.
- If you are running SunOS 4.1.1 software without QuickCheck or Backup Copilot utilities, SunOS 4.0.x, or SunOS 4.1 software, the file systems are upwardly and backwardly compatible, although not identical in all cases.

Saving Disk Partition Information

Before you begin the installation process, you should save a hard copy of the system’s existing disk partitions. It can serve as a reference for many decisions that are made about configuring the Solaris 2.6 system. The following procedure is one way to obtain the disk partition information.

1. **Obtain the names of the disks attached to the system.**

   To obtain the names of the disks attached to the system, use the `format`(8) command.

2. **Save the disk partition information.**

   To obtain the partition information encoded on each disk, use the `dkinfo`(8) command. You can pipe the output to a printer or to a file that you can save to another system.

Note - Using the previous command provides you with information only on the configured partitions. All nonconfigured partitions are displayed with the message: “`No such device or address.`”

Saving File System Information

The mappings between file system names (for example, `/usr`, `/home`) and device names (for example, `/dev/sd0g`) reside in the configuration file `/etc/fstab`. Before proceeding, you should make a printed copy of the `/etc/fstab` file to help you construct the Solaris 2.6 file.

Saving Metadevice Configuration Information

Use this section only if you are upgrading a system running the SPARCserver™ Manager or Solstice DiskSuite unbundled products. (These products are used to mirror, concatenate, or stripe multiple disks.)

To upgrade your system without this product, you have to modify your multiple-partition configurations to use single partitions. In particular, a concatenated or striped file system must be reorganized onto a single disk, and partitions and mirrors can no longer be used.

If the system is running SPARCserver Manager or Solstice DiskSuite utilities, you should save the metadevice configuration information before installing Solaris 2.6 software. This enables you to recover the state of the metadevices when you install Solaris 2.6 software, and serves as a reference as you construct the list of disks attached to your system.

1. **Use the `metastat`(8) command to save information, as in the following example.**

```
# /etc/metastat -p | lpr
```

2. **Save the output of the `metadb`(8) command.**

   For example.

```
# /etc/metadb -i | lpr
```

   The output of `metadb` tells you the state database configuration information. This information is necessary to reconstruct the state databases if you reinstall the Solstice DiskSuite product.

Determining What To Back Up

You should create a list of the SunOS 4.x files and file systems that you want to back up and restore after installing Solaris 2.6 software.

Making a List of System Components to Back Up

Make a list of all the system components in the existing SunOS release 4.x environment and decide which are critical to the user's system. Consider:

- Locally developed applications
- Any unbundled software products
- Third-party applications
- Third-party peripheral devices and drivers (8 mm tape drives and SBus cards, for example)

Making a List of Files and File Systems to Back Up

Use the following guidelines to make the list of file systems to save:

- As a general rule, do not transfer file systems containing “system” files (for example, the `/usr` or `/` file systems) in their entirety.
- Do not save temporary file systems, such as `/tmp`.
- Do extract and transfer the data files that have changed locally or those on which the server depends for administrative data, such as some `/etc` files (for example, `/etc/hosts`), exported file systems (use the `exportfs` command to list them), and `/tftpboot` directory, which you should save as a safety precaution.
- Do completely preserve file systems containing only locally generated data, such as `spool` and user home directories.
- Save file systems that contain information about clients if you are migrating a server for SunOS release 4.x clients. Typically, `/export` would be such a file.

Making a List of SunOS System Configuration Files to Back Up

There are a number of SunOS 4.x system configuration files that can be merged or converted for the Solaris platform. Use the example list that follows to help select the system configuration files you want to back up.

Note - The list contains suggestions. You should study the items carefully and add to or delete from the list depending on the configuration at your site. For example, if you have special files in directories from third-party software vendors, you may need to save them.

If the system is an NIS master server, you should save all the files that reside in the NIS master directory (for example, `/etc`). Additionally, save any other master files that you added to NIS. The suggestions for files to back up include:

- `./.cshrc`
- `./.profile`
- `./.login`
- `./.logout`
- `./.rhosts`
- `./etc` (if the system is an NIS client or has no name service)
- `./var/spool/calendar`
- `./var/spool/cron`

- `./var/spool/uucp`
- `./var/nis` (if the system is an NIS master server)
- Boot programs in `./tftpboot`

Determining Disk Space Requirements

Make a list of how much disk space each file system that you want to move to the Solaris 2.6 upgrade, uses. Refer to this list when installing the Solaris 2.6 software, since you can partition disk space for your SunOS 4.x file systems when running the Solaris 2.6 installation program.

Deciding the Order of Installation for Networks

If you are converting a network of SunOS 4.x systems to the Solaris 2.6 software, decide the order of the systems to convert to maximize convenience for the users. For example, you might want to convert all client systems before you convert any servers. The first system you convert should be a standalone system with a locally attached CD-ROM drive.

For a while, you will probably manage a network consisting of both SunOS 4.x and Solaris 2.6 systems, and part of your planning should involve determining priorities. For example, you may want to convert one domain and use it for system administration testing and for porting internally developed applications before you convert the entire network environment.

Backing Up Files and File Systems Before You Install

Once you decide which files or file systems you need to back up from the SunOS 4.x system, you can use the standard commands and procedures given in the SunOS 4.x documentation to do backups. The command you use depends on whether the tape drive is local or remote. No matter how you plan to handle the data transfer, it is still a good idea to back up all disk partitions by doing full dumps before you begin the installation process.

Installing Solaris Software

Install the Solaris 2.6 software on the server or standalone system using the software installation procedures given in *Installation Instructions for Solaris 2.6 (SPARC Platform Edition)* or *Installation Instructions for Solaris 2.6 (Intel Platform Edition).*

Preserve Option

The Solaris 2.6 installation program has a preserve screen that enables you to preserve existing file systems during installation. This is a good way to preserve any SunOS 4.x file systems so you don't have to restore them.

If you cannot preserve a SunOS 4.x file system or you choose not to (because you want to change how the system's disks are partitioned), you should create new file systems with sufficient disk space for the SunOS 4.x file system that you want to restore (using the disk space requirements you recorded earlier). Then you can restore the SunOS 4.x file systems into the new file systems after Solaris is installed.

Restoring Files and File Systems After You Install

This section describes issues related to restoring SunOS 4.x files and file systems you backed up before installing the Solaris 2.6 software.

Restoring SunOS 4.x File Systems and User Files

You can restore the SunOS 4.x file systems that you could not or chose not to preserve into the new file systems you created during the Solaris 2.6 installation. For information about backup and restore procedures, see *System Administration Guide.*

Note - Before proceeding make sure that the target slice is large enough to accommodate the file system being restored.

Restore any SunOS 4.x user files that you backed up, and copy them to the new system.

Restoring SunOS 4.x System Configuration Files

First, you must restore the SunOS 4.x system configuration files to a temporary directory on the Solaris 2.6 system. After the information is back on the system in the temporary directory, you need to make it available in the Solaris 2.6 operating environment. Some of the data can just be merged into the files, while some types of data must be converted to new formats.

The system's configuration defines which files you need to work with. Complete the restore by merging or converting files as follows:

- Systems with no name service: If the system has no name service, merge or convert all the relevant system files located in `/etc` and `/var`.
- Systems that are NIS clients: If the system is an NIS client, merge or convert only the local system configuration files located in `/etc` and `/var` that are not provided via the NIS name service.
- Systems that are NIS master servers: If the system is an NIS master server, merge or convert all the files that reside in the NIS master directory (for example, `/etc`). Additionally, update other local configuration files in `/etc` and `/var`.

Files to Merge

To make data from any of the following files available, merge the changes into the Solaris 2.6 version of the same file. Note, however, that not all of these files were modified on the SunOS 4.x system. Identify files that were changed on the SunOS 4.x system and merge these only. As you read the list, note that some of the file names are slightly different. For example, `/etc/auto.*` files are now `/etc/auto_*`.

The following is an example list of the SunOS release 4.x files backed up using the instructions in the first part of this chapter. These files are candidates for merging into the Solaris 2.6 operating environment. See Appendix D, to examine SunOS release 4.x files for changes.

- All automounter maps, including `/etc/auto.master` and any others
- `/etc/aliases`
- `/etc/bootparams`
- `/etc/ethers`
- `/etc/hosts`

- /etc/format.dat
- /etc/inetd.conf
- /etc/netmasks
- /etc/networks
- /etc/protocols
- /etc/publickey
- /etc/rpc
- /etc/services
- /etc/hosts.equiv
- /etc/remote
- /.cshrc
- /.profile
- /.login
- /.logout
- /.rhosts
- /var/spool/cron
- /var/spool/mail
- /var/spool/calendar
- /var/spool/uucp

Files to Convert

Many system files, such as the /etc/fstab file, have been replaced and do not exist under the Solaris 2.6 operating environment. Information from these files must be extracted and manually converted in the Solaris 2.6 environment. See Appendix D, to examine SunOS release 4.x files for changes.

Caution - Do not restore operating system executable files (such as system commands in /usr/bin) from the SunOS release 4.x system to your system after installing the Solaris 2.6 software.

You must change the following files before merging the data onto the Solaris 2.6 system:

- /etc/uucp – There have been some changes to the UUCP system. The Config, Grades, and Limits files are new in the Solaris 2.6 operating environment. The files Devconfig, Devices, Dialcodes, Dialers,

`Permissions`, `Poll`, `Sysfiles`, and `systems` are the same in the Solaris 2.6 operating environment as they were in the SunOS release 4.x software. These files can be merged together. There are also several SunOS release 4.x files that are not used in the Solaris 2.6 operating environment.

- `/etc/group` – The basic format of this file is the same as it was in the SunOS 4.1 and SunOS 4.1.x releases. However, previous releases used a group entry beginning with a plus sign (+) or minus sign (-) to selectively incorporate entries from NIS maps for group. See the `group`(4) man page if that compatibility is needed under the Solaris 2.6 operating environment.
- `/etc/netgroup` – There is no `/etc/netgroup` file in the SunOS release 5.6 environment.
- `/etc/exports` – File systems to be shared on the network under the Solaris 2.6 operating environment use the `/etc/dfs/dfstab` file instead of `/etc/exports`. The format of entries in this file follows.

```
share –F fstype –o options –d "text" pathname resource
```

See the `dfstab`(4) man page for additional information.

- `/etc/fstab` - File systems to be mounted under the Solaris 2.6 operating environment use the `/etc/vfstab` file instead of `/etc/fstab`. The format of entries in the `/etc/vfstab` file follows.

```
dev raw_dev mnt_pt fs_type
fsck_pass auto_mnt mnt_option
```

Refer to the `vfstab`(4) man page for additional information.

- `/etc/passwd` – The format of the `passwd` file is the same as that under the SunOS release 4.x software. However, user passwords are now stored in the `/etc/shadow` file. Refer to the `passwd`(4) and `shadow`(4) man pages for additional information.
- `/etc/sendmail.cf` – The format of `sendmail.cf` is the same as that under the SunOS release 4.x structure. The location of the file is now `/etc/mail/sendmail.cf`.
- `/etc/ttytab` – Under the SunOS release 4.x system, `ttytab` was used to control serial ports and the characteristics of the terminals on those serial lines. Under the Solaris 2.6 operating environment, the Service Access Facility is used to configure this capability.
- `/etc/printcap` – Under the Solaris 2.6 operating environment, printers are configured using the SunOS release 5.6 LP print service. See *System Administration Guide* for additional information.

CHAPTER 4

Using the Compatibility Packages

The SunOS release 5.6 software is neither source nor binary compatible with the SunOS release 4.x software. This means that SunOS release 4.x programs and user applications based on those releases may not run correctly under the Solaris 2.6 operating environment. Compatibility packages make it possible for these programs to run on a Solaris 2.6 system.

This chapter briefly discusses two compatibility packages: the SunOS/BSD Source Compatibility Package and the Binary Compatibility Package. These packages make the transition easier by enabling you to use SunOS release 4.x commands and applications while your environment and applications migrate to the Solaris 2.6 operating environment.

Some SunOS release 4.x commands are not available in the Solaris 2.6 operating environment. Others exist, but have changed. For information about changes to SunOS release 4.x commands in the Solaris 2.6 operating environment, see Appendix A.

Why Port Applications?

Although the SunOS Binary Compatibility Package and the SunOS/BSD Source Compatibility Package allow you to use applications as they are, you should port applications as soon as possible. Long-term reliance on the compatibility packages is not advised for the following reasons:

- The application's performance is reduced.

- You will not be able to take advantage of the Solaris 2.6 operating environment's increased range of operations and portability.
- Compatibility packages are temporary aids to help sites through the transition.

SunOS/BSD Source Compatibility Package

The SunOS BSD/Source Compatibility Package is an optional package available with the Solaris 2.6 operating environment. The package contains a collection of SunOS release 4.x and BSD commands, library routines, and header files otherwise not available with the Solaris 2.6 operating environment. The Binary Compatibility Package must be installed in order to use the SunOS/BSD Source Compatibility Package.

The interfaces in the SunOS/BSD Source Compatibility Package are installed in the `/usr/ucb` directory, thereby avoiding conflicts with existing SunOS release 5.6 interfaces. These interfaces provide a familiar SunOS environment while your environment and applications are migrating to the SunOS release 5.6 software. To use these interfaces, you must either specify the full path name or modify your `PATH` environment variable. When modifying your `PATH` environment variable, note that `/usr/ucb` should precede `/usr/bin`.

For detailed information about the Source Compatibility Package, see *Source Compatibility Guide*

Binary Compatibility Package

The Binary Compatibility Package is an optional package available with the Solaris 2.6 operating environment. The package allows existing SunOS release 4.x applications, both statically and dynamically linked, to run under the Solaris 2.6 operating environment without modification or recompilation. It handles most binary interface discrepancies between the two releases transparently. This results in a Solaris 2.6 operating environment where SunOS release 4.x applications can run properly.

See *Binary Compatibility Guide* for procedures about setting up your environment to access this package. This guide also details the limitations of the Binary Compatibility Package.

Using the Binary Compatibility Package to Run SunOS Release 4.x Applications

The Binary Compatibility Package allows most applications to run under the Solaris 2.6 operating environment, making them available for use before they are ported to SunOS release 5.6. With this package, well-behaved application binaries based on SunOS release 4.x system software will run under the SunOS release 5.6 software without modifications or recompilation.

The Binary Compatibility Package is intended for end-user environments, not for use as a development environment. All SunOS release 5.6 application development should be done under the base SunOS release 5.6 environment.

CHAPTER 5

Security

Security for the Solaris 2.6 operating environment combines several features from SunOS release 4.x and AT&T SVR4 with capabilities added specifically for the new environment. There are also changes in the packaging of some SunOS release 4.x security programs.

This chapter describes major differences between SunOS release 4.x and Solaris 2.6 operating environment security, and points out how those changes may affect system administration procedures. *System Administration Guide* describes the administration and use of these features more fully.

Solaris 2.6 Security Features

Most of the security features from SunOS release 4.x systems are also available in the Solaris 2.6 operating environment. These include:

- Internet security
- `.rhosts` and `.rhosts.equiv` files
- Secure RPC and NFS

NFS Administration Guide documents secure NFS and the `.rhosts` files. *TCP/IP and Data Communications Administration Guide* describes administering Internet security.

Security for local SunOS release 5.6 systems includes storing encrypted passwords in a separate file, controlling login defaults, and restricted shells. Equivalent NIS+ security, described in *NIS+ Transition Guide* and *NFS Administration Guide*, controls network-wide access to systems.

The subsections below summarize security features under local system control.

/etc/passwd and /etc/shadow Files

The SunOS release 5.6 `passwd` command stores encrypted versions of passwords in a separate file, `/etc/shadow`, and allows only root access to it. This prevents general access to the encrypted passwords that formerly appeared in the `/etc/passwd` file, which anyone could read.

The `/etc/shadow` file also includes entries that force password aging for individual user login accounts. The mechanism for changing entries to the `passwd` and `shadow` files is described in *System Administration Guide*.

/etc/default Files

Several files that control default system access are stored in the `/etc/default` directory. These files limit access to specific systems on a network. Table 5–1 summarizes the files in the `/etc/default` directory.

TABLE 5–1 Files in `/etc/default` Directory

`/etc/default/login`	Controls system login policies, including root access. The default is to limit root access to the console.
`/etc/default/passwd`	Controls default policy on password aging
`/etc/default/su`	Controls which root (`su`) access to the system will be logged and where it will be displayed

Restricted Shells

System administrators can use restricted versions of the Korn shell (`rksh`) and Bourne shell (`rsh`) to limit the operations allowed for a particular user account.

Restricted shells do not allow the following operations:

- Changing directories
- Setting the $PATH variable
- Specifying path or command names beginning with "/"
- Redirecting output

See the `ksh` and `sh` man pages for a description of these shells.

Note that the restricted shell and the remote shell have the same command name (`rsh`) with different path names:

- `/usr/lib/rsh` is the restricted shell
- `/usr/bin/rsh` is the remote shell

Password Aging Changes

The SunOS release 5.6 system features password aging. This feature assigns a limited lifetime to each user password to maintain password secrecy. As a password reaches the end of its life, the password owner is notified and prompted to select a new one.

You can implement password aging using one of the following methods:

- *Method 1* – Use Admintool to manage users if you are running an X-window environment. For information about this method, see *Solaris Advanced User's Guide.*
- *Method 2* – Use new `passwd` or `nispasswd` command options (depending on which name service stores the account).

A system administrator can also set up password aging.

You can change a user password in one of two ways:

- *Method 1*– Use either `passwd` or `nispasswd`, depending on which name service is used to store your account.
- *Method 2* – Use Admintool to manage users if you are running an X-window environment. For information about this method, see *Solaris Advanced User's Guide.*

For more information on `passwd` and `nispasswd`, see the command tables in Appendix D.

Access Control Lists (ACLs)

Access control lists (ACLs), supported in both UFS and NFS, provide greater flexibility in managing file permissions than traditional UNIX file protection. The traditional UNIX file protection provides read, write, and execute permissions for three user classes: owner, group, and other.

Using ACLs allows you to define file permissions for the owner, owner's group, others, specific users and groups, and default permissions for each of those categories. For example, you can set up an ACL that defines read permission to a group of users and write permission to only one user in the group. You could not do this with standard UNIX file permissions.

The `setfacl`(1) command sets, adds, modifies, and deletes ACL entries, and the `getfacl`(1) command displays ACL entries.

See *System Administration Guide* for more information about using ACLs.

Automated Security Enhancement Tool (ASET)

The Automated Security Enhancement Tool (ASET), available as a separate option with SunOS release 4.x systems, is included with the Solaris 2.6 operating environment. ASET enables you to specify an overall system security level (low, medium, or high) and automatically maintain systems at those levels. This tool can be set up to run on a server and all its clients or on individual clients.

ASET performs these tasks:

- Verifies system file permissions
- Verifies system file contents
- Checks integrity of group file entries
- Checks system configuration files
- Checks environment files (`.profile`,`.login`, and `.cshrc`)
- Verifies EEPROM settings to restrict console login access
- Allows establishment of a firewall or gateway system

System Administration Guide describes ASET setup and monitoring in detail.

Security Options

Currently available bundled security options are Kerberos security, the SunSHIELD™ package, and Pluggable Authentication Module (PAM).

Kerberos 4.0 Security

The Solaris 2.6 operating environment includes support for Kerberos V4 authentication for secure RPC. (Kerberos source code and administrative utilities are available from MIT.) Included in this release are:

- Client applications library that can use Kerberos
- Kerberos option to Secure RPC
- Sun's NFS™ distributed computing file system application with Kerberos
- Commands to administer user tickets on the client

System Administration Guide describes the client-side utilities. included in the release. *NFS Administration Guide* describes the use of Kerberos with the NFS application.

SunSHIELD Package

The Solaris 2.6 release includes the SunSHIELD Basic Security Module (BSM) package. This product provides the security features defined as C2 in the Trusted Computer System Evaluation Criteria (TCSEC). The features provided by the BSM are a security auditing subsystem and a device allocation mechanism. C2 discretionary access control and identification and authentication features are provided in the operating system.

The administration of BSM is included in *SunSHIELD Basic Security Module Guide*.

PAM

The Pluggable Authentication Module (PAM) framework enables new authentication technologies to be "plugged-in" without changing commands, such as `login`, `ftp`, `telnet` and so on. The framework enables a system administrator to choose any combination of services to provide authentication. Mechanisms for account, session, and password management can also be "plugged-in" using this framework.

System Administration Guide describes the administration of PAM.

CHAPTER 6

User Environment Administration

This chapter describes differences in tasks you may perform to set up the local user environment after installing the Solaris 2.6 software.

Selecting a Default Shell

The login shell is the command interpreter that runs when you are logged in. The Solaris 2.6 operating environment offers three shells:

- Bourne shell, the default shell (`/bin/sh`)
- C shell (`/bin/csh`)
- Korn shell (`/bin/ksh`)

If you use the shell often, you may prefer to use the C shell or the Korn shell because of their interactive capabilities. lists the features of all three shells.

TABLE 6–1 Basic Features of the Bourne, C, and Korn Shells

Feature	Bourne	C	Korn
Syntax compatible with `sh`	Yes	No	Yes
Job control	Yes	Yes	Yes
History list	No	Yes	Yes
Command-line editing	No	Yes	Yes
Aliases	No	Yes	Yes
Single-character abbreviation for login directory	No	Yes	Yes
Protect files from overwriting (noclobber)	No	Yes	Yes
Ignore Control-D (`ignoreeof`)	No	Yes	Yes
Enhanced `cd`	No	Yes	Yes
Initialization file separate from `.profile`	No	Yes	Yes
Logout file	No	Yes	No

To change from one shell to another, use one of the following methods:

- *Method 1* – Edit the information in the last field of the line in the `/etc/passwd` file that begins with your login name. If this entry is blank or `sh`, the login shell is the Bourne shell. If the entry is `csh`, the login shell is the C shell. If the entry is `ksh`, the login shell is the Korn shell.
- *Method 2* – In a windows environment, use Admintool. See *Solaris Advanced User's Guide* for information.

After you change to a new shell, log out and log in again to activate the shell.

Customizing User Environments

This section describes how to determine which initialization files you can edit to customize the local environment based on your choice of login shell, and where to find them in the SunOS release 5.6 file systems. Set up your environment by editing the variables in the initialization files. The default shell determines which files you need to edit: `.profile`, `.login`, or `.cshrc`. Table 6–2 shows the initialization files for the Bourne, C, and Korn shells.

TABLE 6–2 Initialization Files for Bourne, C, and Korn Shells

Shell	Initialization File	Purpose
Bourne	`/etc/profile`	Defines system profile at login
	`$HOME/.profile`	Defines user's profile at login
C	`/etc/.login`	Defines system environment at login
	`$HOME/.cshrc`	Defines user's environment at login
	`$HOME/.login`	Defines user's profile at login
Korn	`/etc/profile`	Defines system profile at login
	`$HOME/.profile`	Defines user's profile at login
	`$HOME/`*ksh_env*	Defines user's environment at login in the file specified by the *ksh_env* variable

In this release, the shell initialization-file templates have moved to the `/etc/skel` directory from `/usr/lib`, where they were in the SunOS release 4.x software. The template file locations are shown in . Copy the template file (or files) for the appropriate default shell to your home directory before you modify it.

TABLE 6–3 Default Home Directory Startup Files

Shell	File
Bourne	/etc/skel/local.profile
C	/etc/skel/local.login
	/etc/skel/local.cshrc
Korn	/etc/skel/local.profile

For information on setting up initialization files, see *System Administration Guide*.

Using the SunOS 4.x Work Environment With the Solaris Software

The SunOS release 5.6 software can use the old SunOS release 4.x system files and initialization files such as `.login`, `.cshrc`, and `.profile` to re-create the look and feel of the SunOS release 4.x work environment. Many of these SunOS release 4.x files can be converted, or used as they are, and executed easily.

The installation process in Chapter 3, explains how to re-create the SunOS release 4.x environment within the Solaris 2.6 operating environment.

Window Systems

The Common Desktop Environment (CDE) is the default Solaris 2.6 windowing environment and offers a simple and intuitive interface. See Chapter 14, for more information about CDE.

The OpenWindows 3.1 software can also be used as your preferred desktop with the Solaris 2.6 environment. If you have been using the OpenWindows 2.0 environment, you will notice that the OpenWindows 3.1 icons have changed and some applications are not compatible with the OpenWindows 3.1 platform.

The OpenWindows Developer's Guide File Chooser (`gfm`) regular-expression file-pattern matching code (`filter_pat`) is slightly different from the regular-expression file-pattern matching code in the XView™ File Chooser object. This could result in the same regular expression matching slightly different sets of files in the two different choosers. The XView File Chooser uses `/usr/include/reexp.h` in the SunOS release 5.6 software and its usage is correct.

SunView™ software is not part of the Solaris 2.6 operating environment. SunView applications are incompatible with the OpenWindows environment and must be converted.

See *OpenWindows Version 3.1 User's Guide* for information about:

- Features of the OpenWindows 3.1 environment
- The applications that are not compatible between OpenWindows Version 2.0 and 3.1 platforms
- Guidelines for modifying incompatible applications

User and Group Administration

This section describes your options for performing user and group administration.

User and Group Administration Choices

You can add, modify, and remove users and groups through the command-line interface using `useradd`, `userdel`, and `usermod`. Although these commands are not as robust as Admintool, they do enable you to do most of the tasks supported by Admintool from the command line without running the OpenWindows or CDE software.

The `useradd`, `userdel`, and `usermod` commands are similar to editing the `/etc` files in that they also affect only the local system. These commands cannot be used to change any information in the network naming service. However, you can use `useradd` to verify the uniqueness of the user name and user ID and the existence of group names in the network naming service.

Adding User Accounts

This section describes changes to the general procedure for adding user accounts.

The general procedure for adding new users to a SunOS release 4.x system was:

1. Edit the `/etc/passwd` file and add an entry for the new user.
2. Create a home directory and set the permissions for the new user.
3. Set up skeletal files for the new user (`.cshrc`, `.login`, `.profile`, and so on).
4. Add the new user to the naming service (NIS).

In the Solaris 2.6 operating environment, there are three ways to add (and maintain) user accounts:

- Use Admintool – This is the most straightforward method to use if the system is running the OpenWindows environment.
- Use command-line interfaces (`useradd`, `usermod`, and `userdel`) – Use this method if you don't want to use Admintool.
- Manually edit files (similar to the SunOS release 4.x procedure with a few exceptions)

Note - Because the SunOS release 5.6 software uses a shadow password file, simply editing the `/etc/passwd` file is no longer sufficient. You should not attempt this method unless you have ample experience with this type of administration.

System Administration Guide describes in detail the policy decisions you should consider before you begin to set up accounts. It also explains security considerations for controlling user access to systems and networks.

Using Mail

The SunOS release 4.x mail programs are different in the Solaris 2.6 operating environment; however, procedures for setting up mail are still the same. The SunOS release 4.x version of `mail` is included in the SunOS/BSD Source Compatibility Package. Its user interface is different from the Solaris 2.6 operating environment's version of `mail`. Additionally, some useful mail facilities are included for compatibility.

In the Solaris 2.6 operating environment, there are three programs for sending and retrieving your mail. All three are backward compatible and can be used to read your SunOS release 4.x mail. They are:

- `mailtool`, the OpenWindows interface for the mail program. New Solaris 2.6 `mailtool` options enable you to attach files to your messages, include third-party messages with your mail, deliver mail to multiple recipients, and send audio messages.

See *OpenWindows Version 3.1 User's Guide* for a complete discussion of `mailtool`.

- `mailx`, which is installed under `/usr/bin/mailx`. This is the Solaris 2.6 mail reading program. It is an enhanced version of SunOS release 4.x `/usr/ucb/mail`. In the Solaris 2.6 operating environment, `/usr/ucb/mail` is a link to `/usr/bin/mailx`. `mailx` offers message headers that enable you to preview the sender and subject of each message before you read them. You can also switch between reading, sending, and editing mail messages.

  See the `mailx`(1) man page for more information on `mailx`.

- `mail` refers to the mail program under `/usr/bin/mail`. The Solaris 2.6 interface is similar to the SunOS release 4.x `/usr/bin/mail` version (see the `bin-mail`(1) manual page in *SunOS 4.x Reference Manual*).

  See the `mail`(1) man page for more information on `mail`.

For a complete discussion of all Solaris 2.6 mail programs, see *Mail Administration Guide.*

Using Document Tools

This section outlines the main differences in using document tools between SunOS release 4.x and the Solaris 2.6 operating environment.

- The Solaris 2.6 operating environment provides a set of PostScript filters and device-independent fonts. However, most SunOS release 4.x TranScript filters have SunOS release 5.6 equivalents while a few less common ones do not. In SunOS release 5.6 systems, there is no T_EX filter, no `pscat` (C/A/T) filter, and no raster image filter.
- The Solaris 2.6 operating environment provides device-independent `troff`, with the following features: SunOS release 4.x `troff` input files work with Solaris 2.6 `troff`; `troff` default output goes to the standard output instead of the printer. Therefore, you must specify a printer when you send `troff` output to the printer.

Man Page Organization Differences

Man page organization has changed to be compatible with SVR4 organization. As a result, some sections have been renamed. For example, `man`(8) is now `man`(1M).

Table 6–4 shows SunOS release 5.6 man page directories.

TABLE 6–4 SunOS release 5.6 man Page Directories

/man Directory	Contents	Suffixes
man1	User commands	1B - SunOS/BSD compatibility commands
		1C - Communication commands
		1F - FMLI commands
		1S - SunOS commands
man1M	System administration commands	
man2	System calls	
man3	Library functions	3B - SunOS/BSD compatibility libraries
		3C - C library functions
		3E - ELF library functions
		3G - C library functions
		3I - Wide Character functions
		3K - Kernel VM library functions
		3M - Math library
		3N - Network functions
		3R - RPC services library
		3S - Standard I/O functions
		3T - Threads library functions
		3X - Miscellaneous library functions
man4	File formats	4B - SunOS/BSD compatibility file formats

TABLE 6–4 SunOS release 5.6 man Page Directories *(continued)*

/man Directory	Contents	Suffixes
man5	Headers, tables, and macros	
man7	Special files	
man9	DDI/DKI	
man9E	DDI/DKI entry points	
man9F	DDI/DKI kernel functions	
man9S	DDI/DKI data structures	

Customizing the man Command Search Path

Unlike in the SunOS release 4.x software, which searched the individual man directories according to a predetermined order, the SunOS release 5.6 software lets you determine the search path. The man command uses the path set in the man page configuration file, man.cf.

Each component of the MANPATH environment variable can contain a different man.cf file. You can modify man.cf to change the order of the search; for example, to search 3b before 3c. The configuration file for the /usr/share/man directory follows.

```
#
# Default configuration file for the on-line manual pages.
#

MANSECTS=1,1m,1c,1f,1s,1b,2,3,3c,3s,3x,3i,3t,3r,3n,3m,3k,3g, \
3e,3b,9f,9s,9e,9,4,5,7,4b,6,l,n
```

The arguments to MANSECTS are derived from the man subdirectories available. The number of subdirectories has increased dramatically in this release because each subsection has its own directory. This new structure improves the performance of the man command and gives you finer control

over the search path. The next two figures compare the man directories for the two releases.

```
sunos4.1% ls /usr/share/man
man1/  man2/  man3/  man4/  man5/  man6/  man7/  man8/
manl/  mann/
```

```
sunos5.6% ls /usr/share/man
man.cf  man1f/  man3/  man3g/  man3n/  man3x/  man6/  man9f/
man1/  man1m/  man3b/  man3i/  man3r/  man4/  man7/  man9s/
man1b/  man1s/  man3c/  man3k/  man3s/  man4b/  man9/  manl/
man1c/  man2/  man3e/  man3m/  man3t/  man5/  man9e/  mann/
```

`whatis` and `windex` Databases

The SunOS release 4.x man page table of contents and keyword database is called `whatis`. In the SunOS release 5.6 software, this information is in the `windex` file. In both releases, the database is created by the `catman` command, and is used by the `man`, `apropos`, and `whatis` commands.

The `windex` file also has a slightly different format than the `whatis` file, as you can see from the following comparison of the two release versions.

```
sunos4.1% man -k tset
tset, reset (1)   - establish or restore terminal characteristics
```

```
sunos5.6% man -k tset
reset  tset (1b)  - establish or restore terminal characteristics
tset  tset (1b)  - establish or restore terminal characteristics
```

Using the `man` Command

Table 6–5 shows that SunOS release 5.6 version of the `man` command has additional search options.

TABLE 6–5 New man Command Options

Option	Description
–a	Displays all man pages that match *file name*. The pages are displayed sequentially in the order they are found.
–l	Lists all man pages that match file name. You can use the output of this command to specify a section number with the –s option.
–s *section-number*	Searches *section-number* for *file name*. In the SunOS release 4.x software, the man command accepted the section number as an option; in this release, the section number must be preceded by –s.
–F	Forces the man command to search all directories until *file name* is found. This option overrides the windex database and the man.cf file.

See the man(1) man page for a complete description of the SunOS release 5.6 man command.

CHAPTER 7

Device Administration

This chapter explains SunOS release 5.6 device naming conventions and discusses changes to device-related tasks such as getting information about disks, adding devices to a system, and using Volume Management.

Device Naming Conventions

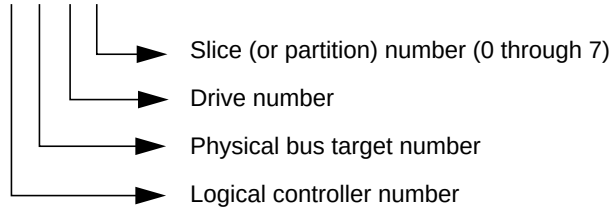
Device naming conventions have changed between the SunOS release 4.x and SunOS release 5.6 platforms. In addition, the `/dev` directory, which contains the special device names, has been changed from a flat directory to a hierarchical one, with a separate subdirectory for each category of device. For example, the location of disk device files is `/dev/dsk`, while raw disks are located in `/dev/rdsk`.

SunOS release 5.6 commands that take device names as arguments must use the SunOS release 5.6 device naming conventions. However, you can still use and recognize the SunOS release 4.x device names if you install the SunOS/BSD Source Compatibility Package. See *Source Compatibility Guide* for additional information.

Convention for Disks

The disk partition slice numbers (0 through 7) correspond to partitions a through h of previous SunOS releases.

```
/dev/dsk/cWtXdYsZ
/dev/rdsk/ cWtXdYsZ
```



Note - Most SCSI disks have embedded controllers. This means that the drive number will always be 0 but the target number varies. For example, if an external disk drive has its rear switch set to 2, the device name for the first slice is `/dev/dsk/c0t2d0s0`, not `/dev/dsk/c0t0d2s0`.

Because the names for SCSI targets 0 and 3 were reversed on some sun4c systems, device naming can be confusing. Under the SunOS 4.1.x software, SCSI target 3 was called `sd0`, but it is now properly named `c0t3d0`. SCSI target 0 was called `sd3`, but it is now named `c0t0d0`. Other SCSI disk names translate normally. For example, in the SunOS release 5.6 software, `sd2a` is `c0t2d0s0` and `sd2b` is `c0t2d0s1`.

Convention for Tape Drives

```
/dev/rmt/X A n
```

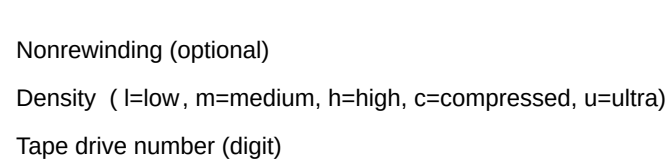


Table 7–1 provides some examples that compare the SunOS release 4.x and SunOS release 5.6 device naming conventions.

TABLE 7–1 SunOS release 4.x and SunOS release 5.6 Device Names

Device Description	SunOS release 4.x Device Name	SunOS release 5.6 Device Name
Disk devices	/dev/sd0g	/dev/dsk/c0t3d0s6
	/dev/rsd3b	/dev/rdsk/c0t0d0s1
	/dev/rsd3a	/dev/rdsk/c0t0d0s0
Magnetic tape devices	/dev/nrmt8	/dev/rmt/8hn
	/dev/rst0	/dev/rmt/0
CD-ROM device	/dev/sr0	/dev/dsk/c0t6d0s2

Obtaining Disk Information

The commands that report disk information in the SunOS release 5.6 software have changed. `df`(1M) and `du`(1M) are still available, but have changed. `dkinfo`(8), and `devinfo`(1M) are replaced by `prtvtoc` and `sysdef –d`. This section provides an overview of those changes.

If you have installed the compatibility packages, SunOS release 4.x command versions can be found under `/usr/ucb/df` and `/usr/ucb/du`.

`df` Command

The `df` command has been changed to support the VFS architecture. As with the other VFS commands, there are generic and file-system versions of the command. The syntax in the SunOS release 5.6 command differs significantly from that used in the SunOS release 4.x version (see Appendix A, for more information). For more information on VFS, see “Virtual File System Architecture” on page 77.

The `df` command now reports disk space in 512-byte blocks instead of kilobytes, but the `–k` option can be used to report disk space in kilobytes. Also, the `–t` option behaves differently; formerly, it restricted the output to

file systems of a specified type (for example, “nfs” or “4.2”). The SunOS release 5.6 version produces a full listing with totals.

Finally, use the SunOS release 5.6 device naming conventions when specifying special device names to this command. See “Device Naming Conventions” on page 57 for details.

du Command

Like `df`, the `du` command reports disk usage in 512-byte blocks instead of kilobytes. There’s also a `–r` option that causes the normally “silent” command to generate messages when it has difficulty reading a directory or opening a file.

dkinfo Command

The SunOS release 4.x `dkinfo` command is no longer available. To print device information, use `prtvtoc`(1M) instead of `dkinfo`.

The `prtvtoc` command reports the important information stored on a disk’s label, including information on the disk’s partitions. For more information about `prtvtoc`, see *System Administration Guide*.

The following screen shows sample output for the SunOS release 5.6 `prtvtoc` command.

```
# prtvtoc /dev/rdsk/c0t2d0s2
* /dev/rdsk/c0t2d0s2 partition map
*
* Dimensions:
*     512 bytes/sector
*      36 sectors/track
*       9 tracks/cylinder
*     324 sectors/cylinder
*    1272 cylinders
*    1254 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First      Sector   Last
* Partition  Tag  Flags   Sector     Count    Sector   Mount
Directory
     0       0    00          0       32724    32723   /
     1       0    00      32724       65448    98171
```

(Continued)

(Continuation)

```
    2       0    00          0      406296   406295
    6       0    00      98172      308124   406295   /usr
```

`devinfo` Command

The SunOS release 4.x version of `devinfo` is incompatible with the SunOS release 5.6 version. To produce output similar to the SunOS release 4.x version, use `prtconf` with the `–v` option.

Adding Devices to the System

At boot time, the system does a self-test and checks for all devices that are attached to it. After you add a new device to the system, use `boot –r` to activate dynamic reconfiguration of the kernel. A reconfiguration script is run to load all the device drivers listed in the module's directories and to create the corresponding hardware nodes. See the `kernel`(1M) man page for more information.

You can also use `boot –a` to interactively add drivers or modules to the system, but if you do, you will be asked to provide other boot parameters, including what to boot and where the root file system is.

Paths to the system files and kernel modules are stored in `/etc/system`. When the system boots, it reads the information in `/etc/system` to determine which modules to load. You can specify a different path by using the `MODDIR` syntax of the `system`(4) file or by using `boot –a`.

For more information about `boot`(1m) or about adding devices and drivers, see *System Administration Guide.*

Using Volume Management

Beginning with the Solaris 2.2 software, a new layer of software manages CD-ROM and diskette devices — Volume Management. This software automates the interaction between you and your CDs and diskettes.

The OpenWindows and CDE File Manager applications have been modified to use Volume Management to provide immediate user access to CDs and diskettes with file systems. See *Solaris User's Guide* for more information on File Manager's new features.

There are also several new commands to help you administer Volume Management on your system.

Volume Management automatically mounts CD and diskette file systems when removable media are inserted into the devices. Any CD or diskette file system will be automatically mounted in the locations described in Table 7–2.

TABLE 7–2 Location of CD-ROM and Diskette With a File System

Media	Location
CD	`/cdrom/`*cdrom_name*
Diskette	`/floppy/`*floppy_name*

If the CD or diskette does not contain a file system, it will be accessible in the locations described in Table 7–3.

TABLE 7–3 Location of CD-ROM and Diskette Without a File System

Media	Location
CD	`/vol/dev/aliases/cdrom0`
Diskette	`/vol/dev/aliases/floppy0`

For security reasons, these file systems are mounted `setuid`. See the `mount`(1M) man page for a description of this and other mount options.

For more information on configuring Volume Management and on using diskettes and CDs, see *System Administration Guide.*

Man pages for Volume Management components are also available, See `rmmount`(1), `rmmount.conf`(4), `volcancel`(1), `volcheck`(1), `vold`(1M), `volmgt`(3), `vold.conf`(4), `volfs`(7), and `volmissing`(1).

Note - Volume Management now controls these CD-ROM paths: `/dev/dsk/c0t6d0s0` and `/dev/rdsk/c0t6d0s0`; and these diskette paths: `/dev/diskette` and `/dev/rdiskette`. Attempts to mount or access a CD or diskette using these paths will result in an error message.

There are several new commands to help you administer Volume Management on your system, as described in Table 7–4.

TABLE 7–4 Volume Management Commands

Command	Description
`rmmount`(1)	Removable media mounter. Used by `vold` to automatically mount `/cdrom` and `/floppy` when a CD or diskette is installed.
`volcancel`(1)	Cancels a user's request to access a particular CD-ROM or diskette file system
`volcheck`(1)	Checks drive for installed media. By default, checks drive pointed to by `/dev/diskette`.
`volmissing`(1)	Notifies user when an attempt is made to access a CD or diskette that is no longer in the drive
`vold`(1)	Volume Management daemon, controlled by `/etc/vold.conf`

There are also two configuration files to define Volume Management's actions: `/etc/vold.conf` and `/etc/rmmount.conf`. See the `vold.conf`(4) and `rmmount.conf`(4) man pages for descriptions of these files, and see *System Administration Guide* for information on managing CD-ROM and diskette devices.

CHAPTER 8

Startup and Shutdown

This chapter describes changes to procedures for booting and shutting down a system.

See *System Administration Guide* for detailed descriptions of boot procedures. Man pages for each command are available on line in the "User Commands" section of *SunOS 4.x Reference Manual*, or in *man Pages(1): User Commands*.

Booting

The Solaris 2.6 boot process makes system administration easier. Some of the major changes include:

- The kernel is self-configuring so you no longer need to rebuild it manually.
- Kernel memory consumption is reduced by automatic loading of devices when first opened.
- File systems are checked only when necessary, improving boot time.
- The boot block can read UNIX file systems, eliminating boot errors when the boot program moves.
- Third-party bootable devices are supported.
- Secondary boot programs, `ufsboot` and `inetboot`, have been modified to read CacheFS file systems. This new booting capability enables Solstice AutoClient™ systems to boot more quickly and with less impact on network resources.

- The SunOS release 4.x `fastboot` command is available only on Solaris 2.6 systems that have the SunOS/BSD Source Compatibility Package installed. The `fastboot` command is obsolete in Solaris 2.6 systems because file systems are only checked if the file system state is identified as not clean.
- The SunOS release 4.x `halt` and `reboot` commands have `shutdown`(1M) and `init`(1M) equivalents in the SunOS release 5.6 software.

In the Solaris 2.6 operating environment, the `shutdown` and `init` commands are the preferred way to halt, shut down, or reboot your system. While the `reboot` command is available in the Solaris 2.6 operating environment, it brings the system down quickly without shutting down services in an orderly way. Table 8–1 shows the SunOS release 5.6 commands that replace SunOS release 4.x commands.

TABLE 8–1 SunOS release 5.6 Replacements for `reboot` and `fastboot`

SunOS release 4.x Command	SunOS release 5.6 Command Replacement
`reboot`	`shutdown –i –6`, `init 6`
`fastboot`	`boot`, `init 6`

`boot` Command Changes

The SunOS release 5.6 software has these additional options for the `boot` command:

- Type `boot –r` when you add new hardware or alter its location. This option creates the physical and logical device names, with the logical device name linked to the physical device name.
- Type `boot –v` when you want to see all the system bootup messages; the default is to boot silently. The messages are always stored in the `/var/adm/messages` file.
- Type `boot –a` when you want to be prompted for the name of an alternate kernel, `/etc/system` file, or path name for kernel module directories.

Booting From the PROM

Be aware of these changes when booting from PROM:

- The PROM loads `bootblk` from the disk. This file is similar to the previous SunOS release 4.x boot block except that it is specific to the UFS file system.

  As in the SunOS release 4.x software, you need to use `installboot`(1M) to install boot blocks on a partition to be used for booting.

- `bootblk` opens the boot device and, using the file system you specify, finds and loads `ufsboot`.
- The boot PROM loads the kernel, `/kernel/genunix`, after `ufsboot` is loaded into memory. SunOS release 4.x systems used `/vmunix`; however, in the SunOS release 5.6 software the `/kernel` directory contains all platform-independent kernel modules, including `unix`, needed to boot the system.
- The kernel, in turn, loads other drivers, such as `esp`, from the `/kernel/drv` directory. These drivers had to be built as part of the SunOS release 4.x kernel but can be dynamically loaded in SunOS release 5.6 systems when they are needed.
- The `/sbin/init` command generates processes to set up the system based on the directions in `/etc/inittab`. The next section describes the run levels that `init` uses.

Summary of Boot Differences

Table 8–2 summarizes booting differences.

TABLE 8–2 Summary of Booting Differences

SunOS release 4.x	SunOS release 5.6	Feature
`bootsd`	`bootblk`	Now loads `ufsboot` from disk
`boot` program	`ufsboot`	Now loads `unix` from disk
`/vmunix`	`/kernel/genunix`	Bootable kernel image
`boot.sun4c.sunos.4.1`	`inetboot`	Mounts and copies `unix` from network

TABLE 8–2 Summary of Booting Differences *(continued)*

SunOS release 4.x	SunOS release 5.6	Feature
`rc.boot rc.single`	`/etc/rcS`	Mounts `/usr` and checks file systems
`rc.local`	`/etc/rc2 /etc/rc3`	System config scripts
`/etc/config`	`modload /etc/system`	Customizes system kernel, loads modules as needed
Prom monitor, single user, multiuser	Run states 0 – 6, and S	System run levels
`/dev/sd1g`	`/dev/dsk/c0t1d0s6`	More descriptive logical device names. See "Device Naming Conventions" on page 57.
`MAKEDEV`	`boot –r`, `add_drv`	Makes device nodes

Using the `init` Command

The `init`(1M) command replaces the SunOS release 4.x `fasthalt` command in the SunOS release 5.6 software. Use it to shut down a single-user system. You can use `init` to place the system in a power-down state (`init 0`) or into single-user state (`init 1`).

`init` Command Changes

Note the following changes to the `init` command:

- SunOS release 5.6 system software has eight initialization states (`init` states or run levels). The default `init` state is defined in the `/etc/inittab` file.
- The SunOS release 5.6 `init` command uses a different script for each run level instead of grouping all the run levels together in the `/etc/rc`,

`/etc/rc.boot`, and `/etc/rc.local` files. The files, named by run level, are located in the `/sbin` directory.

System Administration Guide describes this command in detail.

Changing System Run Levels

The SunOS release 5.6 `init` command enables you to control the run level (initialization state) of your system and move easily between various modes of operation. The SunOS release 5.6 `/sbin/rc` scripts control each individual run level instead of putting all system states into one file. This enables you to make changes in a unique file if you create new scripts or modify existing ones. SunOS release 4.x systems controlled run levels using `/etc/rc`, `/etc/rc.boot`, and `/etc/rc.local` files.

The SunOS release 4.x software had three run levels: prom monitor, single user, and multiuser. These correspond to run levels 0, 1, and 3 in the SunOS release 5.6 software.

Table 8–3 gives an overview of what each run level's `/sbin/rc` script does.

TABLE 8–3 SunOS release 5.6 System Initialization Run Levels

Run Level	Default SunOS release 5.6 Function
0	Shuts down the system so it is safe to turn off power. Stops system services and daemons. Terminates all running processes. Unmounts all file systems.
1	Single-user (system administrator) state for tasks that allow only one user on the system. Stops system services and daemons. Terminates all running processes. Unmounts all file systems.
2	Normal multiuser operation without NFS systems exported. Sets the *timezone* variable. Mounts the `/usr` file system. Cleans up the `/tmp` and `/var/tmp` directories. Loads the network interfaces and starts processes. Starts the `cron` daemon. Cleans up the `uucp` tmp files. Starts the `lp` system. Starts the `sendmail` daemon.
3	Normal multiuser operation of a file server with NFS systems exported. Completes all of the tasks in run level 2. Starts the NFS system daemons.
4	Alternative multiuser state (not used).

TABLE 8–3 SunOS release 5.6 System Initialization Run Levels *(continued)*

Run Level	Default SunOS release 5.6 Function
5	Software reboot, prompts for a boot device other than the EEPROM default.
r	Reboot. Kills all active processes. Unmounts the file systems and reboots to the *initdefault* entries in `/etc/inittab`.
S,s	Single-user state, running with some file systems mounted and accessible.

Shutting Down

Use the `shutdown`(1M) command when shutting down a system with multiple users. The command sends a warning to all logged-in users and, after 60 seconds, shuts the system down to single-user state.

- The SunOS release 4.x `fasthalt` commands are available only on SunOS release 5.6 systems that have the SunOS /BSD Source Compatibility Package installed.
- The SunOS release 4.x `halt` and `reboot` commands have `shutdown` and `init` equivalents.

See *System Administration Guide* for detailed descriptions of shutdown procedures. .

In the SunOS release 5.6 software, the `shutdown` command is the preferred way to halt or shut down a system. `shutdown` and `init` use `rc` scripts to kill running processes. While the `halt` command is available in the SunOS release 5.6 software, it stops the system quickly without shutting down services in an orderly way. Table 8–4 shows the SunOS release 5.6 commands that replace those in the SunOS release 4.x system.

TABLE 8–4 SunOS release 5.6 Replacements for `halt` and `fasthalt`

SunOS release 4.x Command	SunOS release 5.6 Command Replacement
`halt`	`shutdown -i 0`, `init 0`
`fasthalt`	`shutdown -i 0`, `init 0`

The `shutdown` and `init` commands accept a numerical “run-level” argument that controls the shutdown sequence. See the `shutdown` and `init` man pages for information about the run-level numbers.

Changes to the `shutdown` Command

The SunOS release 5.6 `shutdown` command includes only the options in Table 8–5. This command and its options are described in *System Administration Guide*.

TABLE 8–5 SunOS release 5.6 `shutdown` Command Options

Option	Description
`–g`	Selects “grace” period before `shutdown` begins.
`–i` [*init state*]	Specifies an initial run level (see Table 8–3).
`–y`	Runs `shutdown` without asking confirmation questions. Assumes a “yes” response to all questions.
`–message`	Specifies user-supported message. If more than one word, use quotes around the message.

By default, the SunOS release 5.6 `shutdown` command asks you to confirm before an actual `shutdown` begins. You can use the `–y` option to run it without operator intervention.

The `shutdown` options are available only in BSD source compatibility mode on Solaris 2.6 systems.

See Appendix A, for a summary of changes. See `shutdown`(1M) for information about how the command works.

Using the `fasthalt` and `fastboot` Commands

The SunOS release 4.x `fastboot` and `fasthalt` commands are available if you are running the SunOS/BSD Source Compatibility Package on Solaris 2.6 systems. The file-system checking features of these commands are not appropriate to a Solaris 2.6 system.

Using the `halt` and `reboot` Commands

The `halt` and `reboot` commands do not run the `rc` scripts in `/sbin`, so they are not recommended. Since the `halt` and `reboot` commands in SunOS release 5.6 systems are not available on other AT&T SVR4 systems, both commands have `shutdown` and `init` equivalents.

CHAPTER 9

File System Administration

This chapter familiarizes you with changes to file system layout and the changes to file systems, virtual file systems, directories, and files. The chapter also describes changes to file system administration including:

- Mounting file systems
- Monitoring file systems
- Sharing file systems
- Creating new file systems
- Checking file systems
- Backing up and restoring files
- “File System Changes” on page 73
- “Default File Systems and Directories” on page 75
- “Virtual File System Architecture” on page 77
- “Directory and File Changes” on page 82
- “ Using File System Administration Commands” on page 90

For more information on understanding and managing file systems, see *System Administration Guide.*

File System Changes

SunOS release 5.6 and SunOS release 4.x file systems are similar, but there are changes in the locations and names of system directories and files. There are also new file systems and new pseudo file systems, and one directory was removed.

Some of the changes to file system locations and names are:

- The `/dev` directory has changed from a flat directory to a hierarchical one.
- The `/etc` directory has changed and contains specific system configuration information. Several files and subdirectories have been added, removed, or changed.
- The `/etc/vfstab` tab file replaces `/etc/fstab`.
- The `/etc/lp` directory replaces `/etc/printcap`.
- The SunOS release 5.6 `/sbin` directory contains the `rc` scripts used to alter system run levels as well as the `rcs` script used to initialize the system prior to mounting file systems.
- The SunOS release 5.6 `/usr` directory contains sharable files and executables provided by the system.
- The `/var` directory contains files that change size during normal operation. Several files and subdirectories in the `/var` directory have been added, removed, or changed.
- The `/var/mail` directory replaces `/var/spool/mail`.
- The `/sys` directory is no longer needed because the kernel is dynamically loaded.
- The `/RFS` file system has been removed.
- The `terminfo` database replaces `termcap`.

Pseudo File Systems

The TFS pseudo file system is not included in the SunOS release 5.6 software.

The added pseudo file systems are:

- The CACHEFS pseudo file system can be used to improve performance of slow devices such as CD-ROM.
- The PROCFS pseudo file system resides in memory and contains a list of active processes, by process number, in the `/proc` directory. See the `proc`(4) man page.
- The FDFS pseudo file system provides explicit names for opening files using file descriptors.
- The FIFOFS pseudo file system contains pipe files that give processes common access to data.
- The NAMEFS pseudo file system is used mostly by STREAMS for dynamic mounts of file descriptors on top of files.
- The SWAPFS pseudo file system is the default swap device when the system boots or you create additional swap space.

Added File Systems

The following file systems are included in the SunOS release 5.6 directory structure:

- The kernel (now called `unix`) and the kernel modules are stored in the `/kernel` directory.
- The optional `/opt` file system can be used to store third-party or unbundled software. If `/opt` is not a separate file system, it may be a symbolic link to `/usr/opt`.
- The `/vol` file system provides the default file system for the Volume Management daemon, `vold`(1M). See the `volfs`(7) man page.

Default File Systems and Directories

The SunOS release 5.6 file system is hierarchical. Figure 9–1 graphically depicts SunOS release 5.6 default directories and file systems (indicated by dotted lines). Subdirectories shown are just a sample of what the directory or file system actually holds. Table 9–1 gives a brief description of each.

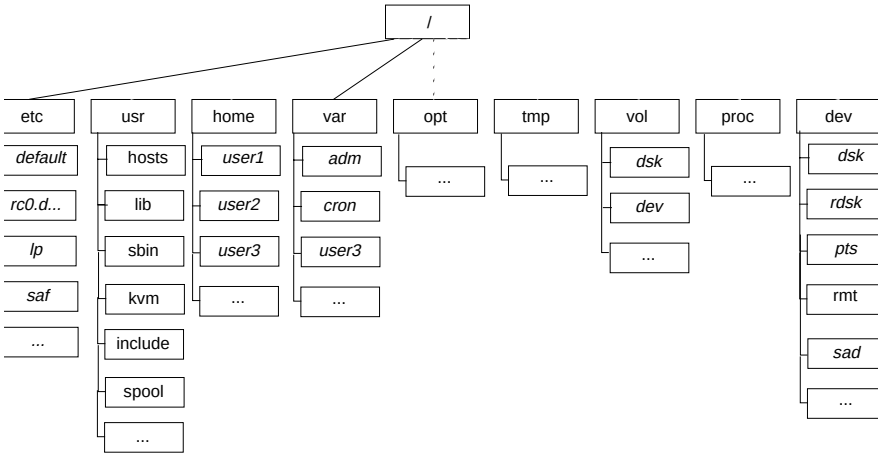


Figure 9–1 Solaris 2.6 Default File Systems and Directory Hierarchy

The Solaris 2.6 software contains a default set of file systems and directories, and uses a set of conventions to group similar types of files together. Table 9–1 lists the default file systems and directories with a brief description.

TABLE 9–1 Solaris 2.6 File Systems and Directories

File System or Directory	Type	Description
`/`	File system	The top of the hierarchical file tree. The root directory contains the directories and files critical for system operation, such as the kernel (`/kernel/unix`), the device drivers, and the programs used to boot the system. It also contains the mount point directories where local and remote file systems can be attached to the file tree.
`/etc`	Directory	Contains system files used in system administration
`/usr`	File system	Contains architecture-dependent and -independent sharable files. Files such as man pages that can be used on all types of systems are in `/usr/share`.
`/home`	File system	The mount point for the users' home directories, which store users' work files. By default, `/home` is now an automounted file system.
`/var`	Directory	Contains system files and directories that are likely to change or grow over the life of the local system. These include system logs, `vi` and `ex` backup files, and `uucp` files.
`/opt`	File system	Mount point for optional third-party software. On some systems `/opt` may be a UFS file system on a local disk partition.
`/tmp`	File system	Temporary files, cleared each time the system is booted or `/tmp` is unmounted
`/vol`	File system	Contains directories for removable media, managed by `vold`(1M)
`/proc`	File system	Contains a list of active system processes, by number. Does not use any disk space.
`/sbin`	Directory	Essential executables used in the booting process and in manual system recovery

Virtual File System Architecture

The SunOS release 5.6 software features a virtual file system (VFS) architecture that simplifies file system management for systems that support multiple file systems.

Over the years, several different UNIX file systems were developed, with each its own set of commands for file system management. Learning all the variations can be confusing and difficult. The SunOS release 5.6 software addresses this issue with a set of generic commands for file system management. These commands are a part of a common VFS interface that makes differences between file systems transparent with respect to maintenance. The subsections below list a summary of supported file systems and the generic file system commands.

Supported File System Types

Most file system types included in the SunOS release 4.x software are also included in the SunOS release 5.6 software. There is one exception: The translucent file system (TFS) type has been withdrawn from the SunOS release 5.6 software. Table 9–2 summarizes file-system type availability in the SunOS release 4.x and SunOS release 5.6 environment.

TABLE 9–2 Summary of File System Types

Category	Name	Description	SunOS release 4.x	SunOS release 5.6
Disk-based	UFS	UNIX file system	Yes	Yes
	HSFS	CD-ROM file system	Yes	Yes
	PCFS	PC file system	Yes	Yes
Network-based	NFS	Sun's distributed computing file system	Yes	Yes

TABLE 9–2 Summary of File System Types *(continued)*

Category	Name	Description	SunOS release 4.x	SunOS release 5.6
Pseudo	SPECFS	Device special file system	Yes	Yes
	TMPFS	/tmp temporary file system	Yes	Yes
	LOFS	Loopback file system	Yes	Yes
	TFS	Translucent file system	Yes	No
	PROCFS	Process access file system	No	Yes
	FDFS	File descriptor file system	No	Yes
	FIFOFS	FIFO/Pipe file system	No	Yes
	NAMEFS	Name file system	No	Yes
	SWAPFS	Swap file system	No	Yes
	CACHEFS	Cache file system	No	Yes

For more information on file systems, see the proc(4) and fd(4) man pages and *System Administration Guide.*

Cache File System (CACHEFS)

The Cache File System can be used to improve performance of remote file systems or slow devices such as CD-ROM. When a file system is cached, the data read from the remote file system or CD-ROM is stored in a cache on the local system.

Swap File Changes

In the SunOS release 5.6 software SWAPFS is the default swap device when the system boots or you create additional swap space. This swap device uses physical memory as swap space, but also requires physical swap space on disk.

In SunOS release 4.x systems, the default physical swap device depends on the system configuration. Standalone systems default to `sd0b` and diskless systems get their swap files from the `bootparam` server. The SunOS release 5.6 software uses the swap file as the default dump device instead of specifying a file on disk.

Unsupported SVR4 File System Types

Table 9–3 shows SVR4 file system types that are not supported in the SunOS release 5.6 software.

TABLE 9–3 Not Supported SVR4 File System Types

Name	Description
`BFS`	Boot file system
`S5`	System V file system
`xnamefs`	XENIX semaphore file system

Generic File System Commands

Most file system administration commands have a generic and a file system component. Use the generic commands, which call the file system component. Table 9–4 lists the generic file-system administrative commands, which are located in the `/usr/bin` directory.

TABLE 9–4 Generic File-System Administrative Commands

Command	Description
`clri`(1M)	Clears inodes
`df`(1M)	Reports the number of free disk blocks and files
`ff`(1M)	Lists file names and statistics for a file system

TABLE 9–4 Generic File-System Administrative Commands *(continued)*

Command	Description
`fsck`(1M)	Checks the integrity of a file system and repairs any damage found
`fsdb`(1M)	File system debugger
`fstyp`(1M)	Determines the file-system type
`labelit`(1M)	Lists or provides labels for file systems when copied to tape (for use by the `volcopy` command only)
`mkfs`(1M)	Makes a new file system
`mount`(1M)	Mounts file systems and remote resources
`mountall`(1M)	Mounts all file systems specified in a file-system table
`ncheck`(1M)	Generates a list of path names with their i-numbers
`umount`(1M)	Unmounts file systems and remote resources
`umountall`(1M)	Unmounts all file systems specified in a file-system table
`volcopy`(1M)	Makes an image copy of a file system

Most of these commands also have a file system counterpart.

Caution - Do not use the file system commands directly. If you specify an operation on a file system that does not support it, the generic command displays this error message:
command: `Operation not applicable for FSType` *type*.

Syntax of Generic Commands

Most of these commands use this syntax:

command [`–F` *type*] [`–V`] [*generic-options*] [`–o` *specific-options*] [*special*|*mount-point*] [*operands*]

The options and arguments to the generic commands are:

–F *type*

Specifies the type of file system. If you do not use this option, the command looks for an entry that matches `special` or `mount point` in the `/etc/vfstab` file. Otherwise, the default is taken from the file `/etc/default/fs` for local file systems and from the file `/etc/dfs/fstypes` for remote file systems.

–V

Echoes the completed command line. The echoed line may include additional information derived from `/etc/vfstab`. Use this option to verify and validate the command line. The command is not run.

generic-options

Options common to different types of file systems.

–o *specific-options*

A list of options specific to the type of file system. The list must have the following format: –o followed by a space, followed by a series of *keyword* [=*value*] pairs separated by commas with no intervening spaces.

special | *mount-point*

Identifies the file system. The name must be either the mount point or the special device file for the slice holding the file system. For some commands, the *special* file must be the raw (character) device; for other commands it must be the block device. In some cases, this argument is used as a key to search the file `/etc/vfstab` for a matching entry from which to obtain other information. In most cases, this argument is required and must come immediately after *specific-options*. However, the argument is not required when you want a command to act on all the file systems (optionally limited by type) listed in the `/etc/vfstab` file.

operands

Arguments specific to a type of file system. See the specific man page of the command (for example, `mkfs_ufs`(4) for a detailed description.

System-wide Default File System Type

The default remote file system type is `/etc/dfs/fstype`. The default local file system type is `/etc/default/fs`. See the `default_fs`(4) man page for more information.

Command Locations

In previous SunOS releases, all file system commands were located in the `/etc` directory. In the SunOS release 5.6 software, file system commands are

organized into separate hierarchies for convenience. All of the file system commands are included in `/usr/lib/fs/`*fstype*. Commands needed before `/usr` is mounted are duplicated in `/etc/fs/`*fstype*.

The generic commands are located in `/usr/sbin`. The commands needed before `/usr` is mounted are duplicated in `/sbin`.

Table 9–5 lists the locations of the file-system commands.

TABLE 9–5 Locations of File System Commands

Type	Location of Primary Version	Location of Duplicate Version (`root`)
Generic	`/usr/sbin`	`/sbin`
Specific	`/usr/lib/fs`	`/etc/fs`

Directory and File Changes

This section describes the changes to directories and files between the SunOS release 4.x and SunOS release 5.6 environment.

`/dev` Directory

The `/dev` directory has changed from a flat directory to a hierarchical one. Table 9–6 describes the subdirectories that have been added.

TABLE 9–6 Additions to the `/dev` Directory

Subdirectory	Description
`/dev/dsk`	Contains block disk devices
`/dev/rdsk`	Contains raw disk devices

TABLE 9–6 Additions to the /dev Directory *(continued)*

Subdirectory	Description
/dev/pts	Contains pseudo terminal (pty) slave devices
/dev/rmt	Contains raw tape devices
/dev/sad	Contains entry points for the STREAMS Administrative Driver
/dev/term	Contains terminal devices

/etc Directory

The /etc directory contains system configuration information. Several files and subdirectories have been added, removed, or changed.

- File system commands, such as mount*, have been moved to subdirectories of the /usr/lib/fs directory.
- The SunOS release 4.x /etc/fstab file has been replaced by /etc/vfstab.

  Initialization scripts, such as rc, rc.boot, rc.local, and rc.single, are not available in the SunOS release 5.6 software. They are replaced by the scripts shown in Table 9–7, which are run by their corresponding run control files. Table 9–8 describes the subdirectories that have been added to the SunOS release 5.6 /etc directory.

TABLE 9–7 Initialization Scripts and Their Run Control Files

Scripts	Run Control Files
/etc/rc0.d	/sbin/rc0
/etc/rc1.d	/sbin/rc1
/etc/rc2.d	/sbin/rc2
/etc/rc3.d	/sbin/rc3

TABLE 9–7 Initialization Scripts and Their Run Control Files *(continued)*

Scripts	Run Control Files
/etc/rc4.d	/sbin/rc4
/etc/rc5.d	/sbin/rc5
/etc/rc6.d	/sbin/rc6
/etc/rcS.d	/sbin/rcS

TABLE 9–8 Additions to the /etc Directory

Subdirectory	Description
/etc/default	Defines default system configuration
/etc/inet	Defines Internet services configuration
/etc/lp	Defines LP system configuration
/etc/opt	Defines installed optional software
/etc/rcn.d	Defines run-state transition operations
/etc/saf	Defines Service Access Facility (SAF) configuration

/etc/vfstab File

In the SunOS release 5.6 software, the virtual file system file /etc/vfstab replaces the /etc/fstab file. In the virtual file system architecture, the /etc/vfstab file provides default file system parameters used by the generic commands for file system management. For information about these commands, see "Generic File System Commands" on page 79.

In addition to the name change, the /etc/vfstab file is different from the /etc/fstab file in the following ways:

- A `device to fsck` field has been added to specify the names of raw devices to be checked by `fsck`.
- An `automount` field has been added to control the routine mounting of file systems by `mountall` (the automount daemon does not use this field).
- The `freq` field, which specified the number of days between dumps, has been eliminated.

The file system table has seven fields, each separated by a tab. Table 9–9 explains the field entries.

Note - You must have an entry in each field in the `/etc/vfstab` file. If there is no value for a field, be sure to type a dash (-).

TABLE 9–9 `/etc/vfstab` File Field Names and Content

Field Name	Content
`device to mount`	The entry in this field may be any of the following: The block special device for local UFS file systems (for example, `/dev/dsk/c0t0d0s0`) The resource name for remote file systems (for example, `myserver:/export/home` for an NFS system) The name of the slice on which to swap (for example, `/dev/dsk/c0t3d0s1`) The `/proc` directory and `proc` file system type CD-ROM as `hsfs` file system type `/dev/diskette` as `pcfs` or `ufs` file system type This field is also used to specify swap file systems. For more information on remote file systems, see *NFS Administration Guide* .
`device to fsck`	The raw (character) special device that corresponds to the file system identified by the `device to mount` field (for example, `/dev/rdsk/c0t0d0s0`). This field determines the raw interface that is used by `fsck`. Use a dash (`-`) when there is no applicable device, such as for a read-only file system or a network-based file system.
`mount point`	The default mount-point directory (for example, `/usr` for `/dev/dsk/c0t0d0s6`).

TABLE 9–9 `/etc/vfstab` File Field Names and Content *(continued)*

Field Name	Content
`FS type`	The type of file system identified by the `device to mount` field.
`fsck pass`	The pass number used by fsck to determine whether to check a file system. When the field contains a dash (-), the file system is not checked. When the field contains a value of 1 or more, the file system is checked; non-UFS file systems with a 0 `fsck pass` are checked. For UFS file systems only, when the field contains a 0, the file system is not checked. When `fsck` is run on multiple UFS file systems that have `fsck pass` values greater than 1 and the preen option (`–o p`) is used, `fsck` automatically checks the file systems on different disks in parallel to maximize efficiency. When the field contains a value of `1`, the file system is checked sequentially. Otherwise, the value of the pass number does not have any effect. In SunOS 5.6 system software, the `fsck pass` field does not explicitly specify the order in which file systems are checked.
`automount?`	`yes` or `no` for whether the file system should be automatically mounted by mountall when the system is booted. An `auto` in the fourth column of your SunOS release 4.x `/etc/fstab` would translate to a "yes" in this column; a `noauto`, a "no." Note that this field has nothing to do with the `automount` program.
`mount options`	A list of comma-separated options (with no spaces) that are used in mounting the file system. Use a dash (-) to show no options. See the `mount`(1M) man page for a list of the available options.

For detailed information about the `/etc/vfstab` file, see *System Administration Guide*.

`/etc/shadow` File

The SunOS release 5.6 software contains an `/etc/shadow` file, which includes entries that force password aging for individual user login accounts. The `/etc/shadow` file also contains encrypted passwords. The `/etc/shadow` file does not have general read permissions. This prevents general access to the encrypted passwords that formerly appeared in the `/etc/passwd` file.

/sbin Directory

The SunOS release 5.6 /sbin directory contains the rc scripts used to alter system run levels as well as the rcs script used to initialize the system prior to mounting file systems. See the rc man pages and “Changing System Run Levels” on page 69 for a description of the scripts.

/usr Directory

The SunOS release 5.6 /usr directory contains sharable files and executables provided by the system. Table 9–10 describes the subdirectories that have been added to the SunOS release 5.6 /usr directory.

TABLE 9–10 Additions to the /usr Directory

Subdirectory	Description
/usr/ccs	C compilation systems
/usr/snadm	Executables and other files used by admintool

Table 9–11 shows files that were in the SunOS release 4.x /usr directory but have been moved in the SunOS release 5.6 software.

TABLE 9–11 Files Changed in the /usr Directory

SunOS release 4.x Location	SunOS release 5.6 Location
/usr/5bin	/usr/bin
/usr/5include	/usr/include
/usr/5lib	/usr/lib
/usr/etc	/usr/sbin
/usr/old	Contents removed
/usr/xpg2bin	/usr/bin

TABLE 9–11 Files Changed in the /usr Directory *(continued)*

SunOS release 4.x Location	SunOS release 5.6 Location
/usr/xpg2lib	/usr/lib
/usr/xpg2include	/usr/include

Appendix E, contains tables with detailed information about the directories and files in each of these file systems.

/var Directory

The /var directory contains files that change sizes during normal operation. Several files and subdirectories in the /var directory have been added, removed, or changed.

- The /var/opt/*packagename* directory contains software package objects that change sizes, such as log and spool files.
- The /var/sadm directory contains databases maintained by the software package management utilities.
- The /var/saf directory contains Service Access Facility (SAF) logging and accounting files.
- The SunOS release 4.x /var/spool/mail directory has been moved to /var/mail.

  Two directories were added to the SunOS release 5.x file system: /kernel and /opt.

/kernel Directory

The SunOS release 5.6 /kernel directory contains the operating system kernel and kernel-level object modules that were in the SunOS release 4.x /sys directory. Table 9–12 describes the subdirectories that have been added to the /kernel directory.

TABLE 9–12 Additions to the /kernel Directory

Subdirectory	Description
/kernel/drv	Device driver and pseudo-device driver modules
/kernel/exec	Kernel modules to run ELF or a.out executable files
/kernel/fs	Kernel modules that implement file systems such as ufs, nfs, proc, fifo, and so on
/kernel/misc	Miscellaneous modules
/kernel/sched	Modules containing scheduling classes and corresponding dispatch tables
/kernel/strmod	STREAMS modules
/kernel/sys	Loadable system calls such as system accounting and semaphore operations
/kernel/unix	Operating system kernel, loaded at boot time

/opt Directory

The SunOS release 5.6 /opt directory contains optional add-on application software packages. These packages were installed in the SunOS release 4.x /usr directory..

/sys Directory

The /sys directory has been retired. Its files, used to reconfigure the kernel, have been made obsolete by the dynamic kernel.

Using File System Administration Commands

The file system administration commands that have changed in the SunOS release 5.6 software include those for:

- Mounting file systems
- Monitoring file systems
- Sharing file systems
- Creating a new file system
- Checking a file system
- Backing up and restoring files

When you are ready to administer file systems on your SunOS release 5.6 system, see *System Administration Guide* for details on performing the tasks involved.

Mounting File Systems and `autofs`

The biggest change to the mounting capability is automatic mounting or `autofs`. The `autofs` program automatically mounts directories when you access them using, for example `cd`(1) or `ls`(1). This capability includes file hierarchies, CD-ROM, and diskette file systems.

`autofs` starts automatically when the system enters run level 3, or you can invoke it from a shell command line.

`autofs` works with the file systems specified in *maps*. These maps can be maintained as NIS, NIS+, or local files. The `autofs` maps can specify several remote locations for a particular file. This way, if one of the servers is down, `autofs` can try to mount from another system. You can specify in the maps which servers are preferred for each resource by assigning each server a weighting factor.

Mounting some file hierarchies with `autofs` does not exclude the ability to mount others with the `mount` command. A diskless system must have entries for `/` (root), `/usr`, and `/usr/kvm` in the `/etc/vfstab` file. Because shared file systems should always remain available, do not use `autofs` to mount `/usr/share`.

The following example shows how to manually mount a file system listed in the `/etc/vfstab` file using the `mount` command.

1. Change to the directory in which you want to create the mount point.

2. **Create the mount-point directory.**
3. **Specify either the mount point or the block device.**

   Specifying the mount point is usually easier. The rest of the information is read from `/etc/vfstab`.
4. **Become root and type the `mount` command, specifying either the mount point or the block device.**

   Specifying the mount point is usually easier. The rest of the information is read from `/etc/vfstab`.

```
# mount mount-point
```

   The file system is now mounted.

   For instructions showing how to mount different types of file systems using `mount` with or without options, see *System Administration Guide.*

Changes to the `mount` Command

Some of the names and forms of the `mount` commands are different, as listed in Table 9–13.

TABLE 9–13 `mount` Command Differences

SunOS Release 4.x	SunOS release 5.6
`mount`	`mount`
`mount –a`	`mountall`
`umount`	`umount`
`umount –a`	`umountall`
`exportfs`	`share`
`exportfs –u`	`unshare`

TABLE 9–13 `mount` Command Differences *(continued)*

SunOS Release 4.x	SunOS release 5.6
`showmount –a`	`dfmounts`
`showmount –e`	`dfshares`

See Appendix A, for more information on changes to these commands.

Automatic Mounting of `/cdrom` and `/floppy`

In this release, the CD-ROM and diskette file systems are automatically mounted in `/cdrom` and `/floppy` when removable media are inserted into these drives. Since these file systems are now managed by the Volume Management daemon, `vold`(1M), you cannot mount these devices yourself. See “Using Volume Management ” on page 61 for more information.

Specifying File Systems in the `/etc/vfstab` File

In the SunOS release 5.6 system, you need to list file systems that you want mounted at system startup in your `/etc/vfstab`, instead of in the `/etc/fstab` file. The format of `/etc/vfstab` differs from that of `/etc/fstab`. For a discussion of the `/etc/vfstab` file, see “`/etc/vfstab` File ” on page 84.

Monitoring File Systems

Table 9–14 shows the file and directory monitoring commands and changes, where they apply.

TABLE 9–14 File and Directory Monitoring Commands

Command	Information Provided	Change (if applicable)
`ls`	Size, age, permissions, owner of files	None
`du`	Total size of directories and their contents	None
`df`	Disk space occupied by file systems, directories, or mounted resources; used and available disk space	The SunOS release 4.x version of this command provides a different output format containing somewhat different output than the SunOS release 5.6 `df` command. The SunOS release 5.6 `–k` option provides output formats similar to those in the SunOS release 4.x command. The SunOS release 4.x `df –t` *filesystem* type reports on files of the specified type, whereas the SunOS release 5.6 `df–t` command prints full listings with totals.
`quot`	Number of blocks owned by users	None
`find`	Names of files meeting search criteria	The `–n` *cpio-device* SunOS release 4.x option is not available in the SunOS release 5.6 command. Write the current file on device in `cpio –c` format.

Sharing File Systems

File systems were “exported” in the SunOS release 4.x software to make them available to other systems. This was done through the `/etc/exports` file and the `exportfs` command. However, only NFS systems could be exported.

In the SunOS release 5.6 software, this same concept is referred to as "sharing resources," and it has been expanded to include more file systems. File systems are shared with the `share`(1M) and `shareall`(1M) commands. The `share` command is similar to the `exportfs` *pathname* command, while `shareall` is similar to the `exportfs –a` command.

The `share –F` *fstype* option specifies the type of file system to be shared. If the `–F` option is not specified, `share` uses the first file-system type listed in the `/etc/dfs/dfstab` file.

File systems that you want to be shared automatically should have `share` command entries in the `/etc/dfs/dfstab` file (which replaces the `/etc/export` file). The commands specified in this file are run automatically when the system enters run level 3 (multiuser mode with network file sharing).

Example of `/etc/dfs/dfstab` File Entries

The following entry gives clients on `mercury`, `venus`, and `mars` read-write access to `/export/home1`; the second entry gives clients on `saturn` and `jupiter` read-only access to `/export/news`.

```
share -F nfs -o rw=mercury:venus:mars -d ‘‘Home Dir’’ /export/home1
share -F nfs -o ro=saturn:jupiter -d ‘‘News Postings’’ /export/news
```

When the system is running in multiuser mode, these file systems are available to the clients listed. The `share` command displays all resources shared by the local system:

```
% share
-          /export/home1  rw=mercury:venus:mars  ‘‘Home Dir’’
-          /export/news   ro=saturn:jupiter  ‘‘News Postings’’
```

Creating New File Systems

You define, specify, and create a new file system using either the `newfs`(1M) or the `mkfs`(1M) command. The following sections highlight changes in the `newfs` and `mkfs` commands.

`newfs` Command

The SunOS release 5.6 `newfs` command is a convenient front end to the `mkfs` command. The `newfs` command does not support the virtual file-system architecture; it is intended for creating UFS-type file systems only.

When you use `newfs`, it calls and passes arguments to `mkfs`, which does the real work when creating a `ufs` file system.

The `newfs` command accepts only names that conform to the SunOS release 5.6 device naming conventions (see “Device Naming Conventions” on page 57).

`mkfs` Command

The SunOS release 5.6 `mkfs` command differs significantly from the SunOS release 4.x version of the command. The SunOS release 5.6 version provides for different file system types, and its command syntax is entirely different (see “Generic File System Commands” on page 79). Like `newfs`, `mkfs` accepts only names conforming to the SunOS release 5.6 device naming conventions.

Although `mkfs` now supports different types of file systems, in practice it is almost always used to create `ufs` file systems. However, `mkfs` isn’t usually run directly; it is usually called by the `newfs` command.

See `mkfs`(1) man pages for additional details.

Checking File Systems

The SunOS release 5.6 `fsck`(1M) command differs significantly from the SunOS release 4.x version of the command. In keeping with the virtual file-system (VFS) architecture, the `fsck` file-checking utility has two parts:

- A generic command that is called first, regardless of the type of file system.
- A specific command that is called by the generic command, depending on the type of the target file system (see “Generic File System Commands” on page 79).

  In addition, `fsck` accepts only names conforming to the SunOS release 5.6 device naming conventions. For more information, see “Device Naming Conventions” on page 57.

  The `fsck` command performs faster consistency checks at mount time. In addition, the SunOS release 5.6 software does not require you to reboot the system after running `fsck` on the root and `/usr` file systems. This results in faster system startup compared to previous SunOS releases. The `fsck –m` command enables you to skip checking for file systems that are clean. See `fsck`(1m) for additional details.

Backing Up and Restoring Files

This section discusses the changes to backup and restore commands and SunOS release 5.6 and describes how to use the `ufsdump`, `ufsrestore`, `dd`, `tar`, and `cpio` commands.

The SunOS release 4.x software supported several utilities for backing up and restoring files: `dump`, `restore`, `tar`, `cpio`, `dd`, and `bar`, as well as the unbundled Backup CoPilot program. This release supports all of these utilities except `bar` and Backup Copilot. SunOS release 4.x `bar` files can be restored on a SunOS release 5.6 system but you cannot create new `bar` files. The `dump`(8) and `restore`(8) commands were renamed `ufsdump`(1M) and `ufsrestore`(1M). Files created with the SunOS release 4.x `dump` command can be restored on a SunOS release 5.6 system with `ufsrestore`.

The SunOS release 5.6 software has two additional utilities for copying file systems: `volcopy`(1M) and `labelit`(1M).

`ufsdump` Command

The `ufsdump` command accepts the same command syntax as the SunOS release 4.x `dump` command. `ufsdump` also accepts options listed in Table 9–15.

TABLE 9–15 `ufsdump` Command Options Not Available With the `dump` Command

Option	Function
`–l`	Autoload. When reaching the end of a tape (before completing the dump), take the drive off line and wait up to two minutes for the tape drive to be ready again. This gives autoloading (stackloader) tape drives a chance to load a new tape. If the drive is ready within two minutes, continue. If it is not ready after two minutes, prompt an operator to load another tape, as usual, and wait.
`–o`	Off line. When finished with a tape or diskette (completing the dump or reaching the end of the medium), take the drive off line. In the case of a diskette drive, also eject the diskette. In the case of a tape drive, also rewind the tape. This prevents another process that rushes in to use the drive from inadvertently converting the data.
`–S`	Estimate size of dump. Determine the amount of space that is needed to perform the dump and output a single number indicating the estimated size of the dump in bytes. This is most useful for incremental backups.

Unlike `dump`, `ufsdump` can detect the end of medium, so you no longer have to use the `–s` size option to force dump programs to move to the next tape before reaching the end. Nevertheless, to ensure compatibility with older versions of the `restore` command, the `–s` option has been retained in `ufsdump`.

Even though `ufsdump` now can detect the end of medium, it has no way to predict the number of diskettes or tapes needed for a dump—unless you specify the medium size with the `–s` option. Therefore, the messages displayed at the start of a backup do not indicate the number of diskettes or tapes required unless you have specified the medium size.

The `–w` and `–W` options behave a little differently in the SunOS release 5.6 software. In the SunOS release 4.x software, these options list all file systems that are scheduled for backup according to the backup frequencies specified in the `/etc/fstab` file. Since the SunOS release 5.6 equivalent file, `/etc/vfstab`, has no provision for specifying backup frequencies, these options now assume that each file system will be backed up daily. Therefore, they now list any file systems that have not been backed up within a day.

When performing backups across the network (backing up local file systems to a remote tape drive), use the device naming convention that's appropriate for the system with the tape drive. If the system with the tape drive is a SunOS release 5.6 system, use the device naming convention to identify the tape drive; otherwise, use the SunOS release 4.x convention.

`ufsrestore` Command

The `ufsrestore` command in the SunOS release 5.6 software is similar to the `restore` command in the SunOS release 4.x software. You will be able to restore backups made with the SunOS release 4.x `dump` command with one exception: you cannot restore multivolume backups from diskette. If you have backup scripts that invoke `restore`, change them to invoke `ufsrestore` instead.

`dd` Command

In the SunOS release 4.x version of the `dd` command, the size suffix `–w` (words) denotes a size unit of 4 bytes. In the SunOS release 5.6 version, `–w` denotes a unit of 2 bytes. In addition, the SunOS release 5.6 version now supports the `–unblock` and `–block` conversion options.

`tar` and `cpio` Commands

Because they use a nonbinary format, the `tar` and `cpio` commands are the only utilities to successfully interchange data between SVR4

implementations. Other backup utilities, such as `ufsdump` and `dd`, are unique to the vendor and are not guaranteed to work successfully from one SVR4 implementation to another.

The `tar` command is unchanged in this release; it accepts the same options and command syntax as the SunOS release 4.x command. However, since the device naming scheme has changed in the SunOS release 5.6 software, the *tarfile* (or *device*) argument is affected. When using the `–f` function modifier, specify the device argument as `/dev/rmt/`*unit*, where *unit* is a tape drive number and density. Table 9–16 shows the tape drive density characters in tape device names.

TABLE 9–16 Tape Drive Density Characters in Tape Device Names

Density	Description
Null	Default “preferred” (highest) density
l	Low
m	Medium
h	High
c	Compressed
u	Ultra

The `tar` command no longer uses `/dev/rmt8` as its default output device. When the `–f` modifier is not used and the `TAPE` environment variable is not set, the `tar` command uses the defaults set in the `/etc/default/tar` file.

The SunOS release 5.6 `cpio` command supports the SunOS release 4.x options and command syntax. `cpio` has been expanded to include many new options, as listed in Table 9–17.

TABLE 9–17 Additional `cpio` Options

Option	Command Available With Option	Description
–A	`cpio –o`	Appends files to an archive.
–k	`cpio –i`	Attempts to skip corrupt file headers and I/O errors encountered. This option lets you copy files from a medium that is corrupted or out of sequence.
–L	`cpio –o` or `cpio –p`	Follows symbolic links.
–V	`cpio –i`, `cpio –o`, or `cpio –p`	Special verbose. Prints a dot for each file read or written. This option assures you that `cpio` is working, without printing all file names.
–C *bufsize*	`cpio –i` or `cpio –o`	Blocks I/O *bufsize* bytes to the record, where *bufsize* is a positive integer. When neither –C nor –B is specified, the default buffer size is 512 bytes.
–E *filename*	`cpio –i`	Specifies and inputs file containing a list of file names to be extracted from the archive.
–H *header*	`cpio –i` or `cpio –o`	Reads or writes header information in *header* format. *header* can be one of: `bar` (read only), `crc`, `CRC`, `odc`, `tar`, `TAR`, `ustar`, or `USTAR`.
–I *filename*	`cpio –i`	Reads *filename* as an input archive.
–M *message*	`cpio –i -I` *filename* or `cpio –o -O` *filename*	Defines a message to use when switching media.

TABLE 9–17 Additional `cpio` Options *(continued)*

–O *filename*	`cpio –o`	Directs the output to *filename*.
–R *userid*	`cpio –i` or `cpio –p`	Reassigns ownership and group information for each file to *userid*.

Note - `cpio` requires one of three mutually exclusive options to specify the action to take: `–i` (copy in), `–o` (copy out), or `–p` (pass).

CHAPTER 10

Setting Up a Solaris 2.6 Server to Support SunOS Release 4.x Diskless / Dataless Clients

This chapter outlines how to set up a Solaris 2.6 system as a server for SunOS release 4.x diskless / dataless clients by using the `discover4x`, `install4x`, and `convert4x` programs.

Make sure you have read Chapter 3, if you are setting up a Solaris 2.6 server for SunOS release 4.x clients on a Solaris 2.6 network.

Adding SunOS Release 4.x Support to a Solaris 2.6 Server

This section explains how to prepare a Solaris 2.6 server so it can serve SunOS release 4.x diskless and dataless clients.

Note - Ensure that all system data has been restored before you use the commands in this procedure. The `/export` file system is particularly important because it contains client information. See Chapter 3.

Some sites will need to continue using SunOS release 4.x clients after the server has been upgraded to Solaris 2.6 software. For instance, Sun-3™ systems cannot run Solaris 2.2 or later software and must continue to use the SunOS release 4.x software.

When a SunOS release 4.x `/export` partition is set up on a server running Solaris 2.6 software, it is referred to as *multiple OS operation*. Multiple OS operation enables the server to continue serving SunOS release 4.x clients while it runs the Solaris 2.6 operating environment.

The multiple OS operation package is called SUNWhinst and includes three programs that you will need to run to set up a SunOS release 4.x `/export` directory on a Solaris 2.6 server. The three programs are:

- `discover4x` – This program analyzes the support that remains for SunOS release 4.x clients after the server has migrated to the Solaris 2.6 operating environment. The program looks at the SunOS release 4.x client support and creates the databases that are required for installation of SunOS release 4.x diskless / dataless clients on the Solaris 2.6 server. If client support for a given architecture is missing, `discover4x` attempts to notify users that they will have to re-install this support using `install4x`. If there are SunOS release 4.x clients with the same architecture as the server that migrated to the Solaris 2.6 operating environment, you must re-install that architecture using the `install4x` command.
- `install4x` – This program is used to install the components of a SunOS release 4.x system required to support diskless / dataless clients that existed before the migration to the Solaris 2.6 operating environment.
- `convert4x` – This program updates the Solaris 2.6 server with information about all the existing SunOS release 4.x clients. This command is used after issuing the `discover4x` and `install4x` commands. The updated information enables the existing SunOS release 4.x clients to work with the Solaris 2.6 server.

Before beginning any of these installation procedures, ensure that the SUNWhinst package is properly loaded. Use the `pkginfo`(1) command to generate a list of installed packages and then check the list to ensure that all necessary packages were installed, including the SUNWhinst package.

For details on adding and removing packages, see *System Administration Guide.*

Running `discover4x`

`discover4x` analyzes the support that remains for SunOS release 4.x clients after the server has migrated to the Solaris 2.6 operating environment.

As superuser (root), type the following.

```
# discover4x
```

The `discover4x` program runs from 1 – 60 seconds, depending on the amount of software examined.

`discover4x` may report messages such as the following.

```
Setting up proto root for sun4c arch
Updating server databases to include sun4c sunos 4.1.2 support
Support for sun4c clients must be added using install4x, if \
        sun4c clients are served by this machine.
```

If your site has completed a custom Solaris 2.6 installation that changed the location of the `/export` directory, `discover4x` examines that directory if you invoke it with the directory name as a single argument. For instance, if the `/export` software is stored in `/clients` directory, use the following command.

```
# discover4x /clients
```

Setting Up the CD-ROM Drive for `install4x`

Run the `install4x` program on a server with the Solaris 2.6 operating environment using one of the three procedures listed in the following section.

- If the system has a local CD-ROM drive, see “Using a Local CD-ROM Drive” on page 103
- If the system will use a remote CD-ROM drive on a system running the Solaris 2.6 operating environment, see “Using a Remote CD-ROM Drive (Solaris 2.6 Software)” on page 104
- If the system will use a remote CD-ROM drive on a system running the SunOS release 4.x software, see “Using a Remote CD-ROM Drive (SunOS Release 4.x Software)” on page 104

Insert the SunOS release 4.x CD into the CD-ROM drive before you proceed.

Using a Local CD-ROM Drive

If you are running `install4x` on a system with a local CD-ROM drive, after you install the CD into the drive, Volume Management automatically mounts the CD directory on `/cdrom/volume1/s0`.

Using a Remote CD-ROM Drive (Solaris 2.6 Software)

If `install4x` is to use a CD-ROM drive on a remote system running the Solaris 2.6 operating environment, after you install the CD into the drive, Volume Management automatically mounts the CD directory on `/cdrom/volume1/s0`. Then type the following command.

```
# share -F nfs -o ro /cdrom/volume1/s0
```

If you are not sharing other NFS systems at boot time, you need to invoke the `mountd`(1M) and `nfsd`(1M) daemons.

Type the following commands on the local system.

```
# mkdir /cdrom
# mount -F nfs -o ro cd-host:/cdrom/volume1/s0 /cdrom
```

Using a Remote CD-ROM Drive (SunOS Release 4.x Software)

If `install4x` is to use a CD-ROM drive on a remote system that is running the SunOS release 4.x software, type the following as superuser on the remote system.

```
# mkdir /cdrom
# mount -t hsfs -r /dev/sr0 /cdrom
```

Once you have typed the previous commands, edit the `/etc/exports` and insert the following line.

```
/cdrom -ro
```

Then type the following command on the remote system.

```
# exportfs /cdrom
```

Type the following commands on the local system.

```
# mkdir /cdrom
# mount -F nfs -o ro cd-host:/cdrom /cdrom
```

Running install4x

After you use one of the previous procedures, the CD is mounted on /cdrom. Now invoke install4x by typing the following command.

```
# /usr/sbin/install4x -m /cdrom/volume1/s0 -e /export
```

If the –m option is not specified, the following prompt is displayed.

```
Enter name of directory where the 4.1* cd is mounted [/cdrom]:
```

If the –e option is not specified, the following prompt is displayed.

```
Enter name of export directory [/export]:
```

As before, if your site has customized the location of the /export directory, you can direct install4x to load software to a different directory by specifying additional arguments, as in the following command.

```
# /usr/sbin/install4x -m /cdrom -e /clients
```

Choosing Software to Load

install4x displays the Install Main Menu shown here.

```
         *** 4.1* Install Main Menu ***

 Choose an Architecture (then select modules to load):

                                        Modules

                                Selected         Loaded

 [a] sun4.sun4c.sunos.4.1.2         8               0
 [b] sun4.sun4.sunos.4.1.2          8               0
 [c] sun4.sun4m.sunos.4.1.2         7               0

 or begin the loading process for all selected modules:

    [L] Load selected module            +----------------------+
                                        |  Disk Usage:         |
 or abort without loading any modules   |     0K Selected      |
                                        |   53634K Free        |
    [Q] Quit without loading            +----------------------+

 Type any bracketed letter to select that function.
```

(Continued)

(Continuation)

```
Type ? for help.
```

The Install Main Menu screen presents several options. The first set (labeled here as a, b, and c) is used to specify the architecture for which software is to be loaded. Other options enable the user to direct software loading to begin (L), quit the program (Q), or ask for help (?).

After you choose each appropriate architecture, the program displays the Module Selection.

```
Select sun4.sun4c.sunos.4.1.2 modules:
+[a] R proto root......240K  |  [o]   User_Diag...........6352K
+[a] R proto root......240K  |  [o]   User_Diag...........6352K
+[b] R usr...........26240K  |  [p]   Manual.............7456K
+[c] R Kvm............4832K  | +[q] D TLI..................48K
+[d] R Install.........936K  |  [r] D RFS..................912K
 [e] D Networking.....1040K  |  [s] D Debugging...........2928K
 [f] D System_V.......4008K  |  [t]   SunView_Programmers.1840K
 [g] D Sys............5288K  |  [u]   Shlib_Custom........1376K
 [h] C SunView_Users..2664K  |  [v]   Graphics............1784K
 [i]   SunView_Demo....512K  | +[w]   uucp.................608K
+[j]   Text............712K  | +[x]   Games...............3136K
 [k]   Demo...........4264K  |  [y]   Versatec............5960K
 [l] C OpenWin_Users.25936K  |  [z]   Security.............312K
 [m] C OpenWin_Demo...4288K  |  [A]   OpenWindows_Progr..10200K
 [n] C OpenWin_Fonts..7840K  |

Module     + = already loaded        R = Required    C= Common
Legend:   ** = selected for loading  D = Desirable   Others opt

Select [a-A] or a Quick-Pick Option:      +-------------------+
 [1] All Req'd Modules [4] All Opt Moduls | Disk Usage:       |
 [2] All Desr'ble  Mod  [5] All Modules   |    0K Selected    |
 [3] All Common Modules                   |   53634K Free     |
or [D] (done) to return to the main scrn  +-------------------+
```

Packages already loaded are shown on the Module Selection screen with a plus sign (+) before the selection letter (that is, in the previous screen the packages associated with letters a, b, c, d, j, q, w, and x are already loaded). Note that loading packages for one architecture may cause those packages to show as being loaded for other architectures since many packages are shared.

Select modules to load by typing the associated character that is shown in brackets. Pressing the key associated with a module toggles the selection status (that is, will select or deselect the module, depending on its previous

status). Modules selected to be loaded have asterisks (**) displayed before the selection character. You can reload modules already present by answering Y or y when asked to confirm the apparent redundancy.

SunSoft has determined which software must be loaded for a release to operate normally (shown with R to the right of the selection letter), which software is commonly loaded (shown as C), and which software should be loaded (shown as D).

Additionally, the Module Selection screen readily enables you to pick groups of modules to be loaded. When you enter a 1, it marks all required modules for loading. When you enter a 2, it marks all recommended modules. When you enter a 3, it marks all commonly loaded modules. When you enter a 4, it marks all optional modules. When you enter a 5, it marks all modules shown on the Module Selection screen.

Return to the Install Main Menu by typing D.

```
            *** 4.1* Install Main Menu ***

 Choose an Architecture (then select modules to load):

                                Modules
                           Loaded      Selected

[a] sun4.sun4c.sunos.4.1.2    8           0
[b] sun4.sun4.sunos.4.1.2     8           0
[c] sun4.sun4m.sunos.4.1.2    7           0

or begin the loading process for all selected modules:

 [L] Load selected modules                   +--------------------+
                                             |  Disk Usage:       |
or abort without loading any modules:        |    0K Selected     |
                                             |   53634K Free      |
[Q] Quit without loading                     +--------------------+

Type any bracketed letter to select that function.
Type ? for help.
```

By typing L on the Install Main Menu, you can load all selected modules. Output similar to the following is displayed.

```
Installing module ‘proto root’ [size: 248K]
        in directory /export/exec/proto.root.sunos.4.1.2 ...

Updating server databases ...
```

(Continued)

(Continuation)

```
Press any key to continue:
```

Running convert4x

convert4x updates the Solaris 2.6 server with information about all SunOS release 4.x clients. The following files and directories are updated when you run convert4x:

- /tftpboot – Directory containing network bootable images
- /etc/dfs/dfstab – File containing file systems exported via NFS
- /etc/inet.conf – File containing list of servers that inetd(1M) invokes when it receives an Internet request
- /etc/bootparams – File containing per-client boot specifications
- /etc/hosts – File containing IP-to-host name mapping

Before running convert4x, make certain that the Ethernet addresses are entered in the /etc/ethers file for the clients you are converting. This is necessary because convert4x invokes the rpc.rarpd(1m) daemon.

As Superuser, run convert4x by typing the following command.

```
# /usr/sbin/convert4x
```

Optionally, you can specify a single fully qualified path to the location to an alternate client hierarchy. By default, convert4x looks in /export.

As convert4x runs, it displays information on the screen about the actions taken by the script. It warns you if there are any discrepancies in client information. If there is insufficient information for a given client, convert4x reports the error and exits.

If the convert4x is successful for existing clients, you do not have to add them again using Solstice Host Manager.

CHAPTER 11

Managing Printers, Terminals, and Modems

This chapter describes how to manage printing, and the differences in print commands, in the Solaris 2.6 environment. It also describes serial port management (which enables terminal and modem connections) through Admintool or the Service Access Facility (SAF),

Printing

This section describes how to set up and administer printers after you install Solaris 2.6 software. This chapter also describes the changes to printer commands that have taken place between the SunOS release 4.x and the Solaris 2.6 release environments.

Summary of Printing Differences

The SunOS release 5.6 LP print service replaces the SunOS release 4.x printing facilities, which were provided by the `lpd` daemon and `lpr`, `lpq`, `lprm`, and `lpc` commands. Admintool enables youto set up and administer printers through a graphical user interface. You can also use a command-line interface for the LP print service to administer SunOS release 5.6 printers.

For detailed information about Admintool and the command-line interface to the LP service, see *System Administration Guide.*

The services provided by the `/etc/printcap` file in the SunOS release 4.x software are handled in the Solaris 2.6 operating environment by the `terminfo` database and by the files in the `/etc/lp` directory.

Print Commands and the Compatibility Package

You can still use many SunOS release 4.x print commands if the system is running the SunOS/BSD Source Compatibility Package. Compatibility mode uses SunOS release 4.x command names as an interface to underlying Solaris 2.6 LP print services and does not actually run them the way a SunOS release 4.x system would. When a user types SunOS release 4.x commands to set up printing or to print files from a Solaris 2.6 system, the commands create message files that are handled by the SunOS release 5.6 LP print service scheduler.

Solaris 2.6 printing provides additional capabilities not available in SunOS release 4.x systems. These capabilities enable you to control forms, print wheels, and interface programs, and to set up network print services.

Using Printer Commands

As discussed in a previous section, you can continue to use SunOS release 4.x print commands if you have the SunOS/BSD Source Compatibility Package. Table 11–1 shows the basic user print command equivalents.

TABLE 11–1 User Print Command Equivalents

SunOS release 4.x	SunOS release 5.6	Function
`lpr` *filename*	`lp` *filename*	Print a file to the default printer
`lpr –P` *printer filename*	`lp –d` *printer file*	Print a file to a specific printer
`lpq`	`lpstat –o` *printer*	Look at a list of the files waiting to print on the default printer

TABLE 11–1 User Print Command Equivalents *(continued)*

SunOS release 4.x	SunOS release 5.6	Function
Check `/etc/printcap`	`lpstat –d`	Determine which is the default printer
Check `/etc/printcap`	`lpstat –a`	Determine which printers are available
`lprm` *jobnumber*	`cancel` *jobid*	Cancel a print job on the default printer

Using SunOS Release 5.6 Printer Administration Commands

This section describes differences between printer setup and administration on SunOS release 4.x and Solaris 2.6 systems. All the underlying system services described are available only in the Solaris 2.6 operating environment. The SunOS release 4.x counterparts are not available even in compatibility mode.

You must use the System V printer administration commands, `lpadmin`(1M) and `lpsystem`(1M) instead. Use the `terminfo` database and the configuration files in the `/etc/lp` directory instead. See *System Administration Guide* for details.

Table 11–2 shows the command equivalents for setting up printing.

TABLE 11–2 Printer Administration, Setup, and File Equivalents

SunOS release 4.x	SunOS release 5.6	Function
`lpc`	`lpadmin`	Control line printer functions
`/etc/printcap`	`terminfo` database and `/etc/lp/printers/`*printername*`/*`	File that defines printer functions

TABLE 11–2 Printer Administration, Setup, and File Equivalents *(continued)*

SunOS release 4.x	SunOS release 5.6	Function
`/var/spool`	`/var/spool/lp`	Directory where printing system stores spool and lock files
Not available	`lpmove`	Move print queues between printers
`lpc down`	`reject`	Stop queueing to a printer

Printing `troff`

In the SunOS release 4.x software, you need the following command to send a troff file to the default printer.

```
% troff filename
```

In the Solaris 2.6 operating environment, you must specify that you want the file printed by piping (|) the output to the `lp` command. Table 11–3 shows the SunOS release 5.6 `troff` commands.

TABLE 11–3 SunOS release 5.6 `troff` Commands

SunOS release 5.6 Command	Function
`troff` *file* | `/usr/lib/lp/postscript/dpost|lp`	Sends to default printer that supports troff jobs
`troff` *file*| `/usr/lib/lp/postscript/dpost|lp –d` *printer*	Sends to a particular printer
`troff` *file* | `lp–Ttroff`	Sends to any printer that supports `troff` jobs

Serial Port Management

This section describes serial port management (which enables terminal and modem connections) through Admintool or the Service Access Facility (SAF).

System Administration Guide describes the details of Solaris 2.6 setup and installation procedures for serial devices.

Terminal and Modem Management

Admintool is a tool that readily enables you to set up and modify serial port software for terminals and modems.

Admintool provides:

- Templates for common terminal and modem configurations
- Multiple port setup, modification, or deletion
- Quick visual status of each port

This tool provides the capabilities of the Service Access Facility's `pmadm` command.

Service Access Facility (SAF)

Using SAF, you can manage access to all services in a similar way, whether they are on the network or attached only to local systems. SAF uses Service Access Control (SAC) commands to set up and manage services. It provides uniform access to system services, such as:

- Adding, removing, and modifying terminal line settings
- Adding, enabling, disabling, or removing a port monitor
- Printing information from administrative database files
- Using and administering port monitors
- Adding, enabling, disabling, and removing `listen`(1M) port monitors

In previous versions of SunOS operating systems, the method for controlling devices depended both on the device providing the access and on the location of that device. Managing user access involved editing many device files.

SAF helps isolate the system administrator from these device dependencies, and provides a common interface for managing a range of services, including the ability to:

- Log in (either locally or remotely)
- Access printers across the network
- Access files across the network

SAF's common interface consists primarily of two commands: `sacadm` and `pmadm`. The `sacadm` command controls daemons called *port monitors*. The `pmadm` command controls the services associated with the port monitors.

Controlling Port Monitors

SAF's common interface helps control services called port monitors. A *port monitor* is a program that continuously monitors for requests to log in or requests to access printers or files.

Once a port monitor detects a request, it sets whatever parameters are required to establish communication between the operating system and the device requesting service. Then the port monitor transfers control to other processes (for example, the `login` program) that provide the services needed.

There are two types of port monitors included in the Solaris 2.6 operating environment: `ttymon` and `listen`. The `listen` port monitor controls access to network services and handles remote print and file system requests. The `ttymon` port monitor provides access to the login services needed by modems and alphanumeric terminals.

SAF Functions and Related Programs

SAF's common interface consists primarily of two commands: `sacadm` and `pmadm`. The `sacadm` command controls the port monitors. The `pmadm` command controls the services associated with the port monitors.

The `sacadm` command enables you to add and remove port monitors. You can also use the `sacadm` command to list the status of a port monitor, and to administer configuration scripts for customizing port monitors.

Using the `pmadm` command, you can add or remove a service, and enable or disable a service. You can, for example, disable all remote logins with one `pmadm` command. You can also install or replace per-service configuration scripts, or display information about a service.

Using only the `sacadm` and `pmadm` commands, a system administrator has complete control over access to resources. However, these two commands are only the interface to the SAF suite of programs and processes that make the integrated management environment possible. The functions and associated programs are:

- Overall administration – `sacadm`
- Port Monitor Service Administrator – `pmadm`

- Service Access Control – `sac`
- Port monitors – `ttymon` and `listen`
- Services – logins, remote procedures

The service access control, `sac`, is the most important program in the SAF suite. It is launched by the `init` program when a machine is first started. In turn, `sac` starts all the port monitors listed in its administrative file.

For more information on the SAF in general, or on the different ways to use the `sacadm` and `pmadm` commands, see *System Administration Guide.*

CHAPTER 12

Network Service Administration

This chapter outlines changes to the network facilities, TCP/IP and UUCP.

Changes to TCP/IP

The user interface to TCP/IP is virtually the same as in previous releases of the Solaris software, but the administration of NIS+ maps is handled through AdminTool, which is different from the process in the SunOS release 4.x software and traditional AT&T SVR4.

The NIS+ maps administered by AdminTool include:

- Hosts
- Services
- RPC
- Ethers

When you are ready to configure SunOS release 5.6 TCP/IP facilities, see *TCP/IP and Data Communications Administration Guide* for information about setting up TCP/IP.

Changes to NFS

The Solaris 2.6 operating environment simplifies resource sharing with a new set of commands and files to administer NFS resources. In specific, `exportfs` and `/etc/exports` has been replaced by `share`, `shareall`, and `/etc/dfs/dfstab`.This new command set was designed to allow for future distributed file system types.

Several of the daemons associated with NFS have been renamed. `rpc.statd`, `rpc.lockd`, and `rpc.mountd` are now simply called `statd`, `lockd`, and `mountd`.

Unlike the SunOS release 4.x environment, there are no client side block I/O daemons (biods) in the Solaris 2.6 release. They have been superceded by kernel threads. Also, the NFS daemon, `nfsd`, has been altered so that it does not spawn multiple copies to handle concurrent requests.

Other features included in this release:

- NFS over TCP
- NFS Version 3
- Improved NFS Lock Manager
- Support for Access Control Lists (ACLs)
- WebNFS
- NFS Client Failover
- Kerberos support for NFS file systems
- NFS Large File Support

All these features are described in *NFS Administration Guide.*

PPP

PPP for Solaris 2.6 systems is an asynchronous implementation of the standard data link-level, point-to-point protocol (PPP) included in the internet protocol suite. PPP enables a network administrator to create a communications link using modems and telephone lines. See *TCP/IP and Data Communications Administration Guide* for detailed information about expanding your network with PPP.

UUCP

The Solaris 2.6 UNIX-to-UNIX Copy (UUCP) is similar to the HoneyDanBer UUCP available with SunOS release 4.x systems. It uses the same set of configuration files, scripts, and commands, so you should be able to restore most changes you made in SunOS release 4.x files and scripts to run with this release. However, the spool directory is organized differently in Solaris 2.6 due to *job grades*, a mechanism to help sort and prioritize the work load.

Table 12–1 describes the new files and commands offered with Solaris 2.6 UUCP that were not part of the SunOS release 4.x implementation. Table 12–2 describes the log files added to Solaris 2.6 UUCP.

TABLE 12–1 New SunOS release 5.6 UUCP Files and Commands

Command or File	Description
`D.` data files `P.` data files	These data files are created when a UUCP command line specifies copying the source file to a spool directory. All data files have this format: *systmxxxxyyy*. *systm* is the first five characters in the name of the remote system. *xxxx* is a four-digit job sequence number assigned by UUCP. *yyy* is a subsequence number used to distinguish between several `D.` files created for a work (`C.`) file.
`/etc/uucp/Grades`	Maps text grade names to system names.
`/etc/uucp/Limits`	Specifies the number of concurrent UUCP sessions that can occur. Replaces `Maxuuscheds` and `Maxuuxqts` files in previous versions.

TABLE 12–1 New SunOS release 5.6 UUCP Files and Commands *(continued)*

Command or File	Description
`/etc/uucp/Config`	Contains information to override UUCP parameters that can be tuned. Currently, the only parameter of this type is `Protocol`, so system administrators normally will not have to modify this file.
`uuglist`	Prints the list of service grades available on the system to use with the `–g` option of `uucp`(1C) and `uux`(1C).

Solaris 2.6 UUCP includes a few additional features that can affect system administration:

- Checkpoint-restart facilities
- Job grades that control UUCP transmission
- Two new configuration files to limit the number of concurrent UUCP sessions that the system can run, and to override UUCP parameters that can be tuned

The following sections describe the system administration differences made by each of these additions.

Checkpoint Restart

When communication link failures interrupt UUCP transmissions between SunOS release 4.x systems, the transmission starts again from the beginning of the file as soon as communication resumes. Communication between two systems running Solaris 2.6 UUCP resumes where it was interrupted instead of restarting at the beginning. This makes better throughput possible, especially on erratic or noisy transmission lines.

The systems use two new files to store sent and received data and to compare the sizes of the files to determine where to restart transmission. The systems use `.P` files to store received data and `.D` files to store transmitted data. These files replace the `TM.` files of previous UUCP versions. If only one system is running SunOS release 5.6 UUCP, no comparison can take place and transmissions restarts from the beginning.

User Job Grades

Job grading enables administrators to divide jobs into work loads that compete against others of similar size, type, priority, or all three. You can sort work loads using any one or a combination of these factors. You can also set access permissions allowing users and groups to obtain each grade of UUCP service.

In the SunOS release 4.x software, the user has to choose the grade when the job is submitted. Grades are a single letter, not a name, as they are in the Solaris 2.6 operating environment. Solaris 2.6 systems enable administrators to define job grades for an entire site.

`Limits` File

The `/etc/uucp/Limits` file specifies the maximum number of concurrent `uucico`, `uuxqt`, and `uusched` processes permitted on a system. This single file replaces the `Maxuusched` and `Maxuuxqt` parameters on previous releases.

`Config` File

The `/etc/uucp/Config` file contains information to override UUCP parameters that can be tuned. Currently the only parameter available is `Protocol` and it should normally not be altered by system administrators.

Log Files

Solaris 2.6 UUCP provides four log files in addition to the four supplied in previous versions. These files record accounting, command, performance, and security information. The command and security log files are created if they do not exist. The accounting and performance log files are written only if they already exist.

TABLE 12–2 Additional SunOS release 5.6 UUCP Log Files

File Name	Function
`/var/uucp/.Admin/account`	Records account information for billing
`/var/uucp/.Admin/perflog`	Records statistics on `uucico` operations

TABLE 12–2 Additional SunOS release 5.6 UUCP Log Files *(continued)*

File Name	Function
`/var/uucp/.Admin/security`	Records attempted security violations
`/var/uucp/.Admin/command`	Records information on commands issued by users or administrators

When you are ready to set up and use SunOS release 5.6 UUCP, see *TCP/IP and Data Communications Administration Guide* for complete information.

CHAPTER 13

Using Name Services

The network information service (NIS), which is part of the SunOS release 4.x environment, is widely being replaced with the *network information service plus* (NIS+). NIS+, introduced with the SunOS 5.0 system, is a completely redesigned name service that takes into account changes in customer client/server environments. DNS (domain name system) is an existing, complementary name service used for intercompany Internet communication. This chapter discusses NIS+ and compares it to NIS and DNS.

For more information about planning an NIS+ upgrade and installing NIS+, see *NIS+ Transition Guide* and *NFS Administration Guide.*

Note - The system administration documentation set for the Solaris 2.6 operating environment emphasizes a system that is using NIS+.

Name Service Switch

The Solaris 2.6 operating environment uses standard naming interfaces (for example, `gethostbyname`) to support multiple naming services (such as NIS, NIS+, and DNS, among others), thereby allowing applications to access data transparently from different services. One instance of this is the *Name*

Service Switch capability in the Solaris 2.6 operating environment, which allows applications to use a UNIX standard naming interface (for example, `get`*xx*`by`*yy* interfaces). See the `nsswitch.conf`(4) man page for more information.

NIS+

NIS+ is a name service built on top of the ONC transport-independent remote procedure call (TI-RPC) interface. NIS+ has significant advantages over NIS in the areas of security, performance, scalability, and administration.

DNS

DNS supports the model of a hierarchical name space with autonomously administered name servers. Although NIS+ uses a similar hierarchical naming model, it focuses on supporting changing system administration data and other requirements of enterprise networks.

DNS and NIS+, therefore, are complementary name services:

- DNS is used for intercompany communication
- NIS+ supports administration of enterprise networks

DNS and NIS+ Comparison

Table 13–1 shows the features and benefits of DNS compared to NIS+.

TABLE 13–1 DNS and NIS+ Features and Benefits Compared

Feature	DNS	NIS+
Security	Unrestricted access to data	All operations can be authenticated
		Administrator designates access rights for objects and entries

TABLE 13–1 DNS and NIS+ Features and Benefits Compared *(continued)*

Feature	DNS	NIS+
API and human interface	Allows read-only access to name service	Allows read-write access to name service. Provides: - Efficient support of changing network environment - API support of administrative operations - Support of administrative and other distributed applications
Updating	By transfer of zone master files	By incremental data transfer - Fast support of changing network environments - Stronger consistency
Compatibility with NIS	Not applicable	Existing NIS applications can migrate smoothly
Data support	ASCII data only with packet size restriction	Binary and ASCII data. Provides: - Support of variable information - Support of larger objects

The main strength of DNS is supporting hierarchical database partitions and replicas containing entries of relatively static information (such as host name and IP address). DNS enables you to access the Internet.

NIS+, in contrast, is a secure repository of changing administrative information (such as email aliases, Ethernet addresses, RPC program numbers) for enterprise networks.

NIS and NIS+ Comparison

Table 13–2 summarizes several major enhancements in NIS+ compared to NIS.

TABLE 13–2 NIS and NIS+ Features Compared

Feature	NIS	NIS+
Name space	Has a flat ; is non-hierarchical organization; centralized flat file database for each independent network domain	Has a hierarchical organization; partitioned into directories to support each network subset or autonomous domain
Data Storage Scheme	Multiple bicolumn “maps” (files) having key-value pairs	Multicolumn database with multiple, searchable columns
Resource Access Across Domains	Not supported	Permitted for authorized users
Privileges for Updating	Updates require superuser privileges on master server	Updates can be performed remotely by authorized users
Update Process	Updates require using make files on master servers	Updates are performed easily through command-line interface
Update Propagation	Is administrator initiated and requires transfer of whole maps	Automatic and high-performance updating via incremental transfer
Security	Database not secure	Fine-grained access control to NIS+ directories, table column, and entries
Commands and Functions Prefixes	Prefixed by the letters *yp*, as in `ypmatch`(1) and `ypcat`(1)	Prefixed by the letters *nis*, as in `nismatch`(1) and `nischown`(1)

NIS+ includes features that enable NIS sites to migrate to the new name service in a smooth, phased manner. NIS sites that migrate to NIS+ will gain the following benefits:

- Distributed and remote administration of network domains by authorized users
- Support for hierarchical domains
- Fast and automatic propagation of updates from master to replica servers
- Fine-grained access to tables and network resources
- Easier and more consistent administrative operations
- Increased naming service reliability and availability

Planning an NIS+ Upgrade

NIS+ supports the following combinations of systems:

- SunOS release 5.6 software installed on all servers and clients
- SunOS release 5.6 software installed on one server, but combined with some SunOS release 4.x servers

For a network, there are three main migration paths from NIS to the NIS+ name service:

- Upgrade all servers and clients to NIS+
- Upgrade all servers at once to NIS+ and enable its compatibility mode to support SunOS 4.x clients
- Use different domain names so NIS and NIS+ can coexist

The first step to upgrading your network is to decide which servers to upgrade to the NIS+ name service and which servers can continue to run NIS. See *NIS+ Transition Guide* for more information.

CHAPTER 14

Solaris Common Desktop Environment

The Solaris Common Desktop Environment (CDE), compatible among various workstation manufacturers, provides users with a desktop graphical interface on a Sun™ Workstation™ running Solaris 2.4 software or later. This window environment helps you organize and manage your work. The desktop provides windows, workspaces, controls, menus, and a Front Panel. When you login to your windows environment the first time, you have a choice of using either OpenWindows or Solaris CDE as your default desktop.

What Is the Solaris Common Desktop Environment?

In March of 1993, Sun, Hewlett–Packard, IBM and Novell announced an agreement to develop a graphical user interface that would bring a consistent look and feel to major UNIX system-based workstations and desktop computers. From the start, the CDE development effort was guided by one goal: to make UNIX easier to use for end users and application developers.

The result of this joint development effort is the Common Desktop Environment. CDE is one of two desktops packaged with the Solaris 2.6 environment (the other is the OpenWindows desktop). Over time, CDE will

emerge as the standard desktop for Sun, Hewlett–Packard, IBM, Novell and many others in the UNIX workstation market.

Solaris CDE includes a desktop server, a Session Manager, a Window Manager (based on Hewlett–Packard's Visual User Environment), and numerous desktop utilities.

To learn how to use Solaris CDE, see *Solaris Common Desktop Environment: User's Guide.*

Developers, End Users, and CDE

Because CDE provides a consistent computing environment across major UNIX platforms, end users can easily move between different machines. CDE also aids application development by supplying a single, standard set of programming interfaces for any conforming Sun, HP, IBM, and Novell platform. A single API enables developers to create applications that are consistent in appearance and behavior across CDE-compliant systems.

The CDE development environment is based on the X11R5 server and produces applications with a look and feel based on the Open Software Foundation's Motif 1.2 specification.

Overview of the Desktop

Some of the features of the Solaris CDE desktop include:

- The Front Panel
- Style Manager
- File Manager
- Application Manager

Front Panel

The Front Panel is a special window at the bottom of the display. It provides controls, indicators, and subpanels you use in your everyday work. The Front Panel also provides the workspace switch for selecting a workspace.

Many controls in the Front Panel, such as the File Manager control, start applications when you click them. Some controls, like the Printer control, are also drop zones. You can drag a file icon from File Manager and drop it on the Printer control to be printed.

Arrow buttons over Front Panel controls identify subpanels—click an arrow button to open a subpanel.

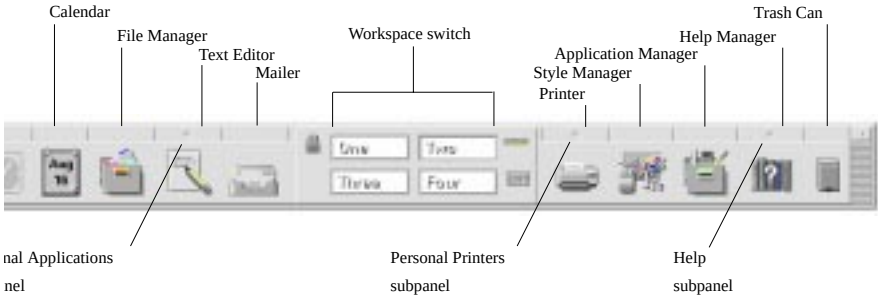


Figure 14–1 Front Panel controls

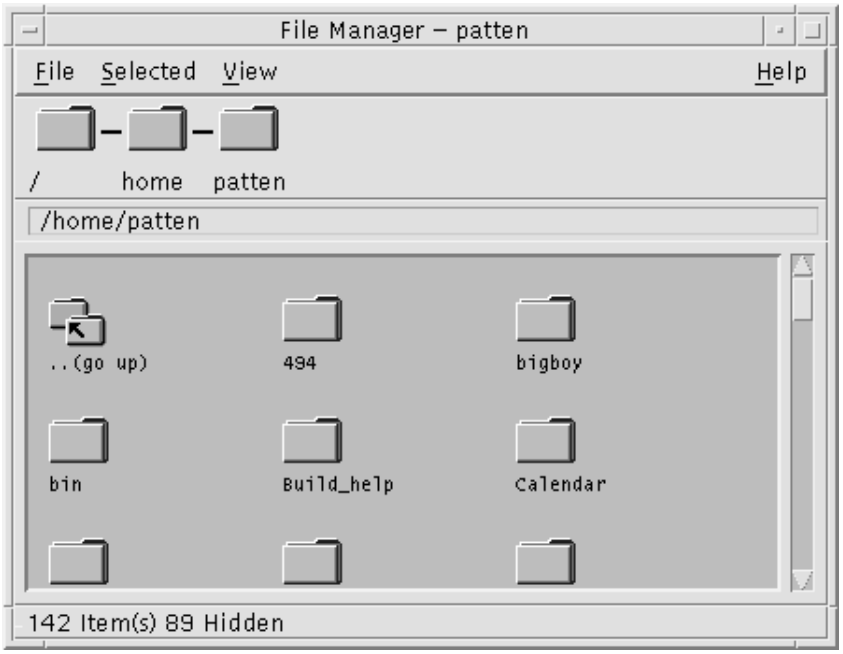
Style Manager

Click this icon: 

To use Style Manager to easily customize many elements of the desktop including:

- Colors
- Workspace backdrops
- Font size
- Keyboard, mouse, and window behavior

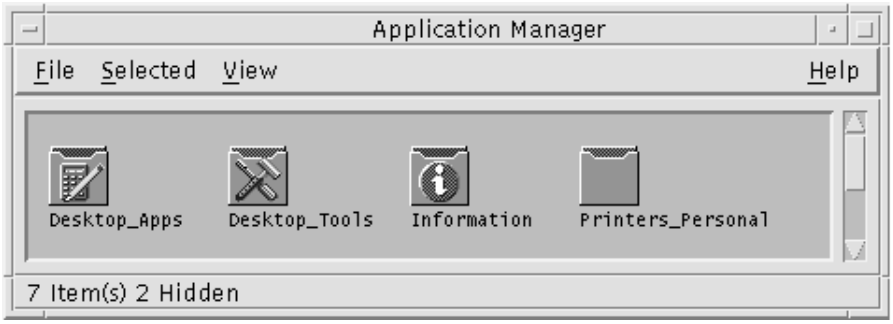
File Manager



Click the File Manager icon: 

To display the files, folders, and applications on your system as icons.

Application Manager



Click this icon to open the Application Manager window: 

Application Manager provides access to applications you use in your everyday work through *action icons.* You use action icons to start applications. Application Manager stores action icons in special folders called *application groups.*

You can place the action icons you use frequently on the workspace backdrop.

Moving From the OpenWindows Environment to CDE

With the Solaris 2.6 software you can choose to log in to either the OpenWindows desktop or the CDE desktop from your login screen. For more information on logging in, refer to the Login Manager help volume or Chapter 2, “Starting a Desktop Session,” in *Solaris Common Desktop Environment: User’s Guide.*

Desktop Services

Some of the desktop services you are used to using in the OpenWindows environment are located in different places in Solaris CDE. Table 14–1 highlights some of the differences.

TABLE 14–1 Location of Desktop Services

Desktop Service	OpenWindows	CDE
Logout	Workspace menu	Front Panel
LockScreen	Utilities menu	Front Panel
Customize Workspace	Workspace menu	Style Manager
Save Workspace	Utilities menu	Style Manager

TABLE 14–1 Location of Desktop Services *(continued)*

Desktop Service	OpenWindows	CDE
Refresh	Utilities menu	Front Panel
Properties	Workspace menu	Style Manager
Help	Workspace menu	Front Panel, Application Manager, Workspace menu

Using Windows, Menus, Buttons, and the Mouse in CDE

Windows, menus, buttons and the mouse are used in Solaris CDE slightly differently than in the OpenWindows environment. For complete information on using window, menus, buttons and the mouse, refer to Chapter 1, “Basic Skills,” in *Solaris Common Desktop Environment: User’s Guide.*

Accessing the Workspace Programs Menu

In the OpenWindows environment, the main way to start an application was through the Workspace menu. A Workspace menu still exists in Solaris CDE; however, the main access point for workspace functionality is the Front Panel.

The applications available through the Workspace menu include the items on the Front Panel and also a subset of the applications available to you within Application Manager. Refer to Chapter 6, “Running Applications from the Desktop,” in *Solaris Common Desktop Environment: User’s Guide* for complete information on Application Manager.

Style Manager and Customizing the Workspace

The items available through Style Manager are: Color, Font, Backdrop, Keyboard, Mouse, Audio, Screen, Window, and Startup. This replaces the Workspace Properties window in the OpenWindows environment. For complete information on Style Manager, refer to Chapter 7, “Customizing the

Desktop Environment," in *Solaris Common Desktop Environment: User's Guide*.

Running OpenWindows Applications in CDE

A folder in CDE Application Manager, titled OpenWindows, contains your OpenWindow applications.

If you ran OpenWindows applications from the command line, you can run them the same way from the terminal emulator (Terminal application) in Solaris CDE. Refer to Chapter 6, "Running Applications from the Desktop," in *Solaris Common Desktop Environment: User's Guide* for complete information on Application Manager.

Application Settings and Properties

In the OpenWindows environment, application-wide settings are set via the Properties dialog box, accessed from the Edit menu. In CDE, application-wide settings are set via the Options areas. Options choices are generally located under the application's File menu or the separate menu item, Options.

In CDE, Properties (if they exist in an application) are found under the application's Edit menu and are used to set characteristics of an object, such as its date or name, or display identifying characteristics of an object, such as typefaces. In CDE, format settings are usually found under the Format menu and enable margin and paragraph alignment to be set for a single paragraph, file, or message.

CDE Global options are like the properties you set from the Workspace menu in the OpenWindows environment. You now set these properties from the CDE Style Manager application. See Chapter 7, "Customizing the Desktop Environment," in *Solaris Common Desktop Environment: User's Guide*.

Changing Keyboard Defaults

If you did not change your keyboard defaults in the OpenWindows environment they should stay the same within CDE. If you want to change your defaults, use the Style Manager Keyboard dialog box. See Chapter 7, "Customizing the Desktop Environment," in *Solaris Common Desktop Environment: User's Guide*. If you need to make changes to your UNIX keyboard bindings, refer to Chapter 10, "Using Text Editor," in *Solaris Common Desktop Environment: User's Guide*.

Changing Mouse Defaults

If you did not change your mouse defaults in the OpenWindows environment they should stay the same within CDE. If you want to change your defaults, use the Style Manager Mouse dialog box. Some of the names have been changed for the functions: You still have double-click, acceleration, and threshold. Mouse button order in CDE is called "handedness. See Chapter 1, “Basic Skills,” in *Solaris Common Desktop Environment: User’s Guide*.

PART II Transition Information for Developers

Changes in the C language and its related tools are among the most obvious differences between the SunOS release 4.x and the Solaris 2.6 operating environments. These changes affect all developers to varying degrees. The operating system kernel and its interfaces have also changed significantly since the SunOS release 4.xsoftware. This part of the guide describes these differences, points out similarities between releases, provides information you need to port existing software, or to write new software for the Solaris 2.6 operating environment, and explains the implications for your programming environment.

CHAPTER 15

Compilers, Linkers, and Debuggers

This chapter discusses the changes to compilers, linkers, and debuggers.

Compilers

The single most significant change for developers migrating from the SunOS release 4.x to the Solaris 2.6 operating environment is the unbundling of the C compiler. One of the factors that allowed the compiler to be unbundled is the dynamic kernel. The compiler is no longer required to reconfigure the kernel as devices are now automatically loaded by the kernel as needed.

An ANSI C compiler is available with unbundled Sun WorkShop™. This compiler produces executables in executable and linking format (ELF), the native object format of Solaris 2.6 executables. `lint` and the `lint` libraries are also unbundled.

Source Compatibility Guide describes the differences between the C language as implemented by the SunOS release 4.x C compiler (Sun C) and as described by the ANSI Programming Language C document (ANSI C). These differences should be addressed when porting source written for the Sun C compiler to an ANSI C conforming compiler.

Compiler Option Differences

Table 15–1 compares the SunOS release 4.x Sun C and ANSI C compilers to the Solaris 2.6 Sun WorkShop™ C++ 4.2. It is provided for your information, and is not intended to describe the range of operations available in default Solaris 2.6 software.

Note - Table 15–1 presents information based on the publicly available copies of the American National Standard for Information Systems - Programming Language XX3.159-1989, approved December 14, 1989.

Please note the following:

- A “yes” in any column indicates that the option is supported by that compiler.
- If the option has changed, the new option is listed.
- A “no” in any column means the option is not supported by that driver
- A “+” in the Sun WorkShop™ C++ 4.2 Solaris 2.6 column indicates an option required by SVID specifications.

TABLE 15–1 Comparison of C Compiler Options

Option or Flag	Sun C	ANSI C	Sun WorkShop™ C++ 4.2	Description
–A symbol	No	Yes	Yes	`cpp` predicate assertion
`–a`	Yes	Yes	`–xa`	Counts # basic block executions
`–align`	Yes	Yes	No	Page aligns (`ld`)
`–assertx`	Yes	Yes	`–z`	Specifies link-time assertion
`–BX`	Yes	Yes	Yes	Specifies binding type (only for libraries in SunOS release 5.6)
`–bnzero`	Yes	Yes	No	Generates nonzero AR
`–C`	Yes	Yes	No	`cpp` comments left in

TABLE 15–1 Comparison of C Compiler Options *(continued)*

–c	Yes	Yes	+Yes	Produces .o file
–cg87	Yes	Yes	No	Sets fp option to –cg87
–cg89	Yes	Yes	Yes	Sets fp option to –cg89
–dryrun	Yes	Yes	Yes	Shows commands constructed by driver
–Dx	Yes	Yes	+Yes	Defines cpp symbol x
–d	Yes	Yes	No	Forces definition of common (ld)
–dalign	Yes	Yes	Yes	Assumes doubles are double-word aligned
–dl	Yes	Yes	No	Generates long data segment refs
–d[y|n]	–BX	–BX	Yes	Dynamic linking [yes|no]
–E	Yes	Yes	+Yes	Runs source through cpp
–e	No	No	No	Entry point for ld
–F	–O*	–O*	+No	Optimization directives
–f	No	No	No	Floating-point support
–fast	Yes	Yes	Yes	Options for best performance
–fsingle	Yes	No	No	Floats are single precision
–fsingle2	Yes	No	No	Passes float (as float not double)
–fnonstd	Yes	Yes	Yes	Non-standard float option

TABLE 15–1 Comparison of C Compiler Options *(continued)*

–fstore	Yes	No	Yes	Forces writes on store
–G	No	No	Yes	Creates shared library, not available with the –dn option
–g	Yes	Yes	+Yes	Generates info for dbx
–go	Yes	No	No	Generates info for adb
–H	Yes	Yes	Yes	Prints paths of included files
–h name	No	No	Yes	Uses name as internal identifier; soname passed to linker
–help	Yes	Yes	–flags	Lists options
–Ix	Yes	Yes	+Yes	Adds x to include path
–J	sun3	No	No	Generates long offset for switch|case
–KPIC	–PIC	–PIC	Yes	Position independent code
–Kpic	–pic	–pic	Yes	PIC with short offsets
–Kminabi	No	No	No	ABI compliant code
–libmil	Yes	Yes	–xlibmil	Passes libm.il as part of –fast
–lx	Yes	Yes	Yes	Reads object library (for ld)
–Lx	Yes	Yes	Yes	Adds x to ld library path
–M	Yes	Yes	No	Collects dependencies
–M *mapfile*	Yes	No	No	Passes *mapfile* to the linker

TABLE 15–1 Comparison of C Compiler Options *(continued)*

`–misalign`	Yes	Yes	Yes	Handles misaligned Sun-4 data
`–N`	Yes	No	No	Does not make shared
`–n`	Yes	No	No	Makes shared
`–native`	Yes	Yes	Yes	Uses appropriate `–cg` option
`–noc2`	Yes	Yes	No	Doesn't do peephole optimization
`–nolibmil`	Yes	Yes	`–xnolibmil`	Doesn't pass `libm.il` with `–fast`
`–o` file	Yes	Yes	+Yes	Sets name of output file
`–O[1,2,3,4]`	Yes	Yes	`–x0[1,2,3,4]`	Generates optimized code
`–O`	Yes	Yes	+Yes	Generates optimized code
`–P`	Yes	Yes	+Yes	Runs source through `cpp`, output to `.i`
`–PIC`	Yes	No	`–KPIC`	Generates pic code with long offset
`–p`	Yes	Yes	+Yes	Collects data for prof
`–pg`	Yes	Yes	`–xpg`	Collects data for gprof
`–pic`	Yes	Yes	`–Kpic`	pic code with short offset
`–pipe`	Yes	No	No	Uses pipes instead of temp files
`–purecross`	Yes	No	No	Doesn't have slash in VROOT
`–Qdir x`	Yes	Yes	`–Y*`	Looks for compiler passes in x

TABLE 15–1 Comparison of C Compiler Options *(continued)*

–Qpath x	Yes	Yes	–Y*	Same as –Qdir
–Qn	No	No	Yes	Doesn't add version stamp info
–Qy	No	No	Yes	Adds version stamp info
–qdir x	Yes	Yes	No	Looks for compiler passes in x
–ql	No	No	No	Collects data for lprof
–qp	–p	–p	–p	Collects data for prof
–qpath x	Yes	Yes	No	Same as –Qdir
–Qoption cpp x	Yes	Yes	Yes	Passes option x on to program cpp
–Qoption iropt x	Yes	Yes	Yes	Passes option x on to program iropt
–Qoption cg x	Yes	Yes	Yes	Passes option x on to program cg
–Qoption inline x	Yes	Yes	No	Passes option x on to program inline
–Qoption as x	Yes	Yes	No	Passes option x on to program as
–Qoption asS x	Yes	Yes	No	Passes option x on to program asS
–Qoption ld x	Yes	Yes	Yes	Passes option x on to program ld
–qoption cpp x	Yes	Yes	No	Passes option x on to program cpp
–qoption ccom x	Yes	Yes	No	Passes option x on to program ccom

TABLE 15–1 Comparison of C Compiler Options *(continued)*

`–qoption lintl x`	Yes	Yes	No	Passes option x on to program `lint1`
`–qoption iropt x`	Yes	Yes	Yes	Passes option x on to program `iropt`
`–qoption cg x`	Yes	Yes	Yes	Passes option x on to program `cg`
`–qoption inline x`	Yes	Yes	No	Passes option x on to program `inline`
`–qoption cat x`	Yes	Yes	No	Passes option x on to program `cat`
`–qoption c2 x`	Yes	Yes	No	Passes option x on to program `c2`
`–qoption as x`	Yes	Yes	No	Passes option x on to program `as`
`–qoption asS x`	Yes	Yes	No	Passes option x on to program `asS`
`–qoption ld x`	Yes	Yes	Yes	Passes option x on to program `ld`
`–Qproduce .o`	Yes	Yes	Yes	Produces type `.o` file (Object file)
`–Qproduce .s`	Yes	Yes	Yes	Produces type `.s` file (Assembler source)
`–Qproduce .c`	Yes	Yes	No	Produces type `.c` file (C source)
`–Qproduce .i`	Yes	Yes	Yes	Produces type `.i` file (C source after `cpp`)
`–qproduce .o`	Yes	Yes	Yes	Produces type `.o` file (Object file)
`–qproduce .s`	Yes	Yes	Yes	Produces type `.s` file (Assembler source)
`–qproduce .c`	Yes	Yes	No	Produces type `.c` file (C source)

TABLE 15–1 Comparison of C Compiler Options *(continued)*

`–qproduce .i`	Yes	Yes	Yes	Produces type `.i` file (C source after `cpp`)
`–r`	Yes	Yes	No	Makes relocatable; pass to linker
`–R`	Yes	Yes	No	Merges data into text segment
`–R`	No	No	Yes	Specifies search directories for the run-time linker
`–S`	Yes	Yes	+Yes	Produces `.s` file only
`–s`	Yes	Yes	Yes	Strips (4.1); pass to linker
`–sb`	Yes	Yes	`–xsb`	Collects information for code browser
`–strconst`	No	Yes	No	Places string literals in read-only text segment
`–sun2`	Yes	No	No	Generates code for a Sun-2™ system
`–sun3x`	Yes	No	No	Generates code for a Sun-3x™ system
`–sun386`	Yes	No	No	Generates code for a Sun386i™
`–sun3`	Yes	No	No	Generates code for a Sun-3 system
`–sun4c`	Yes	No	No	Generates code for a Sun4c™ system
`–sun4`	Yes	No	No	Generates code for a Sun-4™ system
`–target`	Yes	No	`–xtarget`	Sets target architecture to x
`–temp=dir`	Yes	Yes	Yes	Set directory for temps to dir
`–time`	Yes	Yes	Yes	Reports the execution times

TABLE 15–1 Comparison of C Compiler Options *(continued)*

–u	Yes	Yes	No	Enters symbol arg as undef (ld)
–Ux	Yes	Yes	+Yes	Undefines cpp symbol x
–v	Yes	Yes	Yes	Verbose mode
–v	No	No	Yes	Strict semantic checking
–V	Yes	Yes	+Yes	Reports versions of programs
–W	No	No	No	Arguments to other components
–w	Yes	Yes	Yes	Does not print warnings
–X[t,a,c,s]	No	Yes	No	Compatibility options
–Y	No	No	No	Changes path name to components
–yx	Yes	No	No	Traces symbol
–z	–assert	–assert	No	Turns on asserts in linker
–#	–v	–v	No	Verbose mode
–EOF	No	No	No	File argument

The C compiler accepts the types of file-name arguments shown in Table 15–2.

TABLE 15–2 File-name Extensions Used by the C Compiler

Suffix	File Type
`.a`	Object library
`.il`	In-line expansion file
`.o`	Object file
`.so`	Shared object
`.s`	Assembler source
`.S`	Assembler source for `cpp`
`.c`	C source
`.i`	C source after `cpp`

"file.X=.Y" will read the file "file.X" but treat it as if it had suffix "Y"

Linkers

There are several changes to the link editor, `ld`(1), in this release. The most important change is its ability to handle the new ELF native file format.

Note - The recommended method for building libraries and executables is through the compiler driver rather than by invoking the linker directly. The compiler automatically supplies several files needed by the linker.

Link Editor Option Differences

Some options have been renamed in the new linker, some have remained the same, and others are no longer needed. Table 15–3 compares the SunOS release 4.x `ld` command to the Solaris 2.6 `ld` command.

The sections following Table 15–3 explain how certain linking tasks are affected by the option differences.

TABLE 15–3 Comparison of `ld` Options

SunOS release 4.x Option	Solaris 2.6 Replacement	Notes
–`align` *datum*	–`M` *mapfile*	Uses *mapfile* and distinct sections
–`assert definitions`	Default	
–`assert nodefinitions`	–`znodefs`	Issues a fatal error instead of a warning
–`assert nosymbolic`	–`zdefs`	Issues a fatal error instead of a warning
–`assert pure-text`	–`ztext`	Issues a fatal error instead of a warning
–A name	No replacement	`dlopen`(3X) and `dlclose`(3X) can approximate this behavior
–`Bdynamic`	–`Bdynamic`	Applies only to the inclusion of shared libraries; use –`dy` (the default) to build dynamically linked executables. See "Building Executables" on page 151.
–`Bnosymbolic`	–`zdefs`	
–`Bstatic`	–`dn` & –`Bstatic`	The –`dn` option must be specified to completely eliminate the dynamic linker. Use –`Bstatic` in dynamic mode to include archive libraries. (Used as a toggle. See "Building Executables" on page 151.)
–`Bsymbolic`	–`Bsymbolic`	Also gets –`assert nosymbolic` with this option
–`d` –`dc` –`dp`	Default	Use –`b` option in SVR4 to turn off

TABLE 15–3 Comparison of `ld` Options *(continued)*

`–D hex`	`–M` *mapfile*	*mapfile* contains different mechanisms to accomplish desired effect
`–e` *entry*	`–e` *entry*	
no `–e`	`–G`	Creates a shared object
`–lx[.v]`	`–lx`	Only major number versioning of shared libraries is currently supported
`–Ldir`	`–Ldir`	*dir* not recorded in executable; use `–R` option instead.
`–M`	`–m`	
`–n`	Default	SVR4 executable format compresses disk image as `–n`
`–N`	No replacement	
`–o` *name*	`–o` *name*	
`–p`	Default	Can override with `–M` *mapfile*
`–r`	`–r`	
`–S`	No replacement	
`–s`	`–s`	
`–t`	No replacement	
`–T` *hex*	`–M` *mapfile*	*mapfile* contains different mechanisms to accomplish desired effect

TABLE 15–3 Comparison of `ld` Options *(continued)*

–Tdata *hex*	–M *mapfile*	*mapfile* contains different mechanisms to accomplish desired effect
–u *name*	–u *name*	
–x	No replacement	
–X	No replacement	
–y *sym*	No replacement	
–z	Default	SVR4 executable format demands pages as –z

Building Shared Libraries

The procedure for building shared librariesin the Solaris 2.6 operating environment requires the –G option. In the SunOS release 4.x software, the linker would infer that a shared library was being built by the absense of the –e option. As shared libraries may have entry points, this option can no longer be used.

Building Executables

The –Bdynamic and –Bstatic options are still available, but their behavior is different. These options now refer to library inclusions to the executable rather than binding. Executable binding is set exclusively with the new –dy and –dn options in the Solaris 2.6 software. The –dy option is the default. It is required to create a dynamically linked executable. The –dn option is required to create a statically linked executable.

The –Bdynamic and –Bstatic options apply only when using the –dy option. –Bdynamic tells the link editor to include shared libraries, while –Bstatic tells it to include archive libraries. These options act as a toggle governing subsequent –l arguments until the next –Bdynamic or –Bstatic option is encountered.

The following examples show SunOS release 4.x and Solaris 2.6 commands that can be used to create similar executables.

- sunos4.1% `ld –Bstatic test.o –lx`

  Uses `libx.a` and creates a *static* executable

- sunos5.x% `cc –dn test.o –lx`

  Uses `libx.a` and creates a *static* executable

- sunos4.1% `ld –Bdynamic test.o –lx`

  Uses `libx.so` and creates a *dynamic* executable

- sunos5.x% `cc test.o –lx`

  Uses `libx.so` and creates a *dynamic* executable

- sunos4.1% `ld –Bdynamic test.o –Bstatic –lx`

  Uses `libx.a` and creates a *dynamic* executable

- sunos5.x% `cc test.o –Bstatic –lx`

  Uses `libx.a` and creates a *dynamic* executable

Specifying Library Search Paths

In the SunOS release 4.x software, directories specified with the `–L` option were searched at link time and the information retained for use at execution time. This behavior is now divided between the `–L` and `–R` options. The `–L` option specifies the directories to search at link time; the `–R` option tells the linker the search paths to be retained for use at run time. See “Search Path Rules” on page 152, in the next section for more information.

As with the `–Bdynamic` and `–Bstatic` options, the position of the `–L` option has significance; it applies only to the subsequent `–l` options.

Search Path Rules

The dynamic linker and the runtime linker determine their search paths through a different algorithm from that used by the SunOS release 4.x linker.

The examples below compare the search paths for the dynamic linker and the runtime linker for SunOS release 4.x and the Solaris 2.6 operating environment. Notice that in the latter, the search path for the link editor and the runtime linker are affected by the `LD_LIBRARY_PATH` setting.

SunOS release 4.x linker search paths:

- Link Editor: `–L`, `LD_LIBRARY_PATH`, `/usr/lib`, `/usr/local/lib`

- Runtime Linker: `LD_LIBRARY_PATH`, `–L`, `/usr/lib`, `/usr/local/lib`

Solaris 2.6 linker search paths (with `LD_LIBRARY_PATH=`*dirlist1*):

- Link Editor: `–L`, *dirlist1*, `/usr/ccs/lib`, `/usr/lib`
- Runtime Linker: *dirlist1*, `–R`, `/usr/lib`

Solaris 2.6 linker search paths (with `LD_LIBRARY_PATH=`*dirlist1*,*dirlist2*):

- Link Editor: *dirlist1*, `–L`, *dirlist2*, `/usr/ccs/lib`, `/usr/lib`
- Runtime Linker: *dirlist1*, *dirlist2*, `–R`, `/usr/lib`

Version Numbering

The SunOS release 4.x software supported both major and minor version numbers on shared libraries. The Solaris 2.6 operating environment supports only the major version number. For binary compatibility support, major and minor version numbers are recognized on SunOS release 4.x shared libraries. These libraries are required to retain the same major and minor version number they had in the SunOS release 4.x software.

Table 15–4 shows versions of SunOS release 4.x and Solaris 2.6 shared libraries.

TABLE 15–4 Example Shared Libraries

SunOS release 4.x	Solaris 2.6
`libc.so.1.7`	`libc.so.1`
`libdl.so.1.0`	`libdl.so.1`

In SunOS release 4.x system software, when the `–l` option was specified, the build environment linker searched for a library with both major and minor numbers. For example, if `–ldl` was specified, the library, `libdl.so.1.0` was linked. In the Solaris 2.6 environment, even though major numbers are still supported, the default behavior of the link editor is to ignore version numbers. Using the previous example, the build environment link editor now searches for `libdl.so` and a symbolic link points to a specific version file.

The recording of a *dependency* in a dynamic executable or shared object is, by default, the file name of the associated shared object as it is referenced by the link editor. To provide a more consistent means of specifying dependencies, shared objects can record within themselves the file name by which they

should be referenced at runtime. This is specified with the –h option when linking the library file.

Symbolic links have been created for most libraries in this release. You should build any new shared libraries with major numbers, then create a symbolic link to the version of the library that is used most often.

Examples

A new utility, dump(1), makes it easier to debug object files or to check the static and dynamic linking, see “Backing Up and Restoring Files” on page 96). The dump –L option displays the information needed by the runtime linker that is contained in the executable. This information is contained in the *dynamic section* of an ELF file. The RPATH entry displays search paths specified by the –R option to ld.

The following example:

- Builds a shared library, libx.so.1, using libx.o
- Creates a link from libx.so.1 to libx.so
- Shows dump output, including the SONAME field, which stores the information passed with the –h option.

```
examples% cc -G -o libx.so.1 -h libx.so.1 libx.o

examples% cp libx.so.1 /mylibs
examples% ln -s /mylibs/libx.so.1 /mylibs/libx.so
examples% dump -Lv libx.so.1

libx.so.1:

  **** DYNAMIC SECTION INFORMATION ****
.dynamic :
[INDEX] Tag      Value
[1]     INIT     0x3b8
[2]     FINI     0x3f4
[3]     SONAME   libx.so.1
[4]     HASH     0x94
[5]     STRTAB   0x33c
[6]     SYMTAB   0x14c
[7]     STRSZ    0x62
[8]     SYMENT   0x10
[9]     PLTGOT   0x10404
[10]    PLTSZ    0xc
[11]    PLTREL   0x7
[12]    JMPREL   0x3ac
[13]    RELA     0x3a0
```

(Continued)

(Continuation)

```
[14]    RELASZ   0x18
[15]    RELAENT  0xc
```

If a library needs other dynamic libraries, they should be specified along with an RPATH, as the next example shows.

The next example compiles `prog.c`, dynamically linking `libx.so` (as built in the previous example), and specifies that the binary retain the current directory information for execution. This example shows the output of `dump` from the compiled program, `prog.c`. Here, the information stored in the SONAME field of the previous example is shown as NEEDED by `prog`. When `prog` is run, it will use `libx.so.1` even if `libx.so` is linked to a different version.

```
examples% cc -o prog prog.c -L/mylibs -R/mylibs -lx
example% dump -Lv prog

prog:

  **** DYNAMIC SECTION INFORMATION ****
.dynamic :
[INDEX]   Tag   Value
[1]  NEEDED   libx.so.1
[2]  NEEDED   libc.so.1
[3]  INIT     0x1b1ac
[4]  FINI     0x1b248
[5]  RPATH    /mylibs
[6]  HASH     0x100e8
[7]  STRTAB   0x17f90
[8]  SYMTAB   0x12be0
[9]  STRSZ    0x31e1
[10] SYMENT   0x10
[11] DEBUG    0x0
[12] PLTGOT   0x2b25c
[13] PLTSZ    0x30
[14] PLTREL   0x7
[15] JMPREL   0x1b180
[16] RELA     0x1b174
[17] RELASZ   0x3c
[18] RELAENT  0xc
```

Debuggers

This section describes changes to debugging tools.

dbx and dbxtool

The dbx and dbxtool tools are no longer available with default system software. Enhanced versions of these tools are available as part of Sun WorkShop™, an unbundled product.

adb and kadb

The adb and kadb tools are available in the Solaris 2.6 operating environment. They offer the same capabilities as the SunOS release 4.x tools. kadb has been enhanced to recognize multiple processors. The processor ID is displayed in the kadb prompt. In the following examples, it is 0.

To make kernel debugging under the Solaris 2.6 operating environment easier:

- Enable savecore (uncomment the savecore lines in the /etc/init.d/sysetup file)
- Boot under kadb (type $c when the system crashes)
- Use adb and crash

kadb Macros

The kadb macros described below are particularly useful with the new multithreaded kernel.

thread displays the current thread. The current thread pointer is in SPARC global register g7.

```
kadb[0]: <g7$<thread
```

threadlist shows the stack traces of all the kernel threads in the system. This can be a *long* list.

```
kadb[0]: $<threadlist
```

`mutex` shows you the address of the owning thread. The following example uses the global unsafe driver `mutex`.

```
kadb[0]: unsafe_driver$<mutex
```

```
kadb[0]: moddebug/W 0x80000000
```

`moddebug`

`moddebug` enables you to watch module loading. See the end of `<sys/modctl.h>` for legal values for `moddebug` for debugging purposes only.

Debugging a Live Kernel

Use the following command to debug a live kernel.

```
# adb -k /dev/ksyms /dev/mem
```

`/dev/ksyms` is a pseudo device that contains the complete name list of the running kernel.

`truss` Command

`truss` is a new utility, provided to trace system calls performed, signals received, and machine faults incurred. `truss` offers several significant improvements over the SunOS release 4.x `trace`(1) command, including the ability to follow forked processes and to deal with multithreaded processes.

The following example shows a summary of traced calls for the `date` command. With the `–c` option, `truss` does not display the trace line by line. Instead, it counts the system calls, signals, and faults, and displays a summary.

```
example% truss -c date
Fri Sep 18 14:31:30 PDT 1992
syscall      seconds   calls  errors
_exit            .00       1
read             .00       7
write            .00       1
open             .03      12
close            .00      12
time             .00       1
brk              .01       4
lseek            .00       1
```

(Continued)

(Continuation)

```
fstat            .00       4
ioctl            .00       1
execve           .00       1
mmap             .01      17
munmap           .00       8
                ----     ---     ---
sys totals:      .05      70       0
usr time:        .03
elapsed:         .28
```

See the `truss`(1) man page for complete details on all `truss` options. There are a number of other Solaris 2.6 debugging tools based on `proc`(4) such as `pmap`(1).

CHAPTER 16

Tools and Resources

This chapter discusses the changes to tools and resources for the development environment.

`ioctl()` Requests

All `ioctl`s related to `dkio`(7I), `filio`, `mtio`(7I), `sockio`(7I), `streamio`(7I), `termio`(7I), and `termios`(7I) are supported in this release.

A few incompatibilities exist between the SunOS release 4.x `termios` structure and Solaris 2.6 `termios` structure. For example, the Solaris 2.6 `termios` structure does not include a `c_line` field.

The following `ioctl`s requests, defined in `<sys/ttold.h>`, are not implemented in this release.

- `TIOCMODG`
- `OTTYDISC`

- TABLDISC
- KBLDISC
- TIOCMIDS
- TIOCSETX
- NETLDISC
- NTABLDISC
- TIOCGETX
- NTTYDISC
- MOUSELDISC

The following `ttycom ioctl` requests are not in the Solaris 2.6 operating environment.

- TIOCSCTTY
- TIOCNOTTY
- TIOCISPACE
- TIOCPKT
- TIOCGETPGRP
- TIOCISIZE
- TIOCUCNTL
- TIOCOUTQ
- TIOCTCNTL
- TIOCCONS

Table 16–1 shows the `ioctl`s supported in the Solaris 2.6 operating environment.

TABLE 16–1 `ioctl()` Support

`ioctl()`	**Description**
`DKIOCGPART`	These requests are replaced with `DKIOCGAPART` and `DKIOCSAPART` in Solaris 2.6 software.
`DKIOCGCONF`	This request is replaced with `DKIOCINFO` in Solaris 2.6 software, which includes the combined information of the SunOS release 4.x `DKIOCGCONF` and `DKIOCINFO` structures.
`DKIOCSCMD`	This request succeeds only for IPI drives. This `ioctl` fails for SCSI devices. Use the USCSI `ioctl` for SCSI devices.

TABLE 16–1 `ioctl()` Support *(continued)*

`ioctl()`	**Description**
`DKIOCGLOG`	`EINVAL` is returned. `DKIOCWCHK` toggles the write check on the diskette drive.
`filio`	The following `filio ioctl` requests are not supported in this release or SVR4: `FIOSETOWN`, `FIOGETOWN`, `FIOCLEX`, `FIONCLEX`. `filio ioctl` requests are not defined in the ABI or SVID.
`mtio`	Not all devices support all `mtio ioctl` requests in Solaris 2.6. See the `mtio`(7) man pages.
`sockio`	The following `sockio ioctl` requests are implemented in SVR4 and Solaris 2.6 software: `SIOCSPGRP`, `SIOCGPGRP`, `SIOCATMARK`. `sockio ioctl` requests are not defined in the ABI or SVID.
`streamio`	All SunOS release 4.x `streamio ioctl` requests are implemented in Solaris 2.6 software, the ABI, SVID, and SVR4. The `I_FDINSERT` request requires an argument that points to a `strfdinsert` structure. The SunOS release 4.x `strfdinsert` structure includes an `fd (int)` field, while the ABI, SVID, or SVR4 `strfdinsert` structure includes a `fildes (int)` field instead.
`audioio`	The SunOS release 4.x `<sun/audioio.h>` file has been moved to `<sys/audioio.h>` for Solaris 2.6 software. Additionally, in Solaris 2.6 software, there are enhancements to the interface. See the `audio`(7), `audioamd`(7), or `dbri`(7) man pages for more information.
`termio`, `termios`	All SunOS release 4.x `termio` and `termios ioctl` requests are implemented in Solaris 2.6 software, the ABI, SVID, and SVR4. There are a few incompatibilities between the SunOS release 4.x `termios` structure and Solaris 2.6 software, or the ABI, SVID, or SVR4 `termios` structure. The SunOS release 4.x `termios` structure includes a `c_line` field that is not supported by the other releases. The `c_cflag` (hardware control of the terminal) can have CRTSCTS (enable RTS/CTS flow control) under the SunOS release 4.x software, but this value is not defined in the Solaris 2.6 software, the ABI, SVID, or SVR4. However, the functionality is supported through the `termiox`(7) interface.

ptrace() Request Values

The ptrace() facility is implemented on top of /proc. New applications should use proc(4) directly.

The ptrace() routine in Solaris 2.6 software is present solely to support applications running in BCP mode. It uses integers 1 – 9 as request values, while the SunOS release 4.x routine defines request values as symbolic constants in <sys/ptrace.h>. The following SunOS release 4.x *request* symbolic constants are compatible with Solaris 2.6 software.

- PTRACE_TRACEME
- PTRACE_PEEKTEXT
- PTRACE_PEEKDATA
- PTRACE_PEEKUSER
- PTRACE_POKETEXT
- PTRACE_POKEDATA
- PTRACE_POKEUSER
- PTRACE_CONT
- PTRACE_KILL
- PTRACE_SINGLESTEP

The SunOS release 4.x PTRACE_CONT *addr* argument specifies where the stopped process should resume execution, unless *addr* = 1, in which case execution resumes from where the process had stopped. The equivalent Solaris 2.6 request 7 requires that *addr* always be equal to 1 and that execution always resumes from where the process stopped. Also, the Solaris 2.6 request 7 cancels all pending signals before the process resumes execution except those specified by data. The SunOS release 4.x PTRACE_CONT does not cancel all pending signals.

Table 16–2 shows SunOS release 4.x valid requests that are not supported by the Solaris 2.6 ptrace() routine.

TABLE 16–2 `ptrace()` Requests not Supported by Solaris 2.6 Software

PTRACE_ATTACH	PTRACE_GETWINDOW
PTRACE_DETACH	PTRACE_SETWINDOW
PTRACE_GETREGS	PTRACE_22
PTRACE_SETREGS	PTRACE_23
PTRACE_GETFPREGS	PTRACE_26
PTRACE_SETFPREGS	PTRACE_27
PTRACE_READDATA	PTRACE_28
PTRACE_WRITEDATA	PTRACE_SYSCALL
PTRACE_READTEXT	PTRACE_DUMPCORE
PTRACE_WRITETEXT	PTRACE_SETWRBKPT
PTRACE_GETFPAREGS	PTRACE_SETACBKPT
PTRACE_SETFPAREGS	PTRACE_CLRDR7

Libraries

This release is compliant with the System V Interface Definition, Third Edition (SVID 3). Programs written with the SunOS release 4.1 System V libraries are easy to port to this release. Programs using the SunOS release 4.x BSD C library require more effort.

Reorganized Libraries

Several functions and groups of functions were moved into different libraries. This can cause references to these functions to be flagged as undefined when compiling a SunOS release 4.x application in the Solaris 2.6 environment.

After a compile, check the man page of any functions flagged as undefined. The synopsis list both the `–l` linker option and any include files that you need to resolve the symbol.

Shared Libraries

Shared libraries do not currently support minor version numbers.

Files for shared initialized data (`.sa`) are no longer required; no `.sa` files are provided with the Solaris 2.6 software.

Resource Limits

The Solaris 2.6 operating environment handles resource limits differently. In previous releases, static table allocations were used for resources such as file descriptors and active processes. These resources are now dynamically allocated, so they are limited by the physical memory available. Table 16–3 shows the resource limits.

TABLE 16–3 Resource Limits

Configuration	Limitation
`RLIMIT_CORE`	Maximum size of core file (in bytes) that can be created by a process
`RLIMIT_CPU`	Maximum amount of CPU time (in seconds) that a process can use
`RLIMIT_DATA`	Maximum size of a process's heap (in bytes)
`RLIMIT_FSIZE`	Maximum size of a file (in bytes) that can be created by a process
`RLIMIT_NOFILE`	One more than the maximum number of file descriptors that can be created by a process

TABLE 16–3 Resource Limits *(continued)*

Configuration	Limitation
RLIMIT_VMEM	Maximum size (in bytes) to which a process's mapped address space may grow
RLIMIT_STACK	Maximum size (in bytes) of a process's stack

Note - Any shared objects that need the networking libraries *must* be dynamically linked. The networking libraries require `libdl.so.1`. An archive library is not available.

Table 16–4 shows SunOS release 4.x and Solaris 2.6 libraries and their locations.

TABLE 16–4 Comparison of Library Locations

Library Name	SunOS release 4.x Directory	Solaris 2.6 Directory
libbsdmalloc.a	/usr/lib	/usr/lib
libc.a	/usr/lib and /usr/5lib	/usr/lib
libc.so.1.7	/usr/lib	/usr/lib
libc.so.2.7	/usr/5lib	/usr/lib
libc_p.a	/usr/5lib	Not available
libcurses.a	/usr/lib and /usr/5lib	/usr/ucblib and /usr/ccs/lib
libcurses_p.a	/usr/5lib	Not available
libdbm.a	/usr/lib	/usr/ucblib
libdl.so.1.0	/usr/lib	/usr/lib

TABLE 16–4 Comparison of Library Locations *(continued)*

Library Name	SunOS release 4.x Directory	Solaris 2.6 Directory
libg.a	/usr/lib	Not available
libkvm.a	/usr/lib	Not available
libkvm.so.0.3	/usr/lib	/usr/lib
libl.a	/usr/lib	/usr/ccs/lib
libln.a	/usr/lib	Not available
liblwp.a	/usr/lib	Not available
libm.a	/usr/lib	/usr/lib and /usr/lib/libp
libmp.a	/usr/lib	/usr/lib
libnbio.a	/usr/lib	Not available
libnsl.a	/usr/lib	/usr/lib
libpixrect.a	/usr/lib	Not available
libpixrect.so.2.14	/usr/lib	Not available
libposix.a	/usr/lib	Not available
libresolv.a	/usr/lib	/usr/lib
librpcsvc.a	/usr/lib	/usr/lib
libsuntool.so.0.54	/usr/lib	Not available
libsunwindow.so.0.55	/usr/lib	Not available
libsvidm.a	/usr/5lib	Not available

TABLE 16–4 Comparison of Library Locations *(continued)*

Library Name	SunOS release 4.x Directory	Solaris 2.6 Directory
libsvidm_p.a	/usr/5lib	Not available
libtermcap.a	/usr/lib and /usr/5lib	/usr/ucblib and /usr/ccs/lib
libtermlib.a	/usr/lib and /usr/5lib	/usr/ccs/lib
libxgl.so.1.1	/usr/lib	/opt/SUNWits/Graphics-sw/xgl/lib
libxpg.a	/usr/xpg2lib	Not available
liby.a	/usr/lib and /usr/5lib	/usr/ccs/lib

Using make

There are two make utilities available in the Solaris 2.6 operating environment. The default version, /usr/ccs/bin/make, is identical to the SunOS release 4.x make command. The SVR4 version is available in /usr/ccs/lib/svr4-make.

Using the default version, your Makefiles will not need changes. However, some of the commands used in your Makefiles may have changed. For example, install(1), commonly used in Makefiles, could produce unexpected results because of changes to the options, as shown in the following examples.

- In a SunOS release 4.x Makefile – install:

```
install –o bin –g bin –m 444 target.c /usr/bin/target
```

- In a SunOS release 5.6 Makefile – install:

```
install –u bin –g bin –m 444 target.c /usr/bin/target
```

The version of `install`(1B) in `/usr/ueb` is compatible with the SunOS release 4.x version.

Check the compatibility tables in Appendix A, for information about individual interfaces.

Using SCCS

The Solaris 2.6 operating environment source code control system (SCCS) is slightly different from the SunOS release 4.x version. The same set of commands and subcommands are supported in both environments. SCCS directories and `s.files` used on SunOS release 4.x systems work equally well on Solaris 2.6 systems.

In the SunOS release 4.x software, the SCCS commands were located in the `/usr/sccs` directory. These commands are located with the other programming tools in `/usr/ccs/bin` in the Solaris 2.6 operating environment.

One difference between SunOS release 4.x and Solaris 2.6 utilities is the handling of unreadable `s.files`. The SunOS release 4.x commands print an error and continue when they encounter an unreadable `s.file`. The Solaris 2.6 commands silently ignore the error.

Determining Application Compatibility

Although the Binary Compatibility Package is not provided as a development environment, it requires sound programming practices that can improve binary compatibility with future releases.

The Binary Compatibility Package provides compatibility for dynamically linked and statically linked applications, as well as hybrids that are partially static and partially dynamically linked.

The Binary Compatibility Package works with well-behaved user applications. Well-behaved applications do not:

- Trap directly to the kernel
- Write directly to any system files

- Use `/dev/kmem`, `/dev/mem`, or `libkvm`
- Use unpublished SunOS interfaces
- Rely on customer-supplied drivers

Applications that are not well-behaved can produce unpredictable results.

Information on using the Binary Compatibility Package is available in *Binary Compatibility Guide.*

Packaging Applications

The Solaris 2.6 operating environment is bundled in units called *packages.* These packages contain all the files and information you need to add or remove software from your system.

A package consists of the following components:

- `pkginfo` file – This is an ASCII file that sets characteristics of the package. It consists of a list of *macro=value* pairs that describe the package and set control parameters for its installation. See the `pkginfo`(4) man page for more information.
- `prototype` file – This is an ASCII file that defines the contents of the package. It contains one entry for each deliverable object (for example, files, directories, and links). It also contains installation entries for package *information* files–such as `pkginfo`, `depend`, and `copyright`–and scripts. See the `prototype`(4) man page for more information.
- `copyright` file –This is an ASCII file that provides a copyright notice for the package. Its contents (including comment lines) are displayed during package installation. See the `copyright`(4) man page for more information.
- Package contents – The contents of the package.
- Scripts – Scripts can be used to control installation or removal of a package, to request input from the user, or to perform an action on all objects of a particular class. Scripts must be executable by the Bourne shell.

Add-on application software should be packaged so it can be installed on a Solaris 2.6 system from diskette, tape, or CD-ROM. *Application Packaging Developer's Guide* provides guidelines for building your packages.

Packaging Utilities

Several utilities are provided to create and manipulate packages. Table 16–5 lists commands that are useful for creating packages.

TABLE 16–5 Commands for Creating Packages

`pkgproto`	Generates prototype file entries for input to the `pkgmk` command
`pkgmk`	Produces an installable package
`pkgtrans`	Translates package format

Table 16–6 lists commands that are useful for adding and removing packages.

TABLE 16–6 Commands for Adding and Removing Packages

`pkgadd`	Adds software package to the system
`pkgask`	Stores answers to a request script
`pkgrm`	Removes a package from the system
`pkgchk`	Checks accuracy of installation

Table 16–7 lists commands that provide information about packages.

TABLE 16–7 Commands for Providing Information About Packages

`pkginfo`	Displays software package information about installed packages
`pkgparam`	Displays package parameter values

Toolkits

This section discusses OPEN LOOK Intrinsics ToolKit (OLIT) and XView.

OLIT

The OPEN LOOK Intrinsics Toolkit (OLIT) is based on Xt Intrinsics. It provides a set of functions common to many widget sets to create, employ, and destroy user interface components for an X environment.

XView

The XView Window Toolkit provides an implementation of the OPEN LOOK Graphical User Interface (GUI) specification. It provides a migration path for SunView applications.

XView uses variable-length attribute-value lists based on `varargs` to specify objects to be created, such as windows, menus, and scrollbars. This eliminates most of the boilerplate software usually found in procedural interfaces, since the usual behavior is already defined.

Finding SunOS Release 4.x Tools

Most SunOS release 4.x programming tools are still available and stil provide the same capabilities, but many are in new locations. All bundled programming tools are now in the `/usr/ccs/bin` library except `cpp`, which is now in the `/usr/ccs/lib` library. Table 16–8 shows the programming tools and their SunOS release 4.x locations.

TABLE 16–8 Bundled Programming Tools

SunOS release 4.x Command	SunOS release 4.x Location
admin	/usr/sccs
ar	/usr/bin
as	/usr/bin
cdc	/usr/sccs
comb	/usr/sccs
cpp	/usr/lib/cpp
delta	/usr/sccs
error	/usr/ucb
get	/usr/sccs
help	/usr/sccs
ld	/usr/bin
lex	/usr/bin
lorder	/usr/bin
m4	/usr/bin
make	/usr/bin
nm	/usr/bin
prof	/usr/bin
prs	/usr/sccs
prt	/usr/sccs

TABLE 16–8 Bundled Programming Tools *(continued)*

SunOS release 4.x Command	SunOS release 4.x Location
ranlib	/usr/bin
rmdel	/usr/sccs
sact	/usr/sccs
sccs	/usr/ucb
sccsdiff	/usr/sccs
size	/usr/bin
strip	/usr/bin
symorder	/usr/ucb
tsort	/usr/bin
unget	/usr/sccs
unifdef	/usr/ucb
val	/usr/sccs
vc	/usr/old
what	/usr/sccs
yacc	/usr/bin
yaccpar	/usr/lib

Table 16–9 lists the new Solaris 2.6 programming tools and their descriptions.

TABLE 16–9 New Programming Tools

New Command	Description
`dis`	Object code disassembler
`dump`	Dumps selected parts of an object file
`exstr`	Extracts strings from source files
`mcs`	Manipulates the comment section of an object file
`regcmp`	Regular expression compiler
`truss`	Traces system calls and signals
`ptools`	Miscellaneous `/proc` utilities

Table 16–10 lists the SunOS release 4.x commands that are now unbundled.

TABLE 16–10 Unbundled Programming Tools

Unbundled Command	Description
`cb`	Simple C program beautifier
`cc`	C compiler
`cflow`	Generates a flow graph for a C program
`cscope`	Interactively examines a C program
`ctrace`	Generates a C program execution trace
`cxref`	Generates a C program cross-reference
`dbx`	Source-level debugger
`dbxtool`	Window-based source-level debugger
`gprof`	Displays call-graph profile data

TABLE 16–10 Unbundled Programming Tools *(continued)*

Unbundled Command	Description
`indent`	Indents and formats C program source files
`inline`	In-line procedure call expander
`lint`	C program verifier
`objdump`	Dumps selected parts of a COFF object file
`tcov`	Constructs test coverage analysis and statement-by-statement profile

CHAPTER 17

Networking and Internationalization

This chapter discusses Solaris 2.6 networking features as they relate to the programming environment, and it discusses issues concerning the improved internationalization features.

Networking

The Solaris 2.6 operating environment includes the following networking features:

- Distributed file system (DFS), which centralizes the file system utilities
- Network information services plus (NIS+) including NFS
- Name service switch file

See *NIS+ Transition Guide* and *NFS Administration Guide* for more information on using these services.

NIS, NIS+

The Solaris 2.6 operating environment supports the network information service (NIS), the SunOS 4.x name service, and the network information services plus (NIS+), an enterprise-naming service of heterogenous distributed systems. See "NIS+ " on page 124 for the nature of NIS+ support available in the Solaris 2.6 operating environment.

NIS+ provides a more detailed model for objects in the name space, improved security, and faster updates than NIS.

The NIS+ programmer interfaces are documented in section 3N of the man Pages(3): Library Routines.

`nsswitch.conf` File

The `nsswitch.conf` file simplifies name service administration. Applications can use this file to select a name service. This information no longer needs to be hard-coded into the service itself. See the `nsswitch.conf`(4) man page for more information on the format of this file.

Network Interface Tap

The Network Interface Tap (NIT) provided in the SunOS 4.x release is no longer required. Now Ethernet drivers are real STREAMS drivers that can be opened and communicated with directly.

See `pfmod`(7M), `bufmod`(7M), and `dlpi`(7P)

The Solaris 2.6 Ethernet drivers and other data link drivers support the connectionless Data Link Provider Interface (DLPI) Version 2 specification.

Sockets

Sockets are supported in the Solaris 2.6 operating environment. Unlike the SunOS release 4.x software, sockets are no longer implemented completely in the kernel. They are now in a library, `libsocket`, implemented on STREAMS.

Internationalization

Most of the changes in the Solaris 2.6 operating environment improve on previous internationalization features. For complete information on internationalization support, see *Developer's Guide to Internationalization*.

Application developers concerned with internationalizing their programs should follow these guidelines:

- Call `setlocale`(3C) to set up the `LANG` environment variable
- Use standard code sets and follow 8-bit boundaries
- Use `strftime`(3C) to print the date and time
- Replace `strcmp`(3) with `strcoll`(3C) for user-visible collation
- Call `gettext`(3C) or `catget`(3C) to retrieve translated strings from locale-specific message catalogs

Character Support

The Solaris 2.6 operating environment supports extended UNIX code (EUC), VTF8, PCK, and B165. This allows multibyte and multiple code sets on one system.

The SunOS release 4.x software supported single-byte representation of non-ASCII characters. The Solaris 2.6 operating environment supports multibyte representation. This support is needed for Asian language character sets, which contain thousands of characters.

The multibyte functions are included in libc and provides the following features:

- Multibyte-to-wide character conversions
- Wide character standard I/O
- Wide character classification
- Wide character formatting

The Solaris 2.6 operating environment supports multibyte file names; however, login and machine names should be restricted to ASCII characters.

Message Catalogs

SunOS release 4.x support for message catalogs is enhanced in the Solaris 2.6 operating environment to enable the creation of message catalogs using multibyte characters.

Using message catalogs, an application can display messages at run-time in the native language in which an application was run. These message catalogs must first be created for the native language specified by the language locale.

Locale Database

The SunOS release 5.6 locale database (`/usr/lib/locale/`*locale*) is completely different than the locale database of SunOS 5.x. This is transparent to the user, however.

Commands

Most of the system commands in the Solaris 2.6 operating environment have been messaged. Many of these commands can pass through multibyte character representations. The increased number of messaged commands makes localization efforts easier.

The `installtxt`(1) command has been replaced with `msgfmt`(1). Use the new `xgettext`(1) command to extract messages.

Changes to `strftime`(3C) affect date and time formats. Shell programs that rely on the output format of the `date`(1) command will have to be updated to handle the new format.

`chrtbl`(8) and `catdef`(8) are replaced by `localedef`(1).

Libraries

The `/usr/xpg2lib/libxpg2.a` archive library is no longer available. These routines have been included in `libc`.

Table 17–1 shows the new location of these interfaces.

TABLE 17–1 `xpg2lib` Library Routine Locations

Routine	Solaris 2.6 Location
`bindtextdomain`	`/usr/lib/libc`
`chroot`	`/usr/lib/libc`
`catgets`	`/usr/lib/libc`
`dgettext`	`/usr/lib/libc`
`getcwd`	`/usr/lib/libc`

TABLE 17–1 `xpg2lib` Library Routine Locations *(continued)*

Routine	Solaris 2.6 Location
`getut`	`/usr/lib/libc`
`l3tol`	Not supported.
`logname`	`/usr/lib/libc`
`malloc`	`/usr/lib/libc`
`swab`	`/usr/lib/libc`
`langinfo`	`/usr/lib/libc`
`gettext`	`/usr/lib/libc`
`sbrk`	`/usr/lib/libc`
`textdomain`	`/usr/lib/libc`

Programs that use these routines no longer need to pass `–lxpg2` to the C compiler although some may need to include `libintl.h`. (See Table 17–1 for these routines.)

The `catgetmsg`(3C) routine is no longer available.

The order of locale categories in the string returned by `setlocale`(3C) differs between the SunOS release 4.x and the Solaris 2.6 software. This string is normally used by a subsequent call to `setlocale`(3C), and the order should not matter. Applications should not rely on a specific order of locale categories.

CHAPTER 18

System and Device Configuration

The operating system kernel and its interfaces have changed significantly. Binary compatibility is not provided for SunOS release 4.x device drivers. This chapter discusses changes in the Solaris 2.6 operating environment that affect kernel and system developers.

System Configuration

Changes related to system configuration include the dynamically loaded kernel and kernel layout, the `config` and `boot` commands, and the `/etc/system` file.

Dynamically Loaded Kernel

Unlike previous SunOS releases, the kernel is now dynamically configured. The kernel now consists of a small static core and many dynamically loadable kernel modules. Drivers, file systems, STREAMS modules, and other modules are loaded automatically as needed, either at boot time or at runtime. When these modules are no longer in use, they may be unloaded. Modules are kept in memory until that memory is needed. `modinfo`(1M) provides information about the modules currently loaded on a system.

The `modload`(1M) and `modunload`(1M) commands are still available in this release but they perform differently. They have more limited usage and are

no longer sufficient to correctly install a loadable driver onto the system. `modunload` now includes the capability to unload all unloadable (and not busy) modules. Use `modunload` as follows.

```
# modunload -i 0
```

Kernel Layout

The contents of the kernel, which were formerly in a single file, `/vmunix`, are now contained in modules in a directory hierarchy. By default, the directory hierarchy is`/platform/`*'uname -i'*`/kernel`, `/kernel`, and `/usr/kernel`.

The directory search path for modules can be set by the `moddir` variable in the `/etc/system` file (see the system(4) man page). Typically, `/platform/`*'uname -i'*`/kernel/unix` is the first portion of the kernel to be loaded (see the `kernel`(1M) man page).

`config` Command

In the SunOS release 4.x software, the `config` command was used to generate system configuration files that enabled `/vmunix` to be relinked from object files. The need for this command has been removed by the following Solaris 2.6 features:

- Loadable modules
- The `/etc/system` file (see the system(4) man page)
- Device tree information from the OpenBoot PROM (OBP)
- The `driver.conf` files in `/kernel/drv` and `/usr/kernel/drv`

`/etc/system` File

System configuration information is now set in the `/etc/system` file. This file also modifies the kernel's treatment of loadable modules. The file contains commands of the form:

`set` *parameter=value*

For example, in the SunOS release 4.x software, `MAXUSERS` was set using `config`(8). In the Solaris 2.6 operating environment, it is set in the `/etc/system` file with the following line:

```
set maxusers = number
```

Commands that affect loadable modules are of the form:

`set` *module*:*variable*=*value*

Changes made to the `/etc/system` file only take effect when you reboot your system (see the `system`(4) man pages).

`boot` Command

In this release, the following boot programs are available:

- `ufsboot` – To boot from a disk or a CD
- `inetboot` – To boot from across the network

  When booting from a disk, the PROM assumes that the primary boot block resides in blocks 1 – 15 of the local disk. Use `installboot`(1M) to create the boot block:

```
# installboot /usr/platform/'uname -i'/lib/fs/ufs/bootblk \
/dev/rdsk/c0t3d0s0
```

The system firmware loads the primary bootstrap (the boot block) program into memory and runs it. The boot block is a UFS file system reader. It loads the secondary boot program (`/platform/`*'uname -i'*`/ufsboot`) into memory.

`ufsboot` loads `kernel/unix`, then `/kernel/unix` uses `ufsboot` to load modules from the `kernel` directory hierarchy until it is able to mount the root file system.

During these operations, the boot block and `ufsboot` use the drivers provided by the firmware; neither `ufsboot` nor the boot block contains any driver code. The `ufsboot` code does not have to change to incorporate a new SBus card with a new disk type since `ufsboot` uses the SBus card PROM driver.

When booting over the network, the boot program performs as it did for a diskless boot in the SunOS release 4.x software. However, the boot program is now called `inetboot` and the client `vfstab` file entries are different. See *System Administration Guide* for information on diskless booting.

Summary of Boot Differences

Table 18–1 summarizes the differences in the boot sequence between the SunOS release 4.x and the Solaris 2.6 operating environment.

TABLE 18–1 Summary of Boot Differences

SunOS release 4.x	Solaris 2.6	Description
boot block	`bootblk`	Loads `ufsboot` from disk
boot program	`ufsboot`	Loads `unix` from disk
`vmunix`	`unix`	Bootable kernel image
boot.sun4c.sunos.4.1.1	`inetboot`	Mounts and copies `unix` from network
`rc.boot`, `rc.single`	`/etc/rcS`	Mounts `/usr` and checks file systems
`rc.local`	`/etc/rc2`, `/etc/rc3`, `/etc/rc2.d`, `/etc/rc3.d`	System configuration scripts
`config`	`modload`, `/etc/system`, `add_drv`, `rem_drv`	Customizes system kernel; loads, adds, and removes modules as needed
PROM monitor, single user, multiuser	Run states 0 – 6, and S	System run levels

Reconfiguration Boot

A reconfiguration boot tells the system to probe for all connected devices and build the names for them in `/devices` and `/dev`. A reconfiguration boot, performed when adding new hardware to the system, is triggered by booting with the `–r` option:

```
ok> boot –r
```

If another device of an existing type (with the driver already installed) is added, and you forget to do a reconfiguration boot, you can use the following commands to tell the system to recognize the new device.

```
# touch /reconfigure
# _INIT_RECONFIG=YES /etc/init.d/drvconfig
# _INIT_RECONFIG=YES /etc/init.d/devlinks
```

Device Naming From a Developer's Perspective

This section expands on the discussion in "Device Naming Conventions" on page 57, focusing on aspects of device naming that concern system and kernel developers.

/devices

The `/devices` tree represents the tree of devices recognized by the kernel. This tree is configured by the `drvconfig`(1M) program. `drvconfig` is normally run only when the system is booted with the `–r` flag (see "Reconfiguration Boot" on page 186). `drvconfig` configures `/devices` with information about devices (with drivers) that are connected and ready at boot time.

Entries are exported by device drivers calling `ddi_create_minor_node`(9F) when they have determined that a device exists.

Use the `add_drv`(1M) command to add a device to the system. If the driver was successfully added, `add_drv` will also run `drvconfig`.

/dev

In this release, `/dev` is managed by utility programs that create symbolic links to the real entries in `/devices`. The programs are:

- `disks`(1M)
- `tapes`(1M)
- `ports`(1M)
- `devlinks`(1M)

  You can run a script to create the appropriate links from `/dev` to `/devices`. The `/dev` names have the advantage of being simpler and

more familiar, while the /devices names are unique names for the hardware.

Device Driver Naming

Each device in the system is driven by a device driver. Device drivers manage many instances of a device. Devices are named in several ways:

- Physical names
- Logical names
- Instance names

Physical Names

Physical names are stored in /devices. They describe the hardware, and vary with the platform and configuration. For example:

```
/devices/vme/xdc@6d,ee80/xd@0,0:g
```

Physical names can be used to identify which piece of hardware is in use. For example, xdc@6d,ee80 refers to the disk controller at address 0xee80 in VME A16, D32 space. See the vme(4) and driver.conf(4) man pages.

Logical Names

Logical names are stored in /dev. They attempt to abstract most of the nature of physical device names that are specific to the platform. Logical names might be appropriate for an xd device, such as:

/dev/dsk/c2d0s6 (controller 2, slave 0, slice 6 (4.x partition "g"))

or an sd device, such as:

/dev/dsk/c0t3d0s0 (controller 0, target 3, lun 0, slice 0 (4.x partition "a"))

The logical name conveys nothing about the type of controller. It does not differentiate between SCSI and IPI; they are both just disks.

Disk Names

Disk names use the SVR4 convention of *slice* numbers 0-7 instead of the letters a-h used in the SunOS release 4.x software.

Disk names also use the SVR4 convention of /dev/dsk/* for block disk devices and /dev/rdsk/* for raw disks. For more information, see *System Administration Guide*.

Instance Names

Instance names refer to the *nth* device in the system (for example, `sd20`).

Instance names are occasionally reported in driver error messages. You can determine the binding of an instance name to a physical name by looking at `dmesg`(1M) output, as in the following example.

```
sd9 at esp2: target 1 lun 1
sd9 is /sbus@1,f8000000/esp@0,800000/sd@1,0
     <SUN0424 cyl 1151 alt 2 hd 9 sec 80>
```

Once the instance name has been assigned to a device, it remains bound to that device.

Instance numbers are encoded in a device's minor number. To keep instance numbers persistent across reboots, the system records them in the `/etc/path_to_inst` file. This file is read only at boot time, and is currently updated by the `add_drv`(1M) and `drvconfig`(1M) commands. See the `path_to_inst`(4) man page for more information.

CHAPTER 19

Device Drivers and STREAMS

This chapter discusses device driver issues such as changes to device driver interfaces, the `devinfo` command, porting considerations, STREAMS, and Solaris 2.6 driver architecture.

See the following guides for more information on the topics discussed in this chapter:

- *STREAMS Programming Guide*
- *System Interface Guide*
- *System Administration Guide*

Device Drivers and STREAMS Device Drivers

Some of the many changes to device drivers in the Solaris 2.6 operating environment include the new DDI/DKI routines, Solaris SPARC DDI-specific routines, new software properties, and loadable drivers. In addition, many previous device issues have become opaque to the driver including interrupts, DVMA, and memory mapping.

Device Driver Interfaces

In previous SunOS releases, a driver writer had to cope with changes in the device driver interfaces. Usually, there was a porting effort with each release of the operating system. In addition, the interfaces for each platform varied, so device drivers often required separate releases for each platform. Third-party device driver releases often included complex scripts that would reconfigure and rebuild the operating system in order to integrate a device driver. It was costly to support and maintain device drivers.

Unlike previous releases of SunOS systems (SunOS release 4.1.3 software and earlier), the device driver interfaces in the Solaris 2.6 operating environment are formalized and are referred to as the *Solaris 2.6 SPARC DDI/DKI*. The Solaris 2.6 SPARC DDI/DKI provides binary compatibility of device drivers across all supported platforms and for all future releases of the Solaris 2.6 operating environment on those platforms.

The term *DDI/DKI* is derived from the original specification as supplied in the SVR4 release. It stands for *device driver interface/driver kernel interface*. The interfaces are divided into three groups:

- DDI/DKI
- DKI only
- DDI only

DDI/DKI

The *DDI/DKI interfaces* were standardized in SVR4, and are generic across all implementations of SVR4, regardless of the platform on which they are running.

DKI

The *DKI-only interfaces* are generic like the DDI/DKI interfaces and are supported in all SVR4 implementations. However, they are not guaranteed to be supported in future releases of System V.

DDI

The *DDI-only interfaces* are intended to be architecture-specific; for example, methods to access and control-device and system-specific hardware (that is, I/O registers, DMA services, interrupts, and memory mapping). These interfaces are not guaranteed to work in other SVR4 implementations.

This group of features effectively lowers the cost of driver support and maintenance. These features, combined with the large number of SPARC platforms, are helpful to many new third-party hardware developers.

With this level of binary compatibility, third-party hardware developers can now “shrink-wrap” their DDI-compliant device drivers with their driver hardware. Installing a new driver package can now be entirely automated. The self-configuring kernel removes the necessity for recompiling the kernel to add or remove a driver. Thus, DDI-compliant device driver for Solaris 2.6 environments can be treated like any other consumer software product.

In the Solaris 2.6 DDI/DKI the DDI-only interfaces are generic to all systems that support the Solaris 2.6 DDI/DKI. Note that the interfaces that make up the Sun common SCSI architecture (SCSA), and the locking interfaces used to make the driver behave correctly in a multithreaded kernel are also considered DDI-only interfaces in the Solaris 2.6 operating environment.

SCSA shields device drivers from details specific to the platform relating to host adapter implementations. With SCSA, a SCSI driver can run on all supported platforms.

A device driver that restricts itself to using only interfaces in the previous categories above is said to be *Solaris 2.6 DDI/DKI compliant*. A Solaris 2.6 DDI/DKI compliant device driver is commonly referred to as a *DDI-compliant* device driver.

Documentation

The man pages for the driver routines, structures, and support routines that comprise the DDI/DKI can be found in the sections of man Pages(1M): System Administration Commands listed below. See the Intro(9) man page for more information about these sections.

- Section 9E – Driver entry points
- Section 9F – Driver support functions
- Section 9S – Kernel structures

A Device Driver Developers Kit (DDK) is available separately.

`devinfo` Command

The Solaris 2.6 `devinfo` command performs a different function from the SunOS release 4.x version. The new `prtconf`(1M) command provides the information that the SunOS release 4.x `devinfo` command formerly displayed. The following examples show the output of each command.

```
4.1system% devinfo
Node 'SUNW,Sun 4/50', unit #0 (no driver)
        Node 'packages', unit #0 (no driver)
        Node 'openprom', unit #0 (no driver)
        Node 'zs', unit #0
        Node 'zs', unit #1
        Node 'audio', unit #0
        Node 'eeprom', unit #0 (no driver)
        Node 'counter-timer', unit #0 (no driver)
        Node 'memory-error', unit #0 (no driver)
        Node 'interrupt-enable', unit #0 (no driver)
        Node 'auxiliary-io', unit #0 (no driver)
        Node 'sbus', unit #0
                Node 'dma', unit #0
                Node 'esp', unit #0
                        Node 'sr', unit #0
                        Node 'sd', unit #0
                Node 'le', unit #0
                Node 'cgsix', unit #0
        Node 'memory', unit #0 (no driver)
        Node 'virtual-memory', unit #0 (no driver)
        Node 'fd', unit #0
        Node 'options', unit #0 (no driver)
```

```
5.3system% prtconf

 System Configuration:  Sun Microsystems  sun4c
Memory size: 32 Megabytes
System Peripherals (Software Nodes):

SUNW,Sun 4_75
    packages (driver not attached)
        disk-label (driver not attached)
        deblocker (driver not attached)
        obp-tftp (driver not attached)
    openprom (driver not attached)
    zs, instance #0
    zs, instance #1
    audio (driver not attached)
    eeprom (driver not attached)
    counter-timer (driver not attached)
    memory-error (driver not attached)
    interrupt-enable (driver not attached)
    auxiliary-io (driver not attached)
    sbus, instance #0
        dma, instance #0
        esp, instance #0
            sd (driver not attached)
            st (driver not attached)
            sd, instance #0
            sd, instance #1 (driver not attached)
            sd, instance #2 (driver not attached)
```

(Continued)

(Continuation)

```
            sd, instance #3
            sd, instance #4 (driver not attached)
            sd, instance #5 (driver not attached)
            sd, instance #6
        le, instance #0
        cgsix, instance #0
    memory (driver not attached)
    virtual-memory (driver not attached)
    fd (driver not attached)
    options, instance #0
    pseudo, instance #0
```

Porting Considerations

With the self-configuring kernel, Solaris 2.6 drivers will look more like SBus drivers than other types. All drivers are loadable, and no kernel configuration is required.

Under the SunOS release 4.x software, only one processor could be in the kernel at any one time. This was accomplished by using a *master lock* around the entire kernel. When a processor wanted to execute kernel code, it would acquire the lock (excluding other processors from running the code protected by the lock) and it would release the lock when it finished.

The Solaris 2.6 kernel is *multithreaded*. Instead of one master lock, there are many smaller locks that protect smaller regions of code. For example, there may be a kernel lock that protects access to a particular vnode, and one that protects an inode. Only one processor can be running code dealing with that vnode at a time, but another could be accessing an inode. This allows a greater amount of concurrency.

The multithreaded kernel will have a major impact on how you design the driver. The old model of using splN/splr pairs no longer works (on a uniprocessor or a multiprocessor system[1]). Instead, you have a choice of MT-style locks. The most common of these for drivers will be mutual exclusion locks, *mutexes*, and condition variables (which are an approximate equivalent of `sleep`()/`wakeup`() synchronization).

The old notion that you *owned* the processor until you explicitly called `sleep`() is no longer true. Because of kernel pre-emption, the CPU is switched from thread to thread so you *must* use the appropriate MT lock primitives to guard against concurrent access to device registers, shared data structures, and the like.

1. Strictly speaking, the splN/splr pair do work; however, although they will block interrupts, the effect is useless in protecting data structures in a multiprocessor environment.

A large percentage of the driver code for simple device drivers, which consist primarily of calls to kernel interface routines, will change, but in straightforward ways. For complex device drivers (such as a SCSI driver) which contain large amounts of device-specific handling code, only a small percentage of the driver—the driver interfaces—changes. This driver interface can be a kernel-to-driver interface, a driver-to-kernel interface, or a driver-to-driver interface.

Before you determine how you will support a driver in the Solaris 2.6 operating environment, refamiliarize yourself with how the driver works. Determine what the SunOS release 4.x driver *did* (not the specific implementation, but general behavior). What interfaces did it export? What `ioctl`()s did it provide? How did the hardware work and what peculiarities of the hardware did the driver support? Did the driver support multiple `open`() calls?

The following changes affect drivers and should be considered:

- The entry points to drivers are very different
- ANSI C requirements:
  - `volatile` keyword
  - `const` keyword
  - Function prototype declarations
- Relocated or renamed header files (most, if not all, system header files are now in `/usr/include/sys`)
- Most structures have become opaque or are no longer needed. For example:
  - `struct user`
  - `struct proc`
  - `struct dev_info`

STREAMS

Some areas of change for STREAMS modules are transparent I/O controls, automatic pushing of modules on a stream, and new message types.

Transparent `ioctl`()s

In the SunOS release 4.x software, you had to know that a particular driver was a STREAMS driver before making `ioctl`() requests.

For non-STREAMS drivers, you could do a direct `ioctl`() request:

```
ioctl(fd, DRIVER_IOCTL, arg);
```

For a STREAMS driver, you had to set up a `strioctl` structure and then use:

```
ioctl(fd, I_STR, &strioctl);
```

There was no easy way to determine whether a driver was STREAMS-based. Now, unrecognized `ioctl`s to the stream head are passed on to the driver, eliminating the need to know whether a driver was STREAMS-based.

Message types added in the Solaris 2.6 software support transparent `ioctl`s. There are now “copy in” and “copy out” messages to inform the STREAM head to transfer user data to and from the kernel.

For more information on writing STREAMS drivers, see the *STREAMS Programming Guide.*

`autopush` Command

The SunOS release 4.x `streamtab` structure enabled a driver to specify that certain STREAMS modules be pushed when the device was `open`().

In the Solaris 2.6 operating environment, the system administrator and the `autopush`(1M) command specify when a STREAMS module is pushed. If required, `autopush` can be run at driver installation.

See *STREAMS Programming Guide* for more information about pushing STREAMS modules.

Solaris 2.x Driver Architecture

To achieve binary compatibility across all currently supported hardware platforms, the DDI interfaces were carefully designed around architectural abstractions. The underlying abstraction, the `device` tree, is an extension of the `devinfo` tree in the original SPARCstation™ design. Each node in the `device` tree is described by a device information structure or “`dev_info` node.” The bottom-most nodes in the tree are termed *leaf nodes*. Most devices (such as disks and tape drives, framebuffers, I/O cards, and network interfaces) are examples of leaf devices that would be associated with leaf nodes. The associated device drivers are called *leaf drivers.*

The intermediate nodes in the tree are generally associated with buses (for example, SBus, SCSI, VME). These nodes are called *nexus nodes* and the drivers associated with them are called *nexus drivers.* Bus nexi are intended to encapsulate the architectural details associated with a particular element.

Currently, the Solaris 2.6 DDI/DKI supports only the writing of *leaf drivers* and one type of *nexus driver*, the SCSI host bus adapter driver.

The device tree structure creates a formal parent-child relationship between nodes. This parent-child relationship is the key to platform architecture independence.

When a leaf driver requires a service that is platform dependent (for example, a DMA mapping), the system transparently converts the request into a call to its parent to provide the service. The service providers are always nexus drivers; each nexus driver can in turn pass the request to its parent in order to provide the service. This approach enables leaf drivers to operate regardless of the platform architecture.

Device Driver Commands

The device driver commands are `add_drv`, `rem_drv`, `modload`, and `modunload`.

- `add_drv`(1M) – Informs the system that there is a newly installed device driver.
- `rem_drv`(1M) – Informs the system that the specified driver module is no longer valid.
- `modload`(1M) – Loads the specified loadable module into the running system.
- `modunload`(1M) – Unloads the specified loadable module from the running system.

APPENDIX **A**

Commands Reference Table

This appendix contains a user and system administration commands reference table that lists all SunOS release 4.x command interfaces and shows their status in the Solaris 2.6 environment and the SunOS/BSD Source Compatibility Package.

Using the Reference Table

- If an interface is listed as changed (C), a brief description of differences between SunOS release 4.x command and the Solaris 2.6 command is provided.
- If an interface is listed as the same (S), the Solaris 2.6 interface supports all features of the SunOS release 4.x interface. In some cases the interface has been enhanced, but can be considered a complete superset of the SunOS release 4.x interface.
- If an interface has an alternative (A), check the Notes section for its replacement.
- If an interface is listed as not available (N), check the Notes section for information about its replacement. Replacement commands, when available, are also shown in the SunOS release 5.6 column.

Note - The SunOS release 5.6 directory structure is different than the SunOS release 4.x structure; some commands behave the same, but have a different path name. For example, the SunOS release 4.x `/usr/etc/newfs` command now resides in `/usr/sbin/newfs`, but the interface has not changed. This command, and others like it, are considered the same (S) according to this table's guidelines.

Commands that exist in both `/usr/bin` and `/usr/5bin` have two table entries, the first documents the `/usr/bin` command, and the second entry documents the `/usr/5bin` command.

For complete information on all Solaris 2.6 interfaces, see man Pages(1): User Commands.

Examples

Table A-1 through Table A–4 show sample table entries and are followed by an interpretation

TABLE A–1 Table Entry Example 1

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`fasthalt`(8)	A	The `init 0` command provides similar capabilities	S

The `fasthalt` command is not available in the Solaris 2.6 base release. This command is available if you install the SunOS/BSD Compatibility package on your system. The `init 0` command replaces `fasthalt`. If you use the compatibility package `fasthalt` command in scripts or applications, they will not work on other SVR4 systems. Compatibility package commands can be found in `/usr/ucb` on systems that have this package installed, and they are documented in section 1B of man Pages(1): User Commands; for example, `fasthalt`(1B).

TABLE A–2 Table Entry Example 2

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`cc`(1V)	N	The C compiler is only available with the C language unbundled tools.	C

The C compiler is not available in the SunOS release 5.6 software. A C compiler is available with the SunOS/BSD Compatibility package, but it requires the unbundled C compiler and does not provide the same interface and output as the SunOS release 4.x compiler. See “Compiler Option Differences” on page 140 for details.

TABLE A–3 Table Entry Example 3

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`date`(1V) – SysV	S		N
`date`(1V)	C	The format used when setting the date is slightly different in SunOS release 5.6. See the date(1) man page for more information.	N

The SunOS release 4.x software had two date commands: `/usr/5bin/date` (compared in the SysV entry) and `/usr/bin/date` (compared in the second entry). The `/usr/5bin/date` command is identical to the SunOS release 5.6 command. If you had `/usr/5bin` in your path before `/usr/bin`, you will not notice any difference in this command in the SunOS release 5.6 software. If you are accustomed to using the SunOS release 4.x `/usr/bin/date` command, you should look at the SunOS release 5.6 `date`(1) man page before attempting to set the date on your system.

TABLE A–4 Table Entry Example 4

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`rev`(1)	N		N

The SunOS 4.2 `rev` command is not available in the SunOS release 5.6 software or the BSD release. There is no replacement command available.

Commands Reference Table

Table A–5 lists all SunOS release 4.x command interfaces, and shows their status in the Solaris 2.6 environment and in the SunOS/BSD Source Compatibility Package.

TABLE A–5 Commands Reference Table

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`C2conv`(8)	N	See your system vendor for information on this product.	N
`C2unconv`(8)	N	See your system vendor for information on this product.	N
`Mail`(1)	A	The `mailx`(1) command provides similar capabilities.	N
`ac`(8)	A	The System Accounting Resource package (SAR) provides most of the accounting capabilities available in `ac`.	N
`acctcms`(8)	S		N
`acctcom`(8)	S		N
`acctcon1`(8)	S		N
`acctcon2`(8)	S		N
`acctdisk`(8)	S		N
`acctdusg`(8)	S		N
`acctmerg`(8)	S		N
`accton`(8)	S		N
`acctprc1`(8)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
acctprc2(8)	S		N
acctwtmp(8)	S		N
adb(1)	S		N
adbgen(8)	S		N
add_client(8)	N	admintool(1M)	N
add_services(8)	A	The swmtool(1M) command provides similar capabilities.	N
addbib(1)	S		N
adjacentscreens(1)	A	The OpenWindows environment has two methods for providing multiple displays: (1) Start two servers on a given machine, each server controlling its specific display. (2) Start one server with two displays, using the openwin –dev option.	N
admin(1)	C	The following SunOS release 4.x options are not available in the SunOS release 5.6 system software: –l*release*[,*release* . . .]Lock indicated release against deltas.	N
adv(8)	N	RFS does not exist. This capability is still accessible via the –f flag.	N
aedplot(1G)	N		S
align_equals(1)	A	The OpenWindows Text menu Indent command provides similar capabilities.	N
analyze(8)	A	Use adb(1) on core files to analyze crashes.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`apropos`(1)	C	The SunOS release 4.x command used the `whatis` database. In theSunOS release 5.6 software, this database is called `windex`, and the format is slightly different.	N
`ar`(1V)	S		N
`ar`(1V) – SysV	C		N
`arch`(1)	C	Without options, this command now returns "sun4." Its use is discouraged. Use `uname`(1) instead. To determine the operating system name and release level, use `uname –sr`.	S
`arp`(8C)	S		N
`as`(1)	C	The following SunOS release 4.x options are not available in the SunOS release 5.6 command: `–d2,–h,–j, –J, –k, –O`[*n*].	N
`at`(1)	S	The `at`, `atq`, and `atrm` commands in SunOS release 5.6 systems behave slightly differently than they do in SunOS release 4.x systems. Security for non-privileged users is more restricted on SunOS release 5.6 systems. Non-privileged users cannot display the jobs of any other user.	N
`atoplot`(1G)	N		S

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`atq`(1)	C	The `at`, `atq`, and `atrm` commands in SunOS release 5.6 systems behave slightly differently than they do in SunOS release 4.x systems. In the SunOS release 4.x command, if no user name is specified, the entire queue is displayed. In SunOS release 5.6 system software, the entire queue is displayed only if the invoker is a privileged user; otherwise, only the jobs belonging to the invoker are displayed. A non-privileged user cannot list the jobs of another user. Security for non-privileged users is more restricted on SunOS release 5.6 systems.	N
`atrm`(1)	C	The `at`, `atq`, and `atrm` commands in SunOS release 5.6 systems behave slightly differently than they do in SunOS release 4.x systems. The SunOS release 4.x '-' flag has been renamed to `–a` in the SunOS release 5.6 command. Security for non-privileged users is more restricted on SunOS release 5.6 systems.	N
`audit`(8)	C	`–d` or `–u` options are not available. This command is available only if the Basic Security Module (BSM) has been enabled.	N
`audit_warn`(8)	S		N
`auditd`(8)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
automount(8)	C	The following SunOS release 4.x option is not available in the SunOS release 5.6 command: –m Suppress initialization of directory-map pairs. The auto.master and auto.home files are renamed auto_master and auto_home. The default home directory path is /export/home/*username*.	N
awk(1)	S		N
banner(1V) – SysV	S		N
bar(1)	tar, cpio	The tar(1) command can replace bar for most uses. You can use cpio –iH bar to restore existing SunOS release 4.x bar backups. You can no longer create bar format files.	N
basename(1)	S	The SunOS release 5.6 and SunOS/BSD Compatibility versions are both compatible to the SunOS release 4.x version, but they differ in how they parse arguments: the SunOS release 5.6 version will not accept more than two arguments, the SunOS/BSD Compatibility version ignores all arguments after the second.	S
batch(1)	S	By default, the SunOS release 5.6 batch job *queuename* is not specified. Jobs were always queued on queue b with the SunOS release 4.x command.	N
bc(1)	S		N
bgplot(1G)	N		S
biff(1)	chmod	fiff n: % chmod u+x 'tty' biff y: % chmod u-x 'tty'	S

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
bin-mail(1)	S	Same as the SunOS release 5.6 mail(1) command.	N
biod(8)	N		N
boot(8S)	C	See the boot(1M) man page for more information.	N
bootparamd(8)	S		N
cal(1)	S		N
calendar(1)	S		N
cancel(1)	S		N
capitalize(1)	C	An OpenWindows version of this command is available with the OpenWindows text editor.	N
captoinfo(8V) – SysV	S		N
cat(1V) – SysV	S		N
cat(1V)	S	The SunOS release 5.6 cat command requires the –v option with the –t and –e options. The SunOS release 5.6 command displays FORMFEED characters with the –t option,instead of the –v option as with the SunOS release 4.x command.	N
catman(8)	S		N
cb(1)	S		N
cc(1V) – SysV	N		N
cc(1V)	N	The C compiler is only available with the C language unbundled tools.	C
cd(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`cdc`(1)	C	The two versions differ in how they treat an unreadable s.file. The SunOS release 4.x command prints an error; the SunOS release 5.6 command silently ignores the error.	N
`cflow`(1V) – SysV	N	The `cflow` command is now available as an unbundled product.	N
`cflow`(1V)	N	The `cflow` command is now available as an unbundled product.	N
`chargefee`(8)	S		
`checkeq`(1)	S		N
`checknr`(1)	S		N
`chfn`(1)	N		N
`chgrp`(1)	C	The default behavior of symbolic links has changed from SunOS release 4.x to SunOS release 5.6 system software. In SunOS release 4.x system software, `chgrp` changed ownership of the symbolic itself; in SunOS release 5.6 system software, `chgrp` follows the link. To change ownership of the symbolic link in SunOS release 5.6 system software, use the `–h` option.	N
`chkey`(1)	S		N
`chmod`(1V) – SysV	C	The SunOS release 5.6 `–R` option changes the mode of the target when symbolic links are encountered.	N
`chmod`(1V)	S	The SunOS release 5.6 `–R` option changes the mode of the target when symbolic links are encountered. The SunOS release 5.6 command supports two additional permissions: 'l' and 'T'.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
chown(8)	C	The default behavior of symbolic links has changed. SunOS release 4.x chown changed ownership of the symbolic link. SunOS release 5.6 chown follows the link. To change the ownership of the link, use chown –h. The SunOS release 5.6 chown command does not allow changing the group ID of a file.	S
chroot(8)	S		N
chrtbl(8)	A	In SunOS release 5.6 localedef(1) creates locale database.	N
chsh(1)	N		N
ckpacct(8)	S		N
clear(1)	S		N
clear_colormap(1)	N		N
clear_functions(1)	S		N
click(1)	N		N
clock(1)	A	An OpenWindows command is available in /usr/demo/clock. See the clock(1) man page for information.	N
clri(8)	S		N
cmdtool(1)	A	This command is replaced by the OpenWindows Command Tool.	N
cmp(1)	S		N
col(1V) – SysV	S		N
col(1V)	C		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`colcrt`(1)	N		N
`colldef`(8)	A	In SunOS release 5.6 `localedef`(1) creates locale database.	N
`coloredit`(1)	A	The function of this command is now handled by the OpenWindows property window.	N
`colrm`(1)	N		N
`comb`(1)	C	The two versions differ in how they treat an unreadable s.file. The SunOS release 4.x command prints an error, but the SunOS release 5.6 command silently ignores the error.	N
`comm`(1)	S		N
`compress`(1)	S		N
`config`(8)	N		N
`cp`(1)	C	The `–R` option is replaced by the `–r` option in the SunOS release 5.6 command.	N
`cpio`(1)	S		N
`cpp`(1)	S		N
`crash`(8)	C	The default name list used in SunOS release 4.x is `/vmunix`, but it is `/kernel/unix` in the SunOS release 5.6 software.	N
`cron`(8)	S		N
`crontab`(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`crtplot`(1G)	N		S
`crypt`(1)	S		N
`csh`(1)	S		N
`csplit`(1V) – SysV	S		N
`ctags`(1)	S		N
`ctrace`(1V) – SysV	N	The following SunOS release 4.x option is not available in the SunOS release 5.6 command: –`b` Use only basic functions to trace code. This option is needed for running under an operating system that does not have the `signal`(), `fflush`(), `longjmp`() or `setjmp`() functions available. The syntax of the –`r` option differs between SunOS release 4.x and SunOS release 5.6 system software. The 4.1 format is –`rf`; it is now –`r f`. `ctrace` is available as an unbundled product.	N
`cu`(1C)	S		N
`cut`(1V) – SysV	S		N
`cxref`(1V) – SysV	S		N
`cxref`(1V)	N	`cxref` is available as an unbundled product.	N
`date`(1V) – SysV	S		N
`date`(1V)	C	The format used when setting the date is slightly different in the SunOS release 5.6 software. See the `date`(1) man page for more information.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`dbconfig`(8)	S		N
`dbx`(1)	N	Available with the unbundled SPARCworks product.	N
`dbxtool`(1)	N	Available with the unbundled SPARCworks product as the command debugger.	N
`dc`(1)	S		N
`dcheck`(8)	A	Use the `fsck`(1M) command for normal consistency checking. The `ncheck`(1M) command replaces the function of `dcheck –i` numbers.	N
`dd`(1)	C	In the SunOS release 4.x command, the size used for the size suffix `w` (words) is in units of 4 bytes, while in SunOS release 5.6 system software, `w` is in units of 2 bytes. `k`, `b`, or `w` may be used as a suffix to specify multiplication by 1024, 512, or 2, respectively. The `unblock` and `block` conversion options are new.	N
`defaults_from_input`(1)		The function of this command is now handled by the OpenWindows property window.	N
`defaults_merge`(1)	S		N
`defaults_to_indentpro`(1)		The function of this command is now handled by the OpenWindows property window.	N
`defaults_to_mailrc`(1)		The function of this command is now handled by the OpenWindows property window.	N
`defaultsedit`(1)		The function of this command is now handled by the OpenWindows property window.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`delta`(1)	C	If a directory is specified as the argument, all files in the directory are processed. In the SunOS release 4.x software, an error is produced if a file in a directory generates an error. Such files are silently ignored by the SunOS release 5.6 command.	N
`deroff`(1)	S		N
`des`(1)	S		N
`devinfo`(8S)	C	The `prtconf`(1M) command provides similar capabilities.	N
`devnm`(8)	C	The output format between SunOS release 4.x and SunOS release 5.6 system software is quite different. In SunOS release 4.x system software, the name argument is optional. In the SunOS release 5.6 system software, it is required.	N
`df`(1V) – SysV	C		N
`df`(1V)	C	The SunOS release 4.x version of this command provides a different output format containing somewhat different output than the SunOS release 5.6 `df` command. The SunOS release 5.6 `–k` option provides output formats similar to those in the SunOS release 4.x command. The SunOS release 4.x `df –t` *filesystem* type reports on files of the specified type, whereas the SunOS release 5.6 `df –t` command prints full listings with totals. You can use `df –l` to see local file systems.	S

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`diff`(1)	C	The behavior of several flags differs between the two versions. In SunOS release 4.x system software, the –`c` option takes an optional argument for the number of lines to display for each difference. If no argument is given, the default is 3 lines. In the SunOS release 5.6 command, a space is required between the –`S` option and its argument.	N
`diff3`(1V) – SysV	S		N
`diff3`(1V)	S		N
`diffmk`(1)	S		N
`dircmp`(1V) – SysV	S		N
`dirname`(1V) – SysV	S		N
`dis`(1)	C	The following SunOS release 4.x option is not available in the SunOS release 5.6 command: –`da` *sec* Disassemble *sec* as data, printing the actual address of the data. Use the SunOS release 5.6 –`D` *sec* option to do the same thing.	N
`diskusg`(8)	A	The `acctdusg` (1M) command provides similar capabilities.	N
`dkctl`(8)	N		N
`dkinfo`(8)	A	The `prtvtoc`(1M) command provides similar capabilities.	N
`dmesg`(8)	S		N
`dname`(8)	N	RFS is not available.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`dodisk`(8)	S		N
`domainname`(1)	S		N
`dorfs`(8)	N	RFS is not available.	N
`dos2unix`(1)	S		N
`du`(1V) – SysV	S		N
`du`(1V)	C	The SunOS release 4.x command reports the disk usage in kilobytes while the SunOS release 5.6 `du` command reports disk usage in 512-byte blocks. The –`k` option can be used to report usage in kilobytes.	S
`dumbplot`(1G)	N		S
`dump`(8)	A	The `ufsdump` command provides similar capabilities. The following SunOS release 4.x options are not in the SunOS release 5.6 command: –`a` *archive-file* The SunOS release 5.6 –`a` option dumps the archive header of each member of an archive. –`D` Specify diskette as the dump media. The SunOS release 5.6 –`D` option dumps debugging information. –`v` Verify against the file system being dumped. The SunOS release 5.6 –`v` option dumps information in symbolic, rather than numeric, representation.	N
`dumpfs`(8)	A	The `fstyp` –`F` –`ufs` –`v` command provides similar capabilities.	N
`dumpkeys`(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
e(1)	A	The ex(1) command provides similar capabilities.	S
echo(1V) – SysV	S		N
echo(1V)	C	The SunOS release 4.x –n option suppressed new-line printing. Use a \c in the SunOS release 5.6 software.	S
ed(1)	S		N
edit(1)	S		N
edquota(8)	S		N
eeprom(8S)	S		N
egrep(1V)	S		N
eject(1)	S		N
enroll(1)	N		N
env(1)	S		N
eqn(1)	S		N
error(1)	S		N
etherd(8C)	A	The snoop(1M) command provides similar capabilities.	N
etherfind(8C)	A	The snoop(1M) command provides similar capabilities.	N
ex(1)	S		N
expand(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`exportfs`(8)	A	The `share`(1M) command provides similar capabilities.	N
`expr`(1V) – SysV	S		N
`expr`(1V)	C		S
`extract_files`(8)	A	The `pkgadd`(1M) command provides similar capabilities.	N
`extract_patch`(8)	A	The `pkgadd`(1M) command provides similar capabilities.	N
`extract_unbundled`(8)	A	The `swmtool`(1M) command provides similar capabilities.	N
`false`(1)	S		N
`fastboot`(8)	A	The `init 6` command provides similar capabilities.	S
`fasthalt`(8)	A	The `init 0` command provides similar capabilities.	S
`fdformat`(1)	S		N
`fgrep`(1V)	S		N
`file`(1)	C	The following SunOS release 4.x option is not in the SunOS release 5.6 command: `–L` If a file is a symbolic link, test the file referenced by the link rather than the link itself.	S
`find`(1)	C	The following SunOS release 4.x option is not available in the SunOS release 5.6 command: `–n` *cpio-device* Write the current file on device in `cpio –c` format.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`finger`(1)	S		N
`fingerd`(8)	S		N
`fmt`(1)	C		N
`fmt_mail`(1)	N		N
`fold`(1)	S		N
`fontedit`(1)	N		N
`foption`(1)	N		N
`format`(8S)	S		N
`fpa_download`(8)	N		N
`fparel`(8)	N		N
`fpaversion`(8)	N		N
`fpurel`(8)	N		N
`fpuversion4`(8)	A	This information is available from `psrinfo –v`	N
`from`(1)	N		S
`fsck`(8)	C	The SunOS release 4.x `fsck` command differs significantly from the SunOS release 5.6 command. With the SunOS release 5.6 command, you specify most options after you specify the file system type. `fsck –m` does a quick file-system check. The `–w` option is not available. New options include `–f`, `–v`, and `–o`.	N
`fsck-cdrom`(8)	N		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`fsirand`(8)	S		S
`ftp`(1C)	S		N
`ftpd`(8C)	S		N
`fumount`(8)	S	RFS is no longer available	N
`fusage`(8)	S	RFS is no longer available	N
`fuser`(8)	S		N
`fwtmp`(8)	S		N
`gcore`(1)	S		N
`generic_args`(1)	N		N
`get`(1)	C	The SunOS release 5.6 command generates only ASCII files; there is no such restriction in SunOS release 4.x system software. If a directory is specified and the files inside the directory cannot be obtained successfully, the SunOS release 4.x command reports an error; the SunOS release 5.6 command ignores them silently.	N
`get_alarm`(1)	N		N
`get_selection`(1)	A	The `xv_get_sel`(1) command provides similar capabilities.	N
`getopt`(1V) – SysV	S		N
`getoptcvt`(1)	S		N
`getopts`(1)	S		N
`gettable`(8C)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`getty`(8)	S		N
`gfxtool`(1)	N		N
`gigiplot`(1G)	N		S
`glob`(1)	S		N
`goto`(1)	S		N
`gpconfig`(8)	N		N
`gprof`(1G)	S		N
`graph`(1G)	S		N
`grep`(1V)	S		N
`grep`(1V) – SysV	C	The following option has changed: `–w` Search for the regular expression as a word as if surrounded by `\<` and `\>`.	N
`groups`(1)	S		S
`grpck`(8V)	S		N
`gxtest`(8S)	N		N
`halt`(8)	S		N
`hashcheck`(1)	S		N
`hashmake`(1)	S		N
`hashstat`(1)	S		N
`head`(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
help(1)	S		N
help_open(1)	S		N
hostid(1)	S		S
hostname(1)	S		S
hostrfs(8)	N	RFS is not available.	N
hp7221plot(1G)	N		S
hpplot(1G)	N		S
htable(8)	S		N
i386(1)	S		N
iAPX286(1)	S		N
icheck(8)	A	fsdb() is an alternate command.	N
iconedit(1)	A	This command is replaced by the OpenWindows Icon Edit tool.	N
id(1)			
id(1V) – SysV	S		N
idload(8)	N	RFS is not available.	N
ifconfig(8C)	S		N
imemtest(8C)	N		N
implot(1G)	N		Y
in.comsat(8C)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
in.fingerd(8C)	S		N
in.ftpd(8C)	S		N
in.named(8C)	S		N
in.rexecd(8C)	S		N
in.rlogind(8C)	S		N
in.routed(8C)	S		N
in.rshd(8C)	C	The port range differs between the SunOS release 4.x and SunOS release 5.6 commands. In SunOS release 4.x system software, the range is 512-1023; in SunOS release 5.6 system software, it is 0-1023.	N
in.rwhod(8C)	S		N
in.talkd(8C)	S		N
in.telnetd(8C)	S		N
in.tftpd(8C)	S		N
in.tnamed(8C)	S		N
in.uucpd(8C)	S		N
indent(1)	N	This command is now available as an unbundled product.	N
indentpro_to_defaults(1)	A	The function of this command is now handled by the OpenWindows property sheets.	N
indxbib(1)	S		N
inetd(8C)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`infocmp`(8V) – SysV	C		N
`infocmp`(8V)	C	The syntax of the –`s` option differs between SunOS release 4.x and SunOS release 5.6 system software. In the SunOS release 5.6 command, there must be a space between –`s` and its argument. In the SunOS release 4.x command, the space is optional.	N
`init`(8)	C	The SunOS release 5.6 command is very different from the SunOS release 4.x command. See the `init`(1M) man page for more information.	N
`inline`(1)	N	This command is now available as an unbundled product.	N
`input_from_defaults`(1)	N		N
`insert_brackets`(1)	A	An OpenWindows command with the same name is available with the OpenWindows Text Editor.	N
`install`(1)	C	The functions of the –`c`, –`o`, and –`s` options are different between the SunOS release 4.x and SunOS release 5.6 commands.	S
`installboot`(8S)	C	The path names and syntax have changed.	N
`installtxt`(8)	A	The `msgfmt`(1) command provides similar capabilities.	N
`intr`(8)	N		N
`iostat`(8)	S	New options: –`x` Provide disk statistics –`c` Report the percentage of time the system has spent in user mode, system mode, and idle.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`ipallocd`(8C)	N		N
`ipcrm`(1)	S		N
`ipcs`(1)	S		N
`join`(1)	C	In the SunOS release 4.x command, the `–a` option takes an argument whose value can be 1, 2, or 3. In the SunOS release 5.6 system software, this value can only be 1 or 2. In the SunOS release 4.x command, the argument to `–j` can only be 1 or 2; there is no such restriction in the SunOS release 5.6 command.	N
`kadb`(8S)	S		N
`keyenvoy`(8C)	N		N
`keylogin`(1)	S		N
`keylogout`(1)	S		N
`keyserv`(8C)	S		N
`kgmon`(8)	S		N
`kill`(1)	S		N
`labelit`(8)	S		N
`last`(1)	S		N
`lastcomm`(1)	S		N
`lastlogin`(8)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
ld(1)	C	There are many differences between the SunOS release 4.x ld command and the SunOS release 5.6 command. The following SunOS release 4.x options are not available: –align, –A, –B,– -D, –M, –n, –t, –T, –Tdata, –x, –X, –y and –z. The –assert option has been replaced by the –z option. The –d, –dc, –dp options are the default in SunOS release 5.6 system software. To turn off these options use –b.	S
ldconfig(8)	N		N
ldd(1)	S		N
leave(1)	N	The cron(1M) and at(1) commands provide similar capabilities.	N
lex(1)	C	The following SunOS release 4.x option is not available in the SunOS release 5.6 command: –f Compile faster by not packing resulting tables. This option is limited to small programs.	N
line(1)	S		N
link(8V)	S		N
lint(1V) – SysV	N		N
lint(1V)	N	Available with the unbundled SPARCworks product.	S
listen(8)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`ln`(1V)	C	The SunOS release 4.x `ln` command never removes the target if it already exists. The SunOS release 5.6 `ln` command removes the target, given the proper permissions. The SunOS release 4.x –`f` option forces a hard link to a directory.	S
`ln`(1V) – SysV	C	In SunOS release 4.x `/usr/5bin/ln`, the –`f` option forces files to be linked without displaying permissions, asking questions, or reporting errors. The `/usr/5bin/ln` –`F` option to force a hard link to a directory is not available in SunOS release 5.6 system software.	N
`loadkeys`(1)	S		N
`lockd`(8C)	S		N
`lockscreen`(1)	A	This command is available as the OpenWindows tool `xlock`(1). The capabilities of the `lockscreen` command remains the same in `xlock`, although the foreground pattern differs.	N
`logger`(1)	N		S
`login`(1)	S		N
`logname`(1)	S		N
`look`(1)	S		N
`lookbib`(1)	S		N
`lorder`(1)	S		N
`lp`(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`lpc`(8)	A	The `lpadmin`(1M) command provides similar capabilities.	S
`lpd`(8)	A	The `lpadmin`(1M) command provides similar capabilities.	S
`lpq`(1)	A	The `lpstat`(1) command provides similar capabilities.	S
`lpr`(1)	A	The `lp`(1) command provides similar capabilities.	S
`lprm`(1)	A	The `cancel`(1) command provides similar capabilities.	S
`lpstat`(1)	S		N
`lptest`(1)	N		S
`ls`(1V) – SysV	C		N
`ls`(1V)	S		S
`lsw`(1)	N		N
`m4`(1V)	C	Some small syntactic incompatibilities over expression evaluation.	N
`m4`(1V) – SysV	S		N
`m68k`(1)	S		N
`mach`(1)	S		S
`mail`(1) – UCB	`mailx`		S

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`mail`(1)	C	Now in `/usr/bin/mail`, was in `/usr/ucb/mail` in the SunOS release 4.x software. This entry refers to the mail command installed under `/usr/bin/mail`. The SunOS release 4.x mail is compatible with the SunOS release 5.6 command except for the following: `–i` The `–i` (ignore interrupts) option is not available. In the SunOS release 4.x command, the postmark line is preceded by a '>'; this is not required by the SunOS release 5.6 command.	N
`mailrc_to_defaults`(1)	C	The function of this command is now handled by the OpenWindows property window.	N
`mailstats`(8)	S		N
`mailtool`(1)	C	This command is available as the OpenWindows Mail Tool.	N
`make`(1)	S	SVR4 & SVID make is available in `/usr/ccs/lib/svr4.mke`	N
`makedbm`(8)	C	The SunOS release 5.6 interface for this command is compatible with the SunOS release 4.x interface. The SunOS release 5.6 version uses `/usr/lib/ndbm` rather than `/usr/lib/dbm` as the SunOS release 4.x version does.	N
`makedev`(8)	N		N
`makekey`(8)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
man(1)	C	The organization of the on-line man pages has changed. Refer to intro(1) for a description of all sections. The man command now allows you to specify a default order of directories for man to search. Two new options make it easier to find man pages: –a to display all man pages matching *title* in the order found; and –l to list all man pages matching *title*. Also, the –s option replaces the *section number* argument.	N
mc68010(8)	S		N
mc68020(8)	S		N
mc68881version(8)	N		N
mconnect(8)	S		N
mesg(1)	S		N
mkdir(1)	S		N
mkfile(8)	S		N
mkfs(8)	C	The interface differs significantly between the two versions. The SunOS release 5.6 command provides for different file system types.	N
mknod(8)	S		N
mkproto(8)	C		N
mkstr(1)	N		S
modload(8)	C	Modules are usually automatically loaded using modload.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`modstat`(8)	A	The `modinfo`(1M) command provides similar capabilities.	N
`modunload`(8)	C	Modules are usually automatically unloaded.	N
`monacct`(8)	S		N
`more`(1)	S		N
`mount`(8)	C	The interface differs significantly between the two versions. In the SunOS release 5.6 version, most options must be specified after the file system type has been specified (unless the file system is entered in `/etc/vfstab`).	N
`mount_tfs`(8)	N		N
`mountd`(8C)	S		N
`mt`(1)	S		N
`mv`(1)	S		N
`named`(8C)	C	The name deamon is renamed to `in.named`.	N
`nawk`(1)	S		N
`ncheck`(8)	C	Modified to allow specification of different file system types.	N
`ndbootd`(8C)	N		N
`neqn`(1)	S		N
`netstat`(8C)	S		N
`newaliases`(8)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`newfs`(8)	S	Moved from `/usr/etc/newfs` to `/usr/sbin/newfs`.	N
`newgrp`(1)	S		N
`newkey`(8)	S		N
`nfsd`(8)	S		N
`nfsstat`(8C)	S		N
`nice`(1)	C	There are two versions of `nice` in SunOS release 4.x system software, one built into the `csh` and one installed under `/usr/bin`. The default process priority for the command built into `csh` is 4, and the default value for `/usr/bin/nice` is 10. The SunOS release 5.6 command defaults to 10. The SunOS release 4.x command that is built into the `csh` uses a slightly different syntax than the SunOS release 4.x command found in `/usr/bin`, in that the additional `–+` option (`nice –+`*n*) sets the `nice` value to *n* rather than incrementing it by *n*.	N
`nl`(1V) – SysV	S		N
`nlsadmin`(8)	C	The function of the `–l` option differs between the versions. In the SunOS release 4.x software, changing *addr* does not take effect until the next time the listener for that network is started. In the SunOS release 5.6 software, it happens immediately. In the SunOS release 4.x software, *addr* can be specified in hexadecimal notation while in the SunOS release 5.6 software it cannot. The SunOS release 4.x `–m` option is not available in the SunOS release 5.6 version. This option is used to add a new service to the list of services available through the indicated listener.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
nm(1)	C	The following SunOS release 4.x options are not available with the SunOS release 5.6 version: –g, –p, –s, and –a. The SunOS release 4.x and SunOS release 5.6 versions of the –n, –o and –r options differ.	N
nohup(1V)	C		N
nohup(1V) – SysV	S		N
nroff(1)	S		N
nslookup(8C)	S		N
nsquery(8)	S		N
nulladm(8)	S		N
od(1V)	S		N
od(1V) – SysV	S		N
old-analyze(8)	N		N
old-ccat(1)	N		N
old-clocktool(1)	N		N
old-compact(1)	N		N
old-eyacc(1)	N		N
old-filemerge(1)	N		N
old-make(1)	N		N
old-perfmon(1)	N		N
old-prmail(1)	N		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
old-pti(1)	N		N
old-setkeys(1)	N		N
old-sun3cvt(1)	N		N
old-syslog(1)	N		N
old-uncompact(1)	N		N
old-vc(1)	N		N
on(1C)	S		N
overview(1)	N		N
pac(8)	N		N
pack(1V)	S		N
pack(1V) – SysV	S	With the SunOS release 4.x /usr/5bin/pack command, file names are restricted to 12 characters. In SunOS release 5.6 system software, they are restricted to {NAME_MAX} - 2. The SunOS release 5.6 pack and unpack commands are compatible with the SunOS release 4.x commands.	N
page(1)	S		N
pagesize(1)	S		S
passwd(1)	C	The –F *filename* option is not available. The –f and –s options have different meanings. The –f option forces the user to change the password at the next login. The –s option displays the password attributes for the user's login name.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`paste`(1V) – SysV	S		N
`pax`(1V)	C		N
`paxcpio`(1V)	A	The `cpio`(1) and `pax`(1) commands provide similar capabilities.	N
`pcat`(1V) – SysV	S		N
`pdp11`(1)	S		N
`perfmeter`(1)	A	This command is available in theSunOS release 5.6 software as the OpenWindows Performance Meter tool.	N
`pg`(1V) – SysV	S		N
`ping`(8C)	S		N
`plot`(1G)	N		S
`plottoa`(1G)	N		S
`portmap`(8C)	A	The `rpcbind`(1M) daemon provides similar capabilities.	N
`pr`(1V)	C		N
`pr`(1V) – SysV	S		N
`praudit`(8)	S		N
`prctmp`(8)	S		N
`prdaily`(8)	S		N
`printenv`(1)	A	The `env`(1) command provides similar capabilities.	S

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`prof`(1)	C	The SunOS release 4.x –`v` option is not available with SunOS release 5.6 system software. This option suppresses all printing and produces a graphic version of the profile on the standard output for display by the `plot`(1) filters. The SunOS release 4.x –`a` option requests that all symbols be reported: in the SunOS release 5.6 command, just external symbols are reported.	N
`prs`(1)	C	The versions differ in how they treat an unreadable s.file. The SunOS release 4.x command prints an error and continues if it encounters an unreadable s.file. The SunOS release 5.6 command silently ignores the error.	N
`prt`(1)	S		N
`prtacct`(8)	S		N
`ps`(1)	C	The following SunOS release 4.x options are not available with SunOS release 5.6 system software: –`C`, –`k`, –`n`, –`r`, –`S`, –`U`, –`v`, –`w`, and –`x`. The following option has different meanings in the two versions: –`c` In the SunOS release 4.x command, this option displays the command name. In the SunOS release 5.6 command, it prints information in a format that reflects the new process scheduler design.	S
`pstat`(8)	A	The `sar`(1M) command provides similar capabilities. `swap` –`s` shows the total amount of swap space available on the system.	N
`ptx`(1)	N		N
`pwck`(8V)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`pwd`(1)	S		N
`pwdauthd`(8C)	N	Similar capabilities will be available in future releases with unbundled products. See your system vendor for information on this product.	N
`quot`(8)	S		N
`quota`(1)	S		N
`quotacheck`(8)	S		N
`quotaoff`(8)	S		N
`quotaon`(8)	S		N
`ranlib`(1)	C	The `ar`(1) command automatically provides similar capabilities. `ranlib` remains as a null script.	N
`rarpd`(8C)	S		N
`rasfilter8to1`(1)	N		N
`rastrepl`(1)	N		N
`rc`(8)	N	The configuration scripts under `/etc/init.d` provide similar capabilities. The organization of rc files has changed in SunOS release 5.6 systems. They are now divided by run levels.	N
`rc.boot`(8)	N	The configuration scripts under `/etc/init.d` provide similar capabilities.	N
`rc.local`(8)	N	The configuration scripts under `/etc/init.d` provide similar capabilities.	N
`rcp`(1C)	S		N
`rdate`(8C)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`rdist`(1)	S		N
`rdump`(8)	A	The `ufsdump`(1M) command provides similar capabilities.	N
`reboot`(8)	S		N
`red`(1)	S		N
`refer`(1)	S		N
`rehash`(1)	S		N
`remove_brackets`(1)	A	A version of this command is available with the OpenWindows Text Editor.	N
`renice`(8)	A	The `priocntl`(1) command provides similar capabilities.	S
`repquota`(8)	S		N
`reset`(1)	A	`stty` provides similar capabilities.	S
`restore`(8)	A	The SunOS release 5.6 command, `ufsrestore`, has been enhanced to take advantage of the end-of-media detection done by `ufsdump`.	N
`rev`(1)	N		N
`rexd`(8C)	A	`in.rexd` provides similar capabilities.	N
`rexecd`(8C)	A	`in.rexcd` provides similar capabilities.	N
`rfadmin`(8)	N	RFS is not available.	N
`rfpasswd`(8)	N	RFS is not available.	N
`rfstart`(8)	N	RFS is not available.	N
`rfstop`(8)	N	RFS is not available.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`rfuadmin`(8)	N	RFS is not available.	N
`rfudaemon`(8)	N	RFS is not available.	N
`ring_alarm`(1)	N		N
`rlogin`(1C)	C	The ~`dsusp` sequence for escapes on SunOS release 4.x system software is not available with the SunOS release 5.6 command. Also, the syntax for the –`e` option differs between the SunOS release 4.x and SunOS release 5.6 commands. In SunOS release 4.x system software, the syntax is –e*c*; in SunOS release 5.6 system software, it is –`e` *c*.	N
`rlogind`(8C)	A	`in.rlogind` provides similar capabilities.	N
`rm`(1)	S		N
`rm_client`(8)	A	The `admintool`(1M) utility replaces this command on SunOS release 5.6 systems.	N
`rm_services`(8)	A	The `swmtool`(1M) command provides similar capabilities	N
`rmail`(8C)	C	The SunOS release 4.x version handles remote mail received using `uucp`(1C). It is explicitly designed for use with `uucp`(1C) and `sendmail`(8). The SunOS release 5.6 `rmail` is a link to `mail`(1) and is a command used for reading mail.	N
`rmdel`(1)	C	The versions differ in how they treat an unreadable s.file. The SunOS release 4.x command prints an error and continues if it encounters an unreadable s.file. The SunOS release 5.6 command silently ignores the error.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`rmdir`(1)	S		N
`rmntstat`(8)	N	RFS is not available.	N
`rmt`(8C)	S		N
`roffbib`(1)	S		N
`route`(8C)	C	The SunOS release 4.x route command uses `gethostent`(3) to look up all symbolic names and gateways, while the SunOS release 5.6 command uses `gethostbyname`(3).	N
`routed`(8)	A	`in.routed` provides similar capabilities.	N
`rpc.bootparamd`(8)	S		N
`rpc.etherd`(8C)	N	`snoop`(1m) obsoletes this daemon.	N
`rpc.lockd`(8C)	A	`lockd` provides similar capabilities.	N
`rpc.mountd`(8C)	A	`mountd` provides similar capabilites.	N
`rpc.rexd`(8C)	S		N
`rpc.rquotad`(8C)	S		N
`rpc.rstatd`(8C)	S	Now in `/usr/lib/netsvc/rstat`.	N
`rpc.rusersd`(8C)	S	Now in `/usr/lib/netsvc/rusers`.	N
`rpc.rwalld`(8C)	S	Now in `/usr/lib/netsvc/rwall`.	N
`rpc.showfhd`(8C)	A	The `showfhd`(1M) command provides similar capabilities.	N
`rpc.sprayd`(8C)	S	Now in `/usr/lib/netsvc/spray`.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`rpc.statd`(8C)	S	Now in `/usr/lib/netsvc/rstat`.	N
`rpc.user_agentd`(8C)	N		N
`rpc.yppasswdd`(8C)	N		N
`rpc.ypupdated`(8C)	N		N
`rpcgen`(1)	S		N
`rpcinfo`(8)	S		N
`rrestore`(8)	A	The `ufsrestore`(1M) command provides similar capabilities.	N
`rsh`(1C)	S		N
`runacct`(8)	S		N
`rup`(1C)	S		N
`ruptime`(1C)	S		N
`rusage`(8)	N		S
`rusers`(1C)	S		N
`rwall`(1C)	S		N
`rwho`(1C)	S		N
`sa`(8)	A	`acct` (1M) provides similar capabilities.	N
`sact`(1)	C	The versions differ in how they treat an unreadable s.file. The SunOS release 4.x command will print an error and continue if it encounters an unreadable s.file. The SunOS release 5.6 command silently ignores the error.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`savecore`(8)	S		N
`sccs`(1)	S		N
`sccs-admin`(1)	S		N
`sccs-cdc`(1)	S		N
`sccs-comb`(1)	S		N
`sccs-delta`(1)	S		N
`sccs-get`(1)	S		N
`sccs-help`(1)	S		N
`sccs-prs`(1)	S		N
`sccs-prt`(1)	S		N
`sccs-rmdel`(1)	S		N
`sccs-sact`(1)	S		N
`sccs-sccsdiff`(1)	S		N
`sccs-unget`(1)	S		N
`sccs-val`(1)	S		N
`sccsdiff`(1)	C		N
`screenblank`(1)	C	The OpenWindows `xset –s –600` command provides similar capabilities.	N
`screendump`(1)	N		N
`screenload`(1)	N		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`script`(1)	S		N
`scrolldefaults`(1)	C	The function of this command is now handled by the OpenWindows property window.	N
`sdiff`(1V) – SysV	S		N
`sed`(1V) – SysV	S		N
`sed`(1V)	C	The SunOS release 4.x `/usr/5bin/sed` and the SunOS release 5.6 commands do not strip initial SPACE and TAB characters from text lines.	S
`selection_svc`(1)	N		N
`sendmail`(8)	S		N
`set4`(8)	N		N
`set_alarm`(1)	N		N
`setkeys`(1)	N		N
`setsid`(8V)	N		N
`setup_client`(8)	N		N
`setup_exec`(8)	N		N
`sh`(1)	C	Under SunOS release 4.x system software, the behavior of the builtins `echo` and `test` depend on the relative positions of `/usr/bin` and `/usr/5bin` in the environment variable `PATH`. The behavior is now triggered by the relative ordering of `/usr/ueb` and `/usr/bin`.	N
`shelltool`(1)	C	This command is available as an OpenWindows Shell Tool.	N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
shift_lines(1)	C	An OpenWindows command is available with the OpenWindows Text Editor.	N
showfh(8C)	N		N
showmount(8)	S		N
shutacct(8)	S		N
shutdown(8)	C	The SunOS release 4.x command is very different from the SunOS release 5.6 shutdown(1M) command. By default, the SunOS release 5.6 shutdown(1M) asks for confirmation before starting shutdown activities, while the SunOS release 4.x shutdown(8) does not ask for confirmation. In addition, the following SunOS release 4.x options are not present in the SunOS release 5.6 command: –f,–h, –k, –n, –r.	S
size(1)	C	The SunOS release 4.x command prints sizes in hexadecimal and decimal, and the file name is optional (with a.out as the default). The SunOS release 5.6 command prints them only in decimal, unless the –o or –x option is specified, and the file name is required.	N
skyversion(8)	N		N
sleep(1)	S		N
soelim(1)	S		N
sort(1V) – SysV	S		N
sort(1V)	C		N
sortbib(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`sparc`(1)	S		N
`spell`(1)	C	The SunOS release 4.x –h *spellhist* option is not available with the SunOS release 5.6 command. This option places misspelled words with a user / date stamp in *spellhist*.	N
`spellin`(1)	S		N
`spline`(1G)	S		N
`split`(1)	S		N
`spray`(8C)	C	The SunOS release 4.x–`i` *delay* option is not available with the SunOS release 5.6 command. This option specifies that ICMP echo packets should be used rather than RPC.	N
`startup`(8)	S		N
`strings`(1)	S		N
`strip`(1)	S		N
`stty`(1V) – SysV	C		N
`stty`(1V)	C	The following SunOS release 4.x options are not supported by SunOS release 5.6 stty command: `decctlq`, `tandem`, `cbreak`, `ctlecho`, `prterase`, `crtkill`, `cols`, `tab3`, `crt`, `dec`, `term`.	S
`stty_from_defaults`(1)	N		N
`su`(1V) – SysV	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
su(1V)	C	The SunOS release 4.x –f option is not supported by the SunOS release 4.x /usr/5bin/su or SunOS release 5.6 su command. This option was used for a fast su with csh.	N
sum(1V) – SysV	S		N
sum(1V)	C		S
sun(1)	S		N
sundiag(8)			N
suninstall(8)	C	The command to install SunOS release 5.6 software is still called suninstall, but the installation procedure has changed completely. See *Installation Instructions for Solaris 2.6 (SPARC Platform Edition).*	N
sunview(1)	A	OpenWindows replaces SunView in SunOS release 5.6 systems.	N
sv_acquire(1)	N		N
sv_release(1)	N		N
swapon(8)	A	The swap(1M) command provides similar capabilities. In general, options to the SunOS release 5.6 swap command replace capabilities of individual swap-related commands, such as swapon, in SunOS release 4.x systems.	N
swin(1)	N		N
switcher(1)	N		N
symorder(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`sync`(1)	S		N
`sys-unconfig`(8)	S		N
`syslogd`(8)	S		N
`t300`(1G)	N		S
`t300s`(1G)	N		S
`t4013`(1G)	N		S
`t450`(1G)	N		S
`tabs`(1V) – SysV	S		N
`tail`(1)	S		N
`talk`(1)	S		N
`tar`(1)	S		N
`tbl`(1)	S		N
`tcopy`(1)	S		N
`tcov`(1)	N	Available as an unbundled product.	N
`tee`(1)	S		N
`tek`(1G)	N		S
`tektool`(1)	N		N
`telnet`(1C)	S		N
`test`(1V) – SysV	S		N
`test`(1V)	C		S

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`textedit`(1)	A	This command is available as the OpenWindows Text Edit tool.	N
`textedit_filters`(1)	A	An OpenWindows command is available with the OpenWindows Text Editor.	N
`tfsd`(8)	N		N
`tftp`(1C)	S		N
`tic`(8V)	S		N
`time`(1V) – SysV	S		N
`time`(1V)	C	The SunOS release 4.x command provides a different output than the SunOS release 4.x `/usr/5bin/time` and the SunOS release 5.6 command. The SunOS release 4.x `time` prints the elapsed time, the time spent in the system, and the time spent executing the command all on one line, instead of on three separate lines.	N
`tip`(1C)	S		N
`toolplaces`(1)	N		N
`touch`(1V) – SysV	S		N
`touch`(1V)	C	The SunOS release 4.x –`f` option is not available. This option attempts to force the `touch` in spite of read and write permissions on *filename*.	S
`tput`(1V) – SysV	S		N
`tr`(1V) – SysV	S		N
`tr`(1V)	C		S

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`trace`(1)	A	The `truss`(1) command provides similar capabilities.	N
`traffic`(1C)	N		N
`troff`(1)	S		N
`trpt`(8C)	N		N
`true`(1)	S		N
`tset`(1)	N		S
`tsort`(1)	S		N
`tty`(1)	S		N
`ttysoftcar`(8)	N		N
`tunefs`(8)	S		N
`turnacct`(8)	S		N
`tvconfig`(8)	N		N
`tzsetup`(8)	N		N
`u370`(1)	S		N
`u3b`(1)	S		N
`u3b15`(1)	S		N
`u3b2`(1)	S		N
`u3b5`(1)	S		N
`ul`(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
umask(1)	S		N
umount(8)	C	The interface differs significantly between the two versions. In the SunOS release 5.6 command, most options are changed and must be supplied as file system-specific options.	N
umount_tfs(8)	N		N
unadv(8)	N	RFS not available.	N
uname(1)	S		N
uncompress(1)	S		N
unconfigure(8)	N		N
unexpand(1)	S		N
unget(1)	C	The versions differ in how they treat an unreadable s.file. The SunOS release 4.x version will print an error and continue if it encounters an unreadable s.file. The SunOS release 5.6 version silently ignores the error.	N
unifdef(1)	S		N
uniq(1)	S		N
units(1)	S		N
unix2dos(1)	S		N
unlink(8V)	S		N
unpack(1V) – SysV	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
unpack(1V) – SysV	C	With the SunOS release 4.x /usr/5bin/pack command, file names are restricted to 12 characters. In SunOS release 5.6 system software, they are restricted to {NAME_MAX} - 2. The SunOS release 5.6 pack and unpack commands are compatible with the SunOS release 4.x commands.	
unwhiteout(1)	N		N
update(8)	A	The fsflush process provides this capability.	N
uptime(1)	A	The who –u command provides similar capabilities.	S
users(1)	A	The who –q provides similar capabilities.	S
ustar(1V)	A	The tar(1) command provides similar capabilities.	N
uucheck(8C)	S		N
uucico(8C)	S		N
uucleanup(8C)	S		N
uucp(1C)	S		N
uudecode(1C)	S		N
uuencode(1C)	S		N
uulog(1C)	C	The –u option, which allows printing of information about work done for a specified username, is no longer supported.	N
uuname(1C)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`uupick`(1C)	S		N
`uusched`(8C)	S		N
`uusend`(1C)	N		N
`uustat`(1C)	S		N
`uuto`(1C)	S		N
`uux`(1C)	S		N
`uuxqt`(8C)	S		N
`vacation`(1)	S		N
`val`(1)	S		N
`vax`(1)	S		N
`vedit`(1)	S		N
`vfontinfo`(1)	N		N
`vgrind`(1)	S		N
`vi`(1)	S		N
`view`(1)	S		N
`vipw`(8)	N		S
`vmstat`(8)	C	The `–f` option is no longer available.	N
`vplot`(1)	N		S
`vswap`(1)	N		N
`vtroff`(1)	N		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
vwidth(1)	N		N
w(1)	S		N
wait(1)	S		N
wall(1)	S		N
wc(1)	S		N
what(1)	S		N
whatis(1)	C		N
whereis(1)	N		S
which(1)	S		N
who(1)	S		N
whoami(1)	A	The id(1) command provides similar capabilities. The id command prints the user name and user and group IDs, instead of just the user name.	S
whois(1)	S		N
write(1)	S		N
xargs(1V) – SysV	S		N
xget(1)	N		N
xsend(1)a	N		N
xstr(1)	S		N
yacc(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
yes(1)	N		N
ypbatchupd(8C)	N		N
ypbind(8)	S	Now in /usr/lib/netsvc/yp.	N
ypcat(1)	S		N
ypinit(8)	S		N
ypmatch(1)	S		N
yppasswd(1)	S	The yppasswd command is still available on SunOS release 5.6 systems to access the password information on NIS servers. The equivalent command for NIS+ databases is nispasswd(1). The passwd(1) command can handle passwords in all supported databases (NIS, NIS+, files).	N
yppoll(8)	S		N
yppush(8)	N		N
ypserv(8)	N		N
ypset(8)	S		N
ypupdated(8C)	N		N
ypwhich(8)	S		N
ypxfr(8)	S	Now in /usr/lib/netsvc/yp.	N
ypxfrd(8)	S		N
zcat(1)	S		N

TABLE A–5 Commands Reference Table *(continued)*

SunOS Release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	BSD
`zdump`(8)	S		N
`zic`(8)	S		N

APPENDIX **B**

System Calls Reference Table

This appendix contains the System Calls reference table.This table lists all SunOS release 4.x, and shows their status in the following environments: Solaris 2.6, the ABI, the SVID, SVR4, and the SunOS/BSD Source Compatibility Package.

Using the Reference Table

- If an interface is listed as “changed” (C), a brief description of differences between the SunOS release 4.x system call and the Solaris 2.6 system call is provided.
- If an interface is listed as “the same” (S), the Solaris 2.6 interface will support all features of the SunOS release 4.x interface. In some cases the interface has been enhanced, but can be considered a complete superset of the SunOS release 4.x interface. Note that many system calls are now available as library routines. The Notes column will show the new routine man page reference.
- If an interface has an “alternative” (A), check the Notes section for its replacement.
- If an interface is listed as “not available” (N), you cannot use that interface.
- If the interface includes `errno` values that are not supported in the standard, it is indicated with "#". `errno` differences do not necessarily break compatibility. Note that although `EDQUOT`, `EFAULT`, and `EIO` are often not listed with ABI or SVID, these `errno` values are supported by an ABI or SVID compliant system if appropriate.

The SunOS release 4.x software offers a System V Software installation option that provides System V compatible versions of many utilities, system

calls, and library routines. The System V interfaces are included in the following tables. When referring to the System V version of a SunOS release 4.x interface, the string SysV is appended to the interface.

For complete information on all Solaris 2.6 interfaces, see the man Pages(2): System Calls.

Note - System Calls are functions. Functions in this appendix are identified by an empty set of parentheses immediately following the function name. When you see a second set of parentheses containing a number, this nomenclature identifies the associated man page section.

Examples

Below are sample table entries followed by an interpretation of the table entry.

SunOS release 4.x System Call	SunOS release 5.6 Status	Alternative Available and Notes	ABI	SVID	SVR4	BSD
`mctl`(2)	A	The `memcntl`(2) system call provides similar functionality.	A	A	A	S

The `mctl`() system call is not available in the ABI, SVID, SVR4, or the SunOS release 5.6 software. Any applications that use this system call should be rewritten to use the `memcntl`() call. A version of `mctl`() is available with the SunOS/BSD Compatibility package, but applications that use it will not be compatible with other SVR4 systems.

SunOS release 4.x System Call	SunOS release 5.6 Status	Alternative Available and Notes	ABI	SVID	SVR4	BSD
`getsockname`(2)	S#	The `errno` value `ENOBUFS` used by the SunOS release 4.x `getsockname`() system call has been changed to `ENOSR` in the SVR4 and SunOS release 5.6 version.	N	N	S#	N

The getsockname() system call is not defined in the ABI, or SVID. The getsockname() call in SunOS release 5.6 and SVR4 releases is the same as the one in the SunOS release 4.x software, except the SunOS release 5.6 software sets errno to ENOSR for the error condition that previously would have set errno to ENOBUFS.

System Calls

TABLE B–1 System Calls Reference Table

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
WEXITSTATUS(2)	C	The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in the SVR4 and SunOS release 5.6 versions.	N	N	S	S
WIFEXITED(2)	C	The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in the SVR4 and SunOS release 5.6 versions.	N	N	C	S
WIFSIGNALED(2)	C	The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in the SVR4 and SunOS release 5.6 versions.	N	N	C	S
WIFSTOPPED(2)	C	The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in the SVR4 and SunOS release 5.6 versions.	N	N	C	S
WSTOPSIG(2)	C	The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in the SVR4 and SunOS release 5.6 versions.	N	N	C	S
WTERMSIG(2)	C	The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in the SVR4 and SunOS release 5.6 versions.	N	N	C	S

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`_exit`(2V) — SysV	S		S	S	S	N
`accept`(2)	S	Now `accept`(3N).	N	N	S	N
`access` (2V) — SysV	S		S	S	S	N
`acct`(2)	C#	The following symbolic names are valid for the `acct` structure member `ac_flag` (defined in `<sys/acct.h>`) for SunOS release 4.x version, but not for SunOS release 5.6, ABI, SVID, and SVR4 versions: `ACOMPAT`, `ACORE`, `AXSIG`. Also, the accounting record format differs between SunOS release 4.x and SunOS release 5.6, ABI, SVID, and SVR4 versions.	C#	C#	C#	N
`adjtime`(2)	S		N	S	S	N
`async_daemon`(2)	N		N	N	N	N
`audit`(2)	N		N	N	N	N
`auditon` (2)	N		N	N	N	N
`auditsvc`(2)	N		N	N	N	N
`bind`(2)	S	Now `bind`(3N).	N	N	S	N
`brk`(2)	S		N	N	S	N
`chdir`(2V) — SysV	S		S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
chmod(2V) — SysV	C#	The following symbolic access modes (<sys/stat.h>) are supported by SunOS release 4.x chmod function but not by SunOS release 5.6, ABI, SVID, or SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent SunOS release 5.6, ABI, SVID, or SVR4 symbolic access modes S_IRUSR (00400), S_IWUSR (00200) and S_IXUSR (00100) have the same meanings.	C#	C#	C#	N
chown(2V)	C	In the SunOS release 4.x version, the *owner* and *group* arguments of chown() are of type int. In the SunOS release 5.6 software, ABI, SVID, and SVR4, *owner* is of type uid_t, and *group* is of type gid_t. In the SunOS release 4.x version, if the final component of *path* is a symbolic link, the ownership of the symbolic link was changed. In the SunOS release 5.6 version, chown() changes the ownership of the file or directory referred to by the symbolic link. Use lchown(2) to change the ownership of a symbolic link.	C	C	C	N
chown(2V) — SysV	S		S	S	S	N
chroot(2)	S		S	S	S	N
close(2V) — SysV	S		S	S	S	N
connect(2)	S#	Now connect(3N).	N	N	S#	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`creat` (2V)	C#	In the SunOS release 4.x software, the *mode* argument to `creat`() is of type int, while in SunOS release 5.6, ABI, SVID, and SVR4 versions, the *mode* argument is of type mode_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<fcntl.h>` while the SunOS release 4.x version does not. The following symbolic access modes (`<sys/stat.h>`) are supported by SunOS release 4.x version of `creat`(), but not by SunOS release 5.6, ABI, SVID, or SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent SunOS release 5.6, ABI, SVID, and SVR4 symbolic access modes S_IRUSR (00400), S_IWUSR (00200), and S_IXUSR (00100) do have the same definitions, are defined in SunOS release 4.x `<sys/stat.h>`, and thus should be used. The following `errno` flags are valid for the SunOS release 4.x version of this system call but are not valid in SunOS release 5.6, ABI, SVID, or SVR4 versions: ENXIO, EOPNOTSUPP.	C#	C#	C#	N
`creat`(2V) — SysV	C#	The following symbolic access modes (`<sys/stat.h>`) are supported by the SunOS release 4.x version of `creat`(), but not by SunOS release 5.6, ABI, SVID, or SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent SunOS release 5.6, ABI, SVID, and SVR4 symbolic access modes S_IRUSR (00400), S_IWUSR (00200), and S_IXUSR (00100) do have the same definitions, are defined in SunOS release 4.x `<sys/stat.h>`, and thus should be used. The following `errno` flags are valid for the SunOS release 4.x version of this system call but are not valid in SunOS release 5.6, ABI, SVID, or SVR4 versions: `ENXIO`, `EOPNOTSUPP`.	C#	C#	C#	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
dup(2V) — SysV	S		S	S	S	N
dup2(2V) — SysV	S	Now dup2(3C).	S	S	S	N
execve(2V) — SysV	S		S	S	S	N
fchdir(2V) — SysV	S		S	S	S	N
fchmod(2V) — SysV	C	The following symbolic access modes (<sys/stat.h>) are supported by the SunOS release 4.x version of fchmod(), but not by SunOS release 5.6, ABI, SVID, or SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent SunOS release 5.6, ABI, SVID, and SVR4 symbolic access modes S_IRUSR (00400), S_IWUSR (00200), and S_IXUSR (00100) do have the same definitions, are defined in SunOS release 4.x <sys/stat.h>, and thus should be used.	C	C	C	N
fchown(2)	S		S	S	S	N
fchroot(2)	S		N	N	N	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`fcntl`(2V) — SysV	C	In SunOS release 4.x, the following flags are valid for the `F_SETFL` command: –`O_APPEND`, –`O_SYNC`, and –`O_NDELAY`, and the –`FSYNC`, –`FNDELAY`, and –`FNBIO` flags defined in <`sys/file.h`>. SunOS release 5.6, ABI, SVID, and SVR4 versions support only the –`O_APPEND`, –`O_SYNC`, –`O_NDELAY`, and –`O_NONBLOCK` flags. Thus, –`O_SYNC` should be used in place of –`FSYNC`, and –`O_NONBLOCK` should be used in place of –`FNDELAY` and –`FNBIO`. –`O_NONBLOCK` should also be used in place of –`O_NDELAY`, which is being phased out. SunOS release 4.x `F_GETOWN` and `F_SETOWN` commands are not supported in SunOS release 5.6, ABI, SVID, or SVR4 versions.	C	C	C	N
`flock`(2)	N		N	N	N	S
`fork`(2V)	C	In the SunOS release 4.x software, `fork`() returns a value of type int. In SunOS release 5.6, ABI, SVID, and SVR4 versions, `fork`() returns a value of type pid_t. Also, SunOS release 5.6, ABI, SVID, or SVR4 versions include <`unistd.h`> and <`sys/types.h`> while the SunOS release 4.x version does not.	C	C	C	N
`fork`(2V) — SysV	S		S	S	S	N
`fpathconf` (2V) — SysV	S		S	S	S	N
`fstat`(2V) — SysV	S		S	S	S	N
`fstatfs`(2)	A	The `fstatvfs`(2) system call provides equivalent functionality.	A	A	A	S
`fsync`(2)	S		S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`ftruncate`(2)	S	Now `ftruncate`(3C).	N	N	S	N
`getauid`(2)	N		N	N	N	N
`getdents`(2)	S		N	N	S	N
`getdirentries`(2)	A	The `getdents`(2) system call provides equivalent functionality.	N	N	N	N
`getdomainname`(2)	A	The `sysinfo`(2) system call provides equivalent functionality.	N	N	N	N
`getdtablesize`(2)	A	Now `getdtablesize`(3C). The `getrlimit`(2) system call with the *resource* argument set to `RLIMIT_NOFILE` provides similar functionality.	A	A	A	S
`getegid`(2V)	C	In the SunOS release 4.x software, `getegid`() returns a value of type int. In SunOS release 5.6, ABI, SVID, and SVR4 versions, `getegid`() returns a value of type gid_t. Also, SunOS release 5.6, ABI, SVID, or SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`getegid`(2V) — SysV	S		S	S	S	N
`geteuid` (2V)	C	In the SunOS release 4.x software, `geteuid`() returns a value of type int. In SunOS release 5.6, ABI, SVID, and SVR4 versions, `geteuid`() returns a value of type uid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`geteuid`(2V) — SysV	S		S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`getgid` (2V)	C	In the SunOS release 4.x software, `getgid`() returns a value of type int. In SunOS release 5.6, ABI, SVID, and SVR4 versions, `getgid`() returns a value of type gid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`getgid`(2V) — SysV	S		S	S	S	N
`getgroups`(2V)	C	In the SunOS release 4.x software, the *gidset* argument to `getgroups`() is of type int, while in SunOS release 5.6, ABI, SVID, and SVR4 versions, the *grouplist* argument is of type gid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`getgroups`(2V) — SysV	S		S	S	S	N
`gethostid`(2)	A	Now `gethostid`(3C). The `sysinfo`(2) system call with the *command* argument set to `SI_HW_SERIAL` provides similar functionality.	N	N	N	S
`gethostname`(2)	A	Now `gethostname`(3C). The `sysinfo`(`SI_HOSTNAME`, *name*, *namelen*); routine provides similar functionality.	N	N	N	S
`getitimer`(2)	S		N	S	S	N
`getmsg`(2)	S		S	S	S	N
`getpagesize`(2)	A	Now `getpagesize`(3C). The `sysconf`(3C) routine provides similar functionality.	A	A	A	S

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`getpeername`(2)	S#	Now `getpeername`(3N). The following `errno` flag is valid for the SunOS release 4.x `getpeername`() system call but is not valid in the SVR4 and SunOS release 5.6 version: `ENOBUFS`.	N	N	S#	N
`getpgid`(2V)	S		S	S	S	N
`getpgrp`(2V)	C	The SunOS release 4.x version of `getpgrp`() has an argument pid, and `getpgrp`() returns the process group of the process indicated by pid. SunOS release 5.6, ABI, SVID, and SVR4 versions of `getpgrp`() do not accept an argument, and `getpgrp`() returns the process group ID of the calling process. Also, SunOS release 4.x `getpgrp`() returns a value of type int, while SunOS release 5.6, ABI, SVID, and SVR4 `getpgrp`() returns a value of type pid_t. SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`getpgrp` (2V) — SysV	S		S	S	S	N
`getpid`(2V)	C	In the SunOS release 4.x software, `getpid`() returns a value of type int. SunOS release 5.6, ABI, SVID, and SVR4, `getpid`() returns a value of type pid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`getpid`(2V) — SysV	S		S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`getppid` (2V)	C	In the SunOS release 4.x software, `getppid`() returns a value of type int. SunOS release 5.6, ABI, SVID, and SVR4, `getppid`() returns a value of type pid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`getppid`(2V) — SysV	S		S	S	S	N
`getpriority`(2)	A	Now `getpriority`(3C). The `priocntl`(2) system call provides similar functionality.	A	A	A	S
`getrlimit`(2)	C	In SunOS release 4.x, `RLIMIT_RSS` is a supported resource (the maximum size, in bytes, to which a process's resident set size may grow) which is not supported in SunOS release 5.6, ABI, SVID, and SVR4 versions. SunOS release 5.6, ABI, SVID, and SVR4 versions additionally support the `RLIMIT_AS` resource, the maximum amount of a process's address space that is defined (in bytes). Also, SunOS release 5.6, ABI, SVID, and SVR4 versions of `rlim_cur`(current soft limit) and `rlim_max` (hard limit) fields in the `rlimit` structure are `rlim_t` rather than int.	C	C	C	N
`getrusage`(2)	A	Now `getusage`(3C).	N	N	N	C
`getsockname`(2)	S#	The `errno` value `ENOBUFS` used by the SunOS release 4.x `getsockname`() system call has been changed to `ENOSR` in the SVR4 and SunOS release 5.6 version.	N	N	S#	N
`getsockopt`(2)	S	Now `getsockopt`(3N).	N	N	S	N
`gettimeofday`(2)	S	Now `gettimeofday`(3C).	N	S	S	S

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`getuid`(2V)	C	In the SunOS release 4.x software, `getuid`() returns a value of type int. SunOS release 5.6, ABI, SVID, and SVR4 `getuid`() returns a value of type uid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`getuid`(2V) — SysV	S		S	S	S	N
`ioctl`(2)	C	See "`ioctl()` Requests" on page 159	C	C	C	N
`kill`(2V)	C	In the SunOS release 4.x software, if a signal is sent to a group of processes (as with, if pid is 0 or negative), and if the process sending the signal is a member of that group, the signal is not sent to the sending process as well. In SunOS release 5.6, ABI, SVID, and SVR4 versions, the signal is sent to the sending process as well. In the SunOS release 4.x software, the pid argument is of type int, while in the SunOS release 5.6, ABI, SVID, and SVR4 versions, the pid argument is of type pid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`kill`(2V) — SysV	S		S	S	S	N
`killpg`(2)	A	Now `killpg`(3C). The `kill`(2) system call provides similar functionality. Replace `killpg`(`pgrp`, `sig`) with `kill`(`-pgrp`, `sig`).	A	A	A	S

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`link` (2V) — SysV	C	In the SunOS release 5.6, ABI, SVID, and SVR4 version of `link`(), if the last component of the first argument is a symbolic link, it will not be followed and a hard link will be made to the symbolic link.	C	C	C	N
`listen`(2)	S	Now `listen`(3N).	N	N	S	N
`lseek`(2V) — SysV	S		S	S	S	N
`lstat`(2V) — SysV	S		S	S	S	N
`mctl`(2)	A	The `memcntl`(2) system call provides similar functionality.	A	A	A	S
`mincore`(2)	C	In the SunOS release 4.x software, argument len is of type int, while in SVR4 and SunOS release 5.6 versions, argument len is of type size_t which is defined to be unsigned int. The SunOS release 5.6 version also requires inclusion of `<unistd.h>`.	N	N	C	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`mkdir`(2V)	C	In SunOS release 4.x, the mode argument is of type int, while in SunOS release 5.6, ABI, SVID, and SVR4, the mode argument is of type mode_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<sys/types.h>` and `<sys/stat.h>` while the SunOS release 4.x version does not. The following symbolic access modes (`<sys/stat.h>`) are supported by the SunOS release 4.x version of `mkdir`(), but not by SunOS release 5.6, ABI, SVID, and SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent SunOS release 5.6, ABI, SVID, and SVR4 symbolic access modes S_IRUSR (00400), S_IWUSR (00200), and S_IXUSR (00100) do have the same definitions, are defined in SunOS release 4.x `<sys/stat.h>`, and thus should be used.	C	C	C	N
`mkdir`(2V) — SysV	C	The following symbolic access modes (`<sys/stat.h>`) are supported by the SunOS release 4.x version of `mkdir`(), but not by SunOS release 5.6, ABI, SVID, and SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent SunOS release 5.6, ABI, SVID, and SVR4 symbolic access modes S_IRUSR (00400), S_IWUSR (00200), and S_IXUSR (00100) do have the same definitions, are defined in SunOS release 4.x `<sys/stat.h>`, and thus should be used.	C	C	C	N
`mkfifo`(2V) — SysV	S	Now `mkfifo`(3C).	S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
mknod(2V) — SysV	C	The mode argument to mknod() is of type int in the SunOS release 4.x software and of type mode_t in SunOS release 5.6, ABI, SVID, and SVR4 versions. The *dev* argument is of type int in the SunOS release 4.x software and of type dev_t in the SunOS release 5.6, ABI, SVID, or SVR4 versions. The following symbolic access modes (<sys/stat.h>) are supported by the SunOS release 4.x version of mknod(), but not by SunOS release 5.6, ABI, SVID, and SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent symbolic access modes S_IRUSR (00400), S_IWUSR (00200), and S_IXUSR (00100) do have the same definitions, are defined in SunOS release 4.x <sys/stat.h>, and thus should be used.	C	C	C	N
mmap(2)	C	In the SunOS release 4.x software, –mmap *flag* option value includes MAP_TYPE, defined in <sys/mman.h>, which is not defined in SunOS release 5.6, ABI, SVID, and SVR4 <sys/mman.h>.	C	C	C	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
mount(2)	C#	The SunOS release 4.x version of mount() and the SunOS release 5.6, or the ABI, SVID, or SVR4 version of mount() are incompatible in a number of respects. The first argument in the SunOS release 4.x version, *type*, is the file system type name, while in SunOS release 5.6, ABI, SVID, and SVR4 versions, the first argument, *fs*, is the name of the file system. In SunOS release 5.6, ABI, SVID, and SVR4 versions, the file system type name, *fstype*, is the fourth argument to mount(). The SunOS release 4.x version uses a single parameter (caddr_t *data*, the fourth argument) to pass type-specific arguments, while the SunOS release 5.6, ABI, SVID, and SVR4 version uses two parameters (five and six: const char **dataptr* and int *datalen*). Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include <sys/types.h> before <sys/mount.h> while the SunOS release 4.x version does not. The SunOS release 4.x version of <sys/mount.h> defines symbolic constants for the mount() *flags* argument (M_NEWTYPE, M_RDONLY, M_NOSUID, M_NEWTYPE, M_GRPID, M_REMOUNT , M_NOSUB, M_MULTI) that are not defined in SunOS release 5.6, or the ABI, SVID, or SVR4 <sys/mount.h>. Instead, replace M_RDONLY with MS_RDONLY, M_NOSUID with MS_NOSUID, and M_REMOUNT with MS_REMOUNT. The M_NEWTYPE flag is specific to the SunOS release 4.x version of mount() and no replacement is required for SunOS release 5.6, ABI, SVID, or SVR4 versions. The functionality of the following flags, defined in <sys/mount.h>, is not supported by the SunOS release 5.6, or the ABI, SVID, or SVR4 versions: M_NOSUB, M_GRPID, M_MULTI. SunOS release 4.x mount() uses the following errno values, which are not returned by the SunOS release 5.6, or the ABI, SVID, or SVR4 version: ENODEV, EACCES, EMFILE, ENOMEM.	C#	C#	C#	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
mprotect(2)	S		S	S	S	N
msgctl(2)	S		S	S	S	N
msgget(2)	S		S	S	S	N
msgrcv(2)	S		S	S	S	N
msgsnd(2)	S		S	S	S	N
msync(2)	S		S#	S#	S	N
munmap(2)	S		S	S	S	N
nfssvc(2)	A	This interface is replaced in SunOS release 5.6 by the nfssys(NFS_SVC,...); routine.	N	N	N	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
open(2V)	C#	The *mode* argument to open() is of type int in the SunOS release 4.x software and of type mode_t in SunOS release 5.6, ABI, SVID, and SVR4 versions. In the SunOS release 4.x software, if the *path* argument is an empty string, the kernel maps this empty pathname to '.', the current directory. In SunOS release 5.6, ABI, SVID, and SVR4 versions, if *path* points to an empty string an error results. In the SunOS release 4.x software, if the O_NDELAY or O_NONBLOCK flag is set on a call to open, only the open() call itself is effected. In SunOS release 5.6, ABI, SVID, and SVR4 versions, if the O_NDELAY or O_NONBLOCK flag is set on a call to open(), the corresponding flag is set for that file descriptor and subsequent reads and writes to that descriptor will not block. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include <sys/types.h> and <sys/stat.h> while the SunOS release 4.x version does not. The following errno value is valid for the SunOS 4.1 version of this system call but is not returned in SunOS release 5.6, ABI, SVID, and SVR4 versions: EOPNOTSUPP.	C#	C#	C#	N
open(2V) — SysV	S#	The following errno value is valid for the SunOS release 4.x version of this system call but is not returned in SunOS release 5.6, ABI, SVID, and SVR4 versions: EOPNOTSUPP.	S#	S#	S#	N
pathconf(2V) — SysV	S		S	S	S	N
pipe(2V) — SysV	S		S	S	S	N
poll(2)	S		S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`profil`(2)	S		S	S	S	N
`ptrace`(2)	C#	The optional *addr2* argument to the SunOS release 4.x software `ptrace`() system call is not supported by the SunOS release 5.6 routine. The request argument to `ptrace`() is of type enum `ptracereq` in the SunOS release 4.x software and of type int in the SunOS release 5.6 version. The *pid* argument to `ptrace`() is of type int in the SunOS release 4.x software and of type pid_t in theSunOS release 5.6 version. Also, the SunOS release 5.6 version includes `<sys/types.h>` while the SunOS release 4.x version includes `<signal.h>`, `<sys/ptrace.h>`, and `<sys/wait.h>`. The following `errno` flag is valid for the SunOS release 4.x version of this system call, but is not valid in the SunOS release 5.6 version: `EPERM`. See "`ptrace()` Request Values" on page 162for information on valid *request* values.	C#	C#	C#	N
`putmsg`(2)	S		S	S	S	N
`quotactl`(2)	A	The `Q_QUOTACTL ioctl`system call provides similar functionality.	A	A	A	N
`read`(2V)	C#	The following `errno` flags are valid for the SunOS release 4.x version of this system call but are not valid in SunOS release 5.6, ABI, SVID, and SVR4 versions: `EISDIR`, `EWOULDBLOCK`.	C#	C#	C#	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`read`(2V) — SysV	C#	The *nbyte* argument to `read`() is of type int in the SunOS release 4.x software and of type unsigned in the SunOS release 5.6 version. The SunOS release 5.6 `read`() system call does not support BSD 4.2 style non-blocking I/O (with the `FIONBIO` `ioctl`() request or a call to `fcntl`(2V) using the `FNDELAY` flag from `<sys/file.h>` or the `O_NDELAY` flag from `<fcntl.h>` in the 4.2BSD environment) as does the SunOS release 4.x `read`() routine. The following `errno` flags are valid for the SunOS release 4.x version of this system call, but are not valid in the SunOS release 5.6 version: `EISDIR`, `EWOULDBLOCK`.	C#	C#	C#	N
`readlink`(2)	S		S	S	S	N
`readv`(2V)	C#	The following `errno` flags are valid for the SunOS release 4.x version of this system call but are not valid in SunOS release 5.6, ABI, SVID, and SVR4 versions: `EISDIR`, `EWOULDBLOCK`.	C#	C#	C#	N
`readv`(2V) — SysV	C#	SunOS release 4.x and SunOS release 5.6, or the SVID or SVR4 `iovec` structures (defined in `<sys/uio.h>`) differ slightly. The SunOS release 4.x `iovec` `iov_len` field is defined as integer, while SunOS release 5.6 or the SVID or SVR4 `iov_len` is defined as unsigned. SunOS release 5.6 or the SVID or SVR4 `readv`() system call does not support BSD 4.2 style non- blocking I/O.	C#	C#	C#	N
`reboot`(2)	A	Now `reboot`(3C). The `uadmin`(2) system call provides similar functionality.	N	N	N	S
`recv`(2)	S	Now `recv`(3N).	N	N	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`recvfrom`(2)	S	Now `recvfrom`(3N).	N	N	S	N
`recvmsg`(2)	S	Now `recvmsg`(3N).	N	N	S	N
`rename`(2V) — SysV	S#	The SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` while the SunOS release 4.x version does not. The following `errno` flag is valid for the SunOS release 4.x version of this system call but is not valid in SunOS release 5.6, ABI, SVID, and SVR4 versions: `ENOTEMPTY` . SunOS release 5.6, ABI, SVID, and SVR4 versions set `errno` to flag `EEXIST` instead.	S#	S#	S#	N
`rmdir`(2V) — SysV	S#	The SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` while the SunOS release 4.x version does not. The following `errno` flag is valid for the SunOS release 4.x version of this system call but is not valid in SunOS release 5.6, ABI, SVID, and SVR4 versions: `ENOTEMPTY`. SunOS release 5.6, ABI, SVID, and SVR4 versions set `errno` to flag `EEXIST` instead.	S#	S#	S#	N
`sbrk`(2)	S		N	N	S	N
`select`(2)	S	Now `select`(3C).	N	N	S	N
`semctl`(2)	S		S	S	S	N
`semget`(2)	S		S	S	S	N
`semop`(2)	S		S	S	S	N
`send`(2)	S#	Now `send`(3N). The following `errno` flag is valid for SunOS release 4.x `send`(2) system calls but is not valid in the SVR4 and SunOS release 5.6: `ENOBUFS`.	N	N	S#	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`sendmsg`(2)	S#	Now `sendmsg`(3N). The following `errno` flag is valid for SunOS release 4.x `sendmsg`(2) system calls but is not valid in the SVR4 and SunOS release 5.6: `ENOBUFS`.	N	N	S#	N
`sendto`(2)	S#	Now `sendto`(3N). The following `errno` flag is valid for SunOS release 4.x `sendto`(2) system calls but is not valid in the SVR4 and SunOS release 5.6: `ENOBUFS`.	N	N	S#	N
`setaudit`(2)	N		N	N	N	N
`setauid`(2)	N		N	N	N	N
`setdomainname`(2)	A	The `sysinfo`(2) system call provides similar functionality.	N	N	N	N
`setgroups`(2V)	C	In the SunOS release 4.x software, the gidset argument is of type int, while in SunOS release 5.6, ABI, SVID, and SVR4 versions, the grouplist argument is of type gid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include `<unistd.h>` and `<sys/types.h>` while the SunOS release 4.x version does not.	C	C	C	N
`setgroups`(2V) — SysV	S		S	S	S	N
`sethostname`(2)	A	Now `sethostname`(3C). The `sysinfo`(2) system call with the command argument set to `SI_SET_HOSTNAME` provides similar functionality.	N	N	N	S
`setitimer`(2)	S		N	S	S	N
`setpgid`(2V) — SysV	S		S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
setpgrp(2V)	C	The SunOS release 4.x version of setpgrp() has arguments pid and pgrp, and setpgrp() sets the process group to pgrp of the process indicated by pid. The SunOS release 5.6 version of setpgrp() does not accept an argument, and setpgrp() also creates a new session. However, if pgrp is zero and pid refers to the calling process, then SunOS release 4.x setpgrp() call is identical to a SunOS release 5.6 setpgrp() call with no arguments. Also, SunOS release 4.x setpgrp() returns a value of type int, while SunOS release 5.6, setpgrp() returns a value of type pid_t. The SunOS release 5.6 version includes <unistd.h> and <sys/types.h> while the SunOS release 4.x version does not. The following errno flags are valid for SunOS release 4.x setpgrp() system call but are not valid in SunOS release 5.6, ABI, SVID, and SVR4 versions: EACCES, EINVAL, ESRCH.	C#	C#	C#	N
setpgrp(2V) — SysV	S	The following errno flags are valid for SunOS release 4.x setpgrp(2V) system call but is not valid in SunOS release 5.6, ABI, SVID, and SVR4 versions: EACCES, EINVAL, ESRCH.	S	S	S	N
setpriority(2)	A	Now setpriority(3C). The priocntl(2) system call provides similar functionality.	A	A	A	S
setregid(2)	S	Now setregid(3C).	N	N	N	C
setreuid(2)	S	Now setreuid(3C).	N	N	N	C
setrlimit(2)	C	Now setrlimit(3C).	C	C	C	N
setsid(2V) — SysV	S		S	S	S	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
setsockopt(2)	S	Now setsockopt(3N).	N	N	S	N
settimeofday(2)	S	Now settimeofday(3C).	N	S	S	S
setuseraudit(2)	N		N	N	N	N
sgetl(2)	S	Now xdr_simple(3N).	N	S	S	N
shmat(2)	S		S	S	S	N
shmctl(2)	S		S	S	S	N
shmdt(2)	S		S	S	S	N
shmget(2)	S		S	S	S	N
shutdown(2)	S	Now shutdown(3N).	N	N	S	N
sigaction(2)	C	There is a flag in the Solaris 2.6 version, SA_RESTART, that allows a function that is interrupted by the execution of this signal's handler to be transparently restarted by the system.	N	C	C	S
sigblock(2)	A	The sigprocmask(2) system call with the how argument set to SIG_BLOCK provides similar functionality.	A	A	A	S
sigmask(2)	A	The sigsetops(3C) routines provide similar functionality.	A	A	A	S
sigpause(2V) — SysV	S	The SunOS release 4.x sigpause() system call assigns its argument (sigmask)to the set of masked signals while the ABI and SVID versions of sigpause remove its argument (sig) from the calling process's signal mask. The SVR4 and SunOS release 5.6 sigpause() is compatible with SunOS release 4.x sigpause(2).	C	C	S	S

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
sigpending(2V) — SysV	S		S	S	S	N
sigprocmask(2V) — SysV	S		S	S	S	N
sigsetmask(2)	A	The sigprocmask(2) routine with the how argument set to SIG_SETMASK provides similar functionality.	A	A	A	S
sigstack(2)	A	Now sigstack(3C). The sigaltstack(2) system call provides similar functionality.	A	A	A	S
sigsuspend(2V) — SysV	S		S	S	S	N
sigvec(2)	A	The sigaction(2) system call provides similar functionality.	A	A	A	S
socket(2)	C#	Now socket(3N). The SunOS release 4.x PF_IMPIPNK is a supported *domain* , while in SVR4 and SunOS release 5.6 software PF_IMPIPNK is not supported. The following errno flags are valid for the SunOS release 4.x socket() system call but are not valid in the SVR4 and SunOS release 5.6 version: ENOBUFS, EPROTOTYPE.	N	N	C#	N
socketpair(2)	S	Now socketpair(3N).	N	N	S	N
sputl(2)	S	Now xdr_simple(3N).	N	S	S	N
stat(2V) — SysV	S		S	S	S	N
statfs(2)	A	The statvfs(2) system call provides similar functionality.	A	A	A	N
swapon(2)	A	The swapctl(2) system call provides similar functionality.	N	N	N	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`symlink`(2)	S		S	S	S	N
`sync`(2)	S		S	S	S	N
`syscall`(2)	N		N	N	N	S
`sysconf`(2V) — SysV	S	Now `sysconf`(3C).	S	S	S	N
`tell`(2V) — SysV	S		S	N	S	N
`truncate`(2)	S	Now `truncate`(3C).	N	N	S	N
`umask`(2V) — SysV	C	The following symbolic access modes (`<sys/stat.h>`) are supported by the SunOS release 4.x version of `umask`(), but not by SunOS release 5.6, ABI, SVID, and SVR4 versions: S_IREAD (00400), S_IWRITE (00200), S_IEXEC (00100). However, the equivalent SunOS release 5.6, ABI, SVID, and SVR4 symbolic access modes, S_IRUSR (00400), S_IWUSR (00200), and S_IXUSR (00100) do have the same definitions, are defined in SunOS release 4.x `<sys/stat.h>`, and thus should be used.	C	C	C	N
`umount`(2V) — SysV	S		S	S	S	N
`uname`(2V) — SysV	S		S	S	S	N
`unlink`(2V) — SysV	S		S	S	S	N
`unmount`(2)	A	The `umount`(2) system call provides similar functionality.	A	A	A	N
`ustat`(2)	S		S	S	S	N
`utimes`(2)	S		N	N	N	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
vadvise(2)	N		N	N	N	N
vfork(2)	S		N	N	S	N
vhangup(2)	S		N	N	N	N
wait(2V)	C	In SunOS release 4.x, wait() returns a value of type int. In SunOS release 5.6, ABI, SVID, and SVR4, wait() returns a value of type pid_t. Also, SunOS release 5.6, ABI, SVID, and SVR4 versions include <sys/types.h> while the SunOS release 4.x version does not. The *union wait*, supported in SunOS release 4.x for backwards compatibility with previous SunOS releases, is not supported in SunOS release 5.6, ABI, SVID, and SVR4 versions. In SunOS release 4.x, wait() is automatically restarted when a process receives a signal while awaiting termination, unless the SV_INTERRUPT bit is set in the flags for that signal. In SunOS release 5.6, ABI, SVID, and SVR4, the wait() system call returns prematurely if a signal is received.	C	C	C	N
wait(2V) — SysV	C	The *union wait*, supported in the SunOS release 4.x softwarefor backwards compatibility, is not supported in SunOS release 5.6, ABI, SVID, and SVR4 versions. The SunOS release 4.x,wait(2V) is automatically restarted when a process receives a signal while awaiting termination unless the SV_INTERRUPT bit is set in the flags for that signal. In the SunOS release 5.6, ABI, SVID, and SVR4 versions, the wait(2) function will return prematurely if a signal is received.	C	C	C	N
wait3(2V)	A	Now wait3(3C). The wait(2) and waitpid(2) system calls provide similar functionality.	A	A	A	S

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`wait4`(2V)	A	Now `wait4`(3C). The `wait`(2) and `waitpid`(2) system calls provide similar functionality.	A	A	A	S
`waitpid`(2V)	C	SunOS release 4.x `waitpid`() returns a value of type int. In the SunOS release 5.6, ABI, SVID, and SVR4 versions, `waitpid`() returns a value of type pid_t. The *pid* argument to `waitpid`() is of type int in the SunOS release 4.x software and of type pid_t in the SunOS release 5.6, ABI, SVID, and SVR4 versions. Also, the SunOS release 5.6, ABI, SVID, and SVR4 versions include `<sys/types.h>` while the SunOS release 4.x version does not. The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in SunOS release 5.6, ABI, SVID, and SVR4 versions. The SunOS release 4.x `waitpid`() is automatically restarted when a process receives a signal while awaiting termination unless the `SV_INTERRUPT` bit is set in the flags for that signal. In SunOS release 5.6, ABI, SVID, and SVR4 versions, the `waitpid`() system call returns prematurely if a signal is received.	C	C	C	N
`waitpid`(2V) — SysV	C	The *union wait*, supported in the SunOS release 4.x software for backwards compatibility, is not supported in SunOS release 5.6, ABI, SVID, and SVR4 versions. The SunOS release 4.x `waitpid`(2V) is automatically restarted when a process receives a signal while awaiting termination unless the `SV_INTERRUPT` bit is set in the flags for that signal. In SunOS release 5.6, ABI, SVID, and SVR4 versions, the `waitpid`(2) function will return prematurely if a signal is received.	C	C	C	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`write`(2V)	C#	In the SunOS release 4.x software, if the object that the descriptor refers to is marked for non-blocking I/O, using the `FIONBIO` request to `ioctl`(), or using `fcntl`() to set the `FNDELAY` or `O_NDELAY` flag, `write`() returns `-1` and sets `errno` to `EWOULDBLOCK`. In the SunOS release 5.6 software, on a `write`() to a regular file, if `O_NDELAY` or `O_NONBLOCK` is set, `write`() returns `-1` and sets `errno` to `EAGAIN`. On `write`() requests to a pipe or `FIFO` with `O_NONBLOCK` or `O_NDELAY` set, `write`() does not block the process. If some data can be written without blocking the process, `write`() writes what it can and returns the number of bytes written; otherwise, when `O_NONBLOCK` is set, it returns `-1` and sets `errno` to `EAGAIN` and when `O_NDELAY` is set, it returns `0`. With `O_NDELAY` set, `write`() requests for {`PIPE_BUF`} or fewer bytes either succeed completely and return *nbytes*, or return `0`. A `write`() request for greater than {`PIPE_BUF`} bytes either transfers what it can and returns the number of bytes written, or transfers no data and returns `0`. Also, if a request is greater than {`PIPE_BUF`} bytes and all data previously written to the pipe has been read, `write`() transfers at least {`PIPE_BUF`} bytes. The SunOS release 5.6 `write`() routine does not support 4.2 BSD style non-blocking I/O. The following `errno` flag is valid for the SunOS release 4.x version of this system call but is not valid in the SunOS release 5.6 version: `EWOULDBLOCK`.	C#	C#	C#	N

TABLE B–1 System Calls Reference Table *(continued)*

SunOS Release 4.x System Call	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`write`(2V) — SysV	C#	The SunOS release 5.6, ABI, SVID, and SVR4 versions of `write`() does not support 4.2 BSD style non-blocking I/O. The following `errno` flag is valid for the SunOS release 4.x version of this system call but is not valid in SunOS release 5.6, ABI, SVID, and SVR4 versions: `EWOULDBLOCK`.	C#	C#	C#	N
`writev`(2V)	C#	SunOS release 5.6, ABI, SVID, and SVR4 versions of `writev`() does not support 4.2 BSD style non-blocking I/O. The following `errno` flag is valid for the SunOS release 4.x version of this system call but is not valid in the SunOS release 5.6, or the ABI, SVID, or SVR4 version: `EWOULDBLOCK`.	C#	C#	C#	N

APPENDIX **C**

Library Routines Reference Table

This appendix contains the Library Routine reference table. This table list all SunOS release 4.x library routines and shows their status in the Solaris 2.6, the ABI, the SVID, SVR4, and the SunOS/BSD Source Compatibility Package environments.

Using the Reference Table

- If an interface is listed as “changed” (C), a brief description of differences between the SunOS release 4.x and the Solaris 2.6 routine is provided.
- If an interface is listed as “the same” (S), the Solaris 2.6 interface supports all features of the SunOS release 4.x interface. In some cases the interface has been enhanced, but can be considered a complete superset of the SunOS release 4.x interface.
- If an interface has an “alternative” (A), check the Notes section for its replacement.
- If an interface is listed as “not available” (N), check the Notes section for information about its replacement. Routines listed in the SunOS release 5.6 column replace the SunOS release 4.x interface.

SunOS release 4.x offers a System V Software installation option that provides System V compatible versions of many routines. The System V interfaces are included in the following tables. When referring to the System V version of a SunOS release 4.x interface, the string ‘SysV’ is appended to the interface.

Routines that exist in both `/usr/lib` and `/usr/5lib` have two table entries. The first documents the `/usr/lib` routine, and the second entry documents the `/usr/5lib` routine.

For complete information on all Solaris 2.6 interfaces, see the man Pages(3): Library Routines.

Examples

Below are sample table entries followed by an interpretation of the entry..

SunOS release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	ABI	SVID	SVR4	BSD
`clntraw_create`(3N)	S	This routine is still available, but is superseded by `clnt_raw_create`(3N) in the SunOS release 5.6 and SVR4 versions.	A	A	S	N

The `clntraw_create`() routine exists in this release, but it also has a replacement routine: `clnt_raw_create`(). Applications that use `clntraw_create`() will continue to work in this release and on other SVR4–compliant systems, but these applications should be updated to use `clnt_raw_create`(). `clntraw_create`() is considered obsolete, and may not be available in future releases. If you want your application to be ABI— or SVID— compliant, use `clnt_raw_create`().

SunOS release 4.x Command	SunOS release 5.6 Status	Alternative Available and Notes	ABI	SVID	SVR4	BSD
`putpwent`(3)	S		S	S	S	N

The SunOS release 4.x `putpwent`() routine and the SunOS release 5.6 routine are the same. Applications that use this routine will behave as they did in the SunOS release 4.x software.

Library Routines

TABLE C–1 Library Routines Reference Table

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
_crypt(3)	C	The crypt(3C) routine provides similar functionality.	N	A	A	N
_longjmp(3)	C	Now _longjmp(3C). The siglongjmp(3) routine provides similar functionality.	A	A	A	S
_setjmp(3)	C	Now setjmp(3C). The sigsetjmp(3) routine provides the same functionality when the *savemask* argument is zero. This saves the calling process's registers and stack environment, but not its *signalmask*.	A	A	A	S
_tolower(3V) – SysV	S		S	S	S	N
_toupper(3V) – SysV	S		S	S	S	N
CHECK(3L)	N		N	N	N	N
HUGE(3M)	C	In the SunOS release 4.x software, HUGE is defined in <math.h> as infinity(3M), which produces IEEE Infinity. In SunOS release 5.6, SVID, or SVR4 versions, HUGE is defined in <math.h> as a machine-dependent constant.	N	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
HUGE_VAL(3M)	C	In the SunOS release 4.x software, HUGE_VAL is defined in <math.h> as infinity(3M), which produces IEEE Infinity. In the SunOS release 5.6, SVID, or SVR4 versions, HUGE_VAL is defined in <math.h> as a machine-dependent constant.	N	C	C	N
MONITOR(3L)	N		N	N	N	N
MSG_RECVALL(3L)	N		N	N	N	N
SAMECV(3L)	N		N	N	N	N
SAMEMON(3L)	N		N	N	N	N
SAMETHREAD(3L)	N		N	N	N	N
STKTOP(3L)	N		N	N	N	N
a64l(3)	S		S	S	S	N
abort(3)	S		S	S	S	N
abs(3)	S		S	S	S	N
acos(3M)	C	In the SunOS release 4.x software, if the absolute value of the argument of acos() is greater than 1, NaN is returned with an EDOM error and a DOMAIN math err. The SunOS release 5.6, the SVID, or SVR4 versions return zero with an EDOM error and a DOMAIN math err.	N	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
acosh(3M)	S		N	S	S	N
addch(3V) – SysV	S		N	S	S	N
addexportent(3)	A	The /etc/dfs/sharetab file replaces /etc/exports. Refer to the share(1M), unshare(1M), and sharetab(4) man pages for more information.	N	N	N	N
addmntent(3)	A	The putmntent() routine provides similar functionality. Refer to getmntent(3C).	N	N	N	N
addstr(3V) – SysV	S		N	S	S	N
agt_create(3L)	N		N	N	N	N
agt_enumerate(3L)	N		N	N	N	N
agt_trap(3L)	N		N	N	N	N
aint(3M)	N		N	N	N	N
aiocancel(3)	S		N	N	N	N
aioread(3)	S		N	N	N	N
aiowait(3)	S		N	N	N	N
aiowrite(3)	S		N	N	N	N
alarm(3V)	S		S	S	S	N
alloca(3)	S		N	N	N	N
alphasort(3)	N		N	N	N	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
anint(3M)	N		N	N	N	N
annuity(3M)	N		N	N	N	N
arc(3X)	S		N	N	N	N
asctime(3V)	C	See ctime(3V).	C	C	C	N
asin(3M)	C	In the SunOS release 4.x software, if the absolute value of the argument of asin() is greater than 1, NaN is returned with an EDOM error and a DOMAIN math err. The SunOS release 5.6, SVID, or SVR4 versions return zero with an EDOM error and a DOMAIN math err.	N	C	C	N
asinh(3M)	S		N	S	S	N
assert(3V)	C	The SunOS 4.x version of assert() calls exit(3C) while the SunOS release 5.6, ABI, SVID, or SVR4 versions call abort(3C).	C	C	C	N
assert(3V) – SysV	S		S	S	S	N
atan(3M)	S		N	S	S	N
atan2(3M)	C	The SunOS release 5.6, the SVID, or SVR4 version of atan2(0.0,0.0) returns zero and sets errno to EDOM. In the SunOS 4.x version, the same call might return +/-0.0 or +/- PI in conformance with 4.3BSD in the spirit of ANSI/ IEEEStd754- 1985.	N	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
atanh(3M)	S		N	S	S	N
atof(3)	C	See strtod(3).	C	C	C	N
atoi(3)	S		S	S	S	N
atol(3)	S		S	S	S	N
attroff(3V) – SysV	S		N	S	S	N
attron(3V) – SysV	S		N	S	S	N
attrset(3V) – SysV	S		N	S	S	N
audit_args(3)	N		N	N	N	N
audit_text(3)	N		N	N	N	N
authdes_create(3N)	A	This routine is still available, but is superseded by authdes_seccreate(3N) in SunOS release 5.6, or the ABI, SVID, or SVR4.	A	A	A	N
authdes_getucred(3N)	S		S	S	S	N
auth_destroy(3N)	S		S	S	S	N
authnone_create(3N)	S		S	S	S	N
authunix_create(3N)	A	This routine is still available, but is superseded by authsys_seccreate(3N).	A	A	A	N
authunix_create _default(3N)	A	This routine is still available, but is superseded by authsys_create_default(3N).	A	A	A	N
baudrate(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
bcmp(3)	S	Now bcmp(3C).	A	A	A	S
bcopy(3)	S	Now bcopy(3C).	A	A	A	S
beep(3V) – SysV	S		N	S	S	N
bindresvport(3N)	S		N	N	S	N
bootparam(3R)	S		N	N	N	N
box(3V)	C	The SunOS 4.x version of box() is a function while the SunOS release 5.6, or the SVID or SVR4 version of box(win,verch,horch) is a macro that calls wborder(win, verch, verch, horch, horch, 0, 0, 0, 0). Default values defined in <curses.h> in the SunOS release 5.6, or the SVID or SVR4 environment—ACS_ULCORNER, ACS_URCORNER, ACS_BLCORNER, and ACS_BRCORNER, are used to draw the upper left and right and bottom left and right corners of the box around the window. Also, the type of arguments verch and horch in the SunOS 4.x software is char, while in SunOS release 5.6, or the SVID or SVR4 versions they are ch type.	N	C	C	S
box(3V) – SysV	S		N	S	S	N
bsearch(3)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
byteorder(3N)	S		N	N	S	N
bzero(3)	S	Now bzero(3C).	A	A	A	S
calloc(3)	S		S	S	S	N
callrpc(3N)	A	This routine is still available, but is superseded by rpc_call(3N).	N	N	S	N
catclose(3C)	S		S	S	S	N
catgetmsg(3C)	A	In the SunOS release 5.6, or the ABI, SVID, or SVR4 environment, use catgets(3C) followed by strncpy(3) to copy the catalog message from the internal buffer area to a program buffer.	A	A	A	N
catgets(3C)	S		S	S	S	N
catopen(3C)	S		S	S	S	N
cbc_crypt(3)	S		N	N	N	N
cbreak(3V) – SysV	S		N	S	S	S
cbrt(3M)	S		N	S	S	N
ceil(3M)	S		N	S	S	N
cfgetispeed(3V)	S		S	S	S	N
cfgetospeed(3V)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`cfree`(3)	A	This routine is replaced by `void free(void*ptr)`. Refer to `malloc`(3C)) man page.	A	A	A	N
`cfsetispeed`(3V)	S		S	S	S	N
`cfsetospeed`(3V)	S		S	S	S	N
`circle`(3X)	S		N	N	N	N
`clear`(3V) – SysV	S		N	S	S	S
`clearerr`(3V) – SysV	S		S	S	S	N
`clearok`(3V) – SysV	S		N	S	S	S
`clnt_broadcast`(3N)	S	This routine is still available, but is superseded by `rpc_broadcast`(3N).	A	A	A	N
`clnt_call`(3N)	S		S	S	S	N
`clnt_control`(3N)	S		S	S	S	N
`clnt_create`(3N)	S		S	S	S	N
`clnt_destroy`(3N)	S		S	S	S	N
`clnt_freeres`(3N)	S		S	S	S	N
`clnt_geterr`(3N)	S		S	S	S	N
`clnt_pcreateerror`(3N)	S		S	S	S	N
`clnt_perrno`(3N)	S		S	S	S	N
`clnt_perror`(3N)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`clnt_spcreateerror`(3N)	S		S	S	S	N
`clnt_sperrno`(3N)	S		S	S	S	N
`clnt_sperror`(3N)	S		S	S	S	N
`clntraw_create`(3N)	S	This routine is still available, but is superseded by `clnt_raw_create`(3N) in the SunOS release 5.6 and SVR4 software.	A	A	S	N
`clnttcp_create`(3N)	S	This routine is still available, but is superseded by the `clnt_create`(3N), `clnt_tli_create`(3N), and `clnt_vc_create`(3N) routines in SunOS release 5.6, or the ABI, SVID, or SVR4.	N	N	S	S
`clntudp_bufcreate`(3N)	S	This routine is still available, but is superceded by `clnt_create`(3N), `clnt_tli_create`(3N), and `clnt_dg_create`(3N) routines.	N	N	S	
`clntudp_create`(3N)	S	This routine is still available, but is superseded by the `clnt_create`(3N), `clnt_tli_create`(3N), and `clnt_dg_create`(3N) routines .	N	N	S	S
`clock`(3C)	S		S	S	S	N
`closedir`(3V)	S		S	S	S	N
`closedir`(3V) – SysV	S		S	S	S	N
`closelog`(3)	S		N	N	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
closepl(3X)	S		N	N	N	N
clrtobot(3V) – SysV	S		N	S	S	S
clrtoeol(3V) – SysV	S		N	S	S	S
compound(3M)	N		N	N	N	N
cont(3X)	S		N	N	N	N
copysign(3M)	N		N	N	S	N
copywin(3V) – SysV	S		N	S	S	N
cos(3M)	C	For arguments that are much lower than zero, the SunOS release 5.6, or the SVID or SVR4 version of these routines returns zero because of the loss of significance. In this case, a message indicating TLOSS (see matherr(3M)) appears on the standard output. For cases of partial loss of significance, a PLOSS error is generated, but no error is printed. In both cases, errno is set to ERANGE. In the SunOS 4.x version, an argument reduction takes place for values exceeding PI/4 in magnitude. The reduction could happen in software or hardware. The variable *fp_pi* defined in <math.h> allows changing of the precision at runtime. The error exceptions occur in the IEEE 754 spirit for both versions.	N	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`cosh`(3M)	S		N	S	S	N
`crmode`(3X)	A	This routine is replaced by `cbreak`() . See `curs_inopts`(3X).	A	A	A	N
`crypt`(3)	C	In the SunOS 4.x version, the first two characters of the *salt* argument are interpreted and checked for (##and#$) as special cases in order to call additional authentication routines (`pwdauth`(3) and `grpauth`(3) respectively). If these functions return `TRUE`, the *salt* is returned from `crypt`. Otherwise, `NULL` is returned. In the SunOS release 5.6, or the SVID or SVR4 version, this functionality is not supported.	N	C	C	N
`ctermid`(3V) – SysV	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`ctime`(3V)	C	The SunOS 4.x `tm` structure contains two fields not present in the SunOS release 5.6, or the ABI, SVID, or SVR4 `tm` structure: `tm_zone` and `tm_gmtoff`. Instead, the SunOS release 5.6, or the ABI, SVID, or SVR4 version uses the external variable *timezone* to contain the difference (in seconds) between GMT and local standard time, and the external variable *daylight* to indicate if daylight savings should be applied. Additionally, the SunOS release 5.6, or the ABI, SVID, or SVR4 version uses an external variable *tzname* to store standard and summer time zone names. These external variables (*timezone*, *daylight*, and *tzname*) are supported by the SunOS 4.x System V `ctime`(3V) library routines. The use of the environmental variable `TZ` differs between the SunOS 4.x and the SunOS release 5.6, or the ABI, SVID, or SVR4 versions. In the SunOS release 4.x version, `TZ` contains the pathname of *tzfile-format* file from which to read the time conversion information. In the SunOS release 5.6, or the ABI, SVID, or SVR4 versions, TZ itself contains the time conversion information (of different format than the *tzfile-format*).	C	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
curs_set(3V) – SysV	S		N	S	S	N
cuserid(3V)	S		S	S	S	N
cv_broadcast(3L)	N		N	N	N	N
cv_create(3L)	N		N	N	N	N
cv_destroy(3L)	N		N	N	N	N
cv_enumerate(3L)	N		N	N	N	N
cv_notify(3L)	N		N	N	N	N
cv_send(3L)	N		N	N	N	N
cv_wait(3L)	N		N	N	N	N
cv_waiters(3L)	N		N	N	N	N
dbm_clearerr(3)	S		N	N	N	N
dbm_close(3X)	S	The dbm_close(3) routine provides similar functionality.	N	N	N	N
dbm_delete(3)	S	The dbm_delete(3) routine provides similar functionality.	N	N	N	N
dbm_error(3)	S		N	N	N	N
dbm_fetch(3)	S		N	N	N	N
dbm_firstkey(3)	S		N	N	N	N
dbm_nextkey(3)	S		N	N	N	N
dbm_open(3)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
dbm_store(3)	S		N	N	N	N
dbmclose(3X)	N		N	N	N	S
dbminit(3X)	S		N	N	N	S
decimal_to_double(3)	S		N	N	N	N
decimal_to_extended(3)	S		N	N	N	N
decimal_to_floating(3)	S		N	N	N	N
decimal_to_single(3)	S		N	N	N	N
def_prog_mode(3V) – SysV	S		N	S	S	N
def_shell_mode(3V) – SysV	S		N	S	S	N
del_curterm(3V) – SysV	S		N	S	S	N
delay_output(3V) – SysV	S		N	S	S	N
delch(3V) – SysV	S		N	S	S	S
delete(3X)	A		N	N	N	S
deleteln(3V) – SysV	S		N	S	S	S
des_crypt(3)	N		N	N	N	N
des_setparity(3)	S		N	N	N	N
delwin(3V) – SysV	S		N	S	S	S
dlclose(3X)	S		N	N	S	N
dlerror(3X)	S		N	N	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
dlopen(3X)	S		N	N	S	N
dlsym(3X)	S		N	N	S	N
dn_comp(3)	S		N	N	S	N
dn_expand(3)	S		N	N	S	N
double_to_decimal(3)	S		N	N	N	N
doupdate(3V) – SysV	S		N	S	S	N
draino(3V) – SysV	S		N	N	N	N
drand48(3)	S		N	S	S	N
dysize(3V)	N		N	N	N	N
ecb_crypt(3)	S		N	N	N	N
echo(3V) – SysV	S		N	S	S	S
echochar(3V) – SysV	S		N	S	S	N
econvert(3)	S		N	N	N	N
ecvt(3)	S		N	N	S	N
edata(3)	S		N	N	S	N
encrypt(3)	S		N	S	S	N
end(3)	S		N	N	S	N
endac(3)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
endexportent(3)	A	The /etc/dfs/sharetab file replaces /etc/exports. Refer to share(1M), unshare(1M), and sharetab(4) for more information.	A	A	A	N
endfsent(3)	A	This routine is replaced by fclose(3).	A	A	A	N
endgraent(3)	N		N	N	N	N
endgrent(3V)	S		S	S	S	N
endhostent(3N)	S		N	N	S	N
endmntent(3)	A	This routine is replaced by fclose(3).	A	A	A	N
endnetent(3N)	S		N	N	S	N
endnetgrent(3N)	S		N	N	N	N
endprotoent(3N)	S		N	N	S	N
endpwaent(3)	N		N	N	N	N
endpwent(3V)	S		S	S	S	N
endrpcent(3N)	S		N	N	S	N
endservent(3N)	S		N	N	S	N
endttyent(3)	N	Refer to ttymon(1) and ttydefs(4) for information about SunOS release 5.6 tty system.	N	N	N	N
endusershell(3)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
endwin(3V)	C	The SunOS 4.x version of endwin() return value is undefined, while the SunOS release 5.6, or the SVID or SVR4 version returns OK upon success; otherwise , it returns ERR.	N	C	C	S
endwin(3V) – SysV	S		N	S	S	N
erand48(3)	S		N	S	S	N
erase(3V) – SysV	S		N	S	S	S
erasechar(3V) – SysV	S		N	S	S	N
erf(3M)	S		N	S	S	N
erfc(3M)	S		N	S	S	N
errno(3)	S		N	N	N	N
etext(3)	S		N	N	S	N
ether(3R)	N		N	N	N	N
ether_aton(3N)	S		N	N	S	N
ether_hostton(3N)	S		N	N	S	N
ether_line(3N)	S		N	N	S	N
ether_ntoa(3N)	S		N	N	S	N
ether_ntohost(3N)	S		N	N	S	N
exc_bound(3L)	N		N	N	N	N
exc_handle(3L)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
exc_notify(3L)	N		N	N	N	N
exc_on_exit(3L)	N		N	N	N	N
exc_raise(3L)	N		N	N	N	N
exc_unhandle(3L)	N		N	N	N	N
exc_uniqpatt(3L)	N		N	N	N	N
execl(3V)	C		C	C	C	N
execl(3V) – SysV	S		S	S	S	N
execle(3V)	C		C	C	C	N
execle(3V) – SysV	S		S	S	S	N
execlp(3V)	C		C	C	C	N
execlp(3V) – SysV	S		S	S	S	N
execv(3V)	C		C	C	C	N
execv(3V) – SysV	S		S	S	S	N
execvp(3V)	C		C	C	C	N
execvp(3V) – SysV	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
exit(3)	C	Both the SunOS 4.x and SunOS release 5.6, or the ABI, SVID, or SVR4 exit() routines do additional processing before the process exits. The SunOS 4.x exit() calls all functions registered by the on_exit(3) routine while SunOS release 5.6, or the ABI, SVID, or SVR4 exit() calls all functions registered by the atexit() routine. If no functions have been added using the on_exit(3) routine, then the SunOS 4.x and SunOS release 5.6, or the ABI, SVID, or SVR4 versions of exit() are compatible.	C	C	C	N
exp(3M)	C	In the SunOS release 5.6, or the SVID or SVR4 version, exp() returns HUGE for overflow and 0 for underflow. In the SunOS 4.x version, the return values are IEEE overflow and underflow (implementation-defined). In the SunOS release 4.x version, since HUGE is defined as +Infinity, exp(HUGE) and exp(-HUGE) do not overflow or underflow, hence no errno is produced. The SunOS release 5.6, or the SVID or SVR4 version sets errno to ERANGE.	N	C	C	N
exp10(3M)	N		N	N	N	N
exp2(3M)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
expm1(3M)	N		N	N	N	N
exportent(3)	A	The /etc/dfs/sharetab file replaces /etc/exports. Refer to share(1M), unshare(1M), and sharetab(4) man pages for more information.	A	A	A	N
extended_to_decimal(3)	S		N	N	N	N
fabs(3M)	S		N	S	S	N
fclose(3S)	S		S	S	S	N
fconvert(3)	S		N	N	N	N
fcvt(3)	S		N	N	S	N
fdopen(3V)	S		S	S	S	N
feof(3V)	S		S	S	S	N
ferror(3V)	S		S	S	S	N
fetch(3X)	A	This routine is replaced by dbm_fetch(3) in the SunOS release 5.6 software.	N	N	N	S
fflush(3S)	S		S	S	S	N
ffs(3)	S		N	N	S	N
fgetc(3V)	S		S	S	S	N
fgetgraent(3)	N		N	N	N	N
fgetgrent(3V)	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`fgetpwaent`(3)	N		N	N	N	N
`fgetpwent`(3V)	S		N	S	S	N
`fgets`(3S)	S		S	S	S	N
`fileno`(3V)	S		S	S	S	N
`file_to_decimal`(3)	N		N	N	N	N
`filter`(3V) – SysV	S		N	S	S	N
`finite`(3M)	N		N	N	N	N
`firstkey`(3X)	A	This routine is replaced by `dbm_firstkey`(3) in the SunOS release 5.6 software.	N	N	N	S
`fixterm`(3V)	A	The `reset_prog_mode`(3X) routine provides similar functionality.	N	A	A	N
`flash`(3V) – SysV	S		N	S	S	N
`floatingpoint`(3)	S		N	N	N	N
`floor`(3M)	S		N	S	S	N
`flushinp`(3V) – SysV	S		N	S	S	N
`flusok`(3X)	N		N	N	N	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`fmod`(3M)	C	In the SunOS release 5.6, or the SVID or SVR4 version, `fmod(x,0.0)` returns `x` and sets errno to `EDOM`. In the SunOS 4.x version, the same call returns `NaN` in conformance with 4.3 BSD and in the spirit of ANSI/ IEEE Std 754-1985.	N	C	C	N
`fopen`(3V)	S		S	S	S	S
`fp_class`(3M)	N		N	N	N	N
`fprintf`(3V)	S		S	S	S	S
`fputc`(3S)	S		S	S	S	N
`fputs`(3S)	S		S	S	S	N
`fread`(3S)	S		S	S	S	N
`free`(3)	S		S	S	S	N
`freopen`(3V)	S		S	S	S	S
`frexp`(3M)	S		N	S	S	N
`fscanf`(3V)	S		S	S	S	N
`fseek`(3S)	S		S	S	S	N
`ftell`(3S)	S		S	S	S	N
`ftime`(3V)	S	Now `ftime`(3C).	A	A	A	S
`ftok`(3)	S		S	S	S	N
`ftw`(3)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
func_to_decimal(3)	N		N	N	N	N
fwrite(3S)	S		S	S	S	N
gamma(3M)	S		N	S	S	N
garbagedlines(3V) – SysV	S		N	N	N	N
gcd(3X)	S		N	N	N	N
gconvert(3)	S		N	N	N	N
gcvt(3)	S		N	N	S	N
getacdir(3)	N		N	N	N	N
getacflg(3)	N		N	N	N	N
getacinfo(3)	N		N	N	N	N
getacmin(3)	N		N	N	N	N
getauditflagsbin(3)	N		N	N	N	N
getauditflagschar(3)	N		N	N	N	N
getbegyx(3V) – SysV	S		N	S	S	N
getc(3V)	S		S	S	S	N
getcap(3X)	N		N	N	N	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
getch(3V)	C	In the SunOS release 5.6, or the SVID or SVR4 software, if the window is not a pad, and it has been moved or modified since the last call to wrefresh(), wrefresh() will be called before another character is read. In the SunOS release 4.x software, wrefresh() will not be called under these circumstances.	N	C	C	S
getch(3V) – SysV	C		N	C	C	S
getchar(3V)	S		S	S	S	N
getcwd(3V)	S	The SVR4 and SunOS release 5.6 getcwd() routine is compatible with the SunOS 4.x version getcwd(). In the SunOS release 4.x, if buf is a NULL pointer, getcwd() obtains size bytes of space using malloc(3). This capability is not supported by the ABI and SVID version of getcwd().	C	C	S	N
getenv(3V)	S		S	S	S	N
getexportent(3)	A	The /etc/dfs/sharetab file replaces /etc/exports. Refer to share(1M), unshare(1M), and sharetab(4) man pages for more information.	A	A	A	N
getexportopt(3)	A	The /etc/dfs/sharetab file replaces /etc/exports. Refer to share(1M), unshare(1M), and sharetab(4) man pages for more information.	A	A	A	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`getfauditflags`(3)	N		N	N	N	N
`getfsent`(3)	A	This routine is replaced by `getvfsent`(3).	N	N	N	N
`getfsfile`(3)	A	This routine is replaced by `getvfsfile`(3).	N	N	N	N
`getfsspec`(3)	A	This routine is replaced by `getvfsfile`(3).	N	N	N	N
`getfstype`(3)	A	This routine is replaced by `getvfsany`(3).	N	N	N	N
`getgraent`(3)	N		N	N	N	N
`getgranam`(3)	N		N	N	N	N
`getgrent`(3V)	S		S	S	S	N
`getgrgid`(3V)	S		S	S	S	N
`getgrnam`(3V)	S		S	S	S	N
`gethostbyaddr`(3N)	S		N	N	S	N
`gethostbyname`(3N)	S		N	N	S	N
`gethostent`(3N)	S		N	N	S	N
`getlogin`(3V)	S		S	S	S	N
`getmaxyx`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
getmntent(3)	C	The SunOS 4.x getmntent() routine and the SunOS release 5.6, SVID, or SVR4 getmntent() routine are incompatible. The SunOS 4.x getmntent() returns a pointer to an object of type mntent while SunOS release 5.6, or the SVID or SVR4 getmntent() returns int. Additionally, SunOS release 5.6, or the SVID or SVR4 getmntent() uses a different incompatible structure type (mnttab) to return the file entry type. Additionally, null pointers are returned for corresponding '-' entries in /etc/vfstab.	N	C	C	N
get_myaddress(3N)	S	This routine is still available, but is superseded by netdir_getbyname(3N).	S	N	S	N
getnetbyaddr(3N)	S		N	N	S	N
getnetbyname(3N)	S		N	N	S	N
getnetent(3N)	S		N	N	S	N
getnetgrent(3N)	N		N	N	N	N
getnetname(3N)	S		S	S	S	N
getopt(3)	S		S	S	S	N
getpass(3V)	S		S	S	S	N
getprotobyname(3N)	S		N	N	S	N
getprotobynumber(3N)	S		N	N	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
getprotoent(3N)	S		N	N	S	N
getpublickey(3R)	S		S	S	S	N
getpw(3)	S		N	N	S	N
getpwaent(3)	N		N	N	N	N
getpwanam(3)	N		N	N	N	N
getpwent(3V)	S		S	S	S	N
getpwnam(3V)	S		S	S	S	N
getpwuid(3V)	S		S	S	S	N
getrpcbyname(3N)	S		S	S	S	N
getrpcbynumber(3N)	S		S	S	S	N
getrpcent(3N)	S		S	S	S	N
getrpcport(3R)	A	pmap_getport() can be used to get the same result.	N	N	N	N
gets(3S)	S		S	S	S	N
getsecretkey(3R)	S		S	S	S	N
getservbyname(3N)	S		N	N	S	N
getservbyport(3N)	S		N	N	S	N
getservent(3N)	S		N	N	S	N
getstr(3V) – SysV	C		N	C	C	S
getsubopt(3)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
getsyx(3V) – SysV	S		N	S	S	N
gettext(3)	C	In SunOs 5.4 gettext(3)a searches NLSPATH first for the location of the LC_MESSAGES directory.	N	N	N	N
gettmode(3V)	C	The SunOS release 5.6 header file <curses.h> automatically includes the headers <stdio.h> and <unctrl.h> and if CURS_PERFORMANCE is defined, it defines the most commonly used routines as macros for increased performance.	N	N	N	S
gettmode(3V) – SysV	S		N	N	N	N
getttyent(3)	A	Refer to ttymon(1) and ttydefs(4) for information about the SunOS release 5.6 tty system.	N	N	N	N
getttynam(3)	A	Refer to ttymon(1) and ttydefs(4) for information about the SunOS release 5.6 tty system.	N	N	N	N
getusershell(3)	S		N	N	N	N
getw(3V)	S		S	S	S	N
getwd(3)	S	Now getwd(3C).	A	A	A	S
getyx(3V) – SysV	S		N	S	S	S
gmtime(3V)	C	See ctime(3V).	C	C	C	N
grpauth(3)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
gsignal(3)	S		N	N	S	N
gtty(3C)	A	The termio(7) interface provides similar functionality.	A	A	A	N
halfdelay(3V) – SysV	S		N	S	S	N
has_ic(3V) – SysV	S		N	S	S	N
has_il(3V) – SysV	S		N	S	S	N
hasmntopt(3)	N		N	N	N	N
hcreate(3)	S		S	S	S	N
hdestroy(3)	S		S	S	S	N
host2netname(3N)	S		S	S	S	N
hsearch(3)	S		S	S	S	N
hypot(3M)	S		N	S	S	N
idlok(3V)	C	The SunOS 4.x version of idlok() sets an insert/delete line flag for the window, which is ignored, while SunOS release 5.6, or the SVID, or SVR4 version of idlok() sets a flag that controls whether the insert/delete line feature is actually used.	N	C	C	S
idlok(3V) – SysV	S		N	S	S	N
ieee_flags(3M)	N		N	N	N	N
ieee_functions(3M)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
ieee_handler(3M)	N		N	N	N	N
ieee_retrospective (3M)	N		N	N	N	N
ilogb(3M)	N		N	N	N	N
inch(3V) – SysV	S		N	S	S	S
index(3)	S	Now is index(3C).	A	A	A	S
inet_lnaof(3N)	S		N	N	S	N
inet_makeaddr(3N)	S		N	N	S	N
inet_netof(3N)	S		N	N	S	N
inet_network(3N)	S		N	N	S	N
inet_ntoa(3N)	S		N	N	S	N
infinity(3M)	N		N	N	N	N
initgroups(3)	S		S	S	S	N
initscr(3V)	C	The SunOS 4.x version of initscr() is a function while the SunOS release 5.6, SVID, or SVR4 version is a macro that calls initscr32(). If errors occur, the SunOS 4.x initscr() function returns ERR, while the SunOS release 5.6, SVID, or SVR4 version writes an appropriate error message to the standard error and exits.	N	C	C	S
initscr(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
initstate(3)	S	Now initstate(3C).	N	A	A	S
innetgr(3N)	S		N	N	N	N
insch(3V) – SysV	S		N	S	S	S
insertln(3V) – SysV	S		N	S	S	S
insque(3)	S		N	N	S	N
intrflush(3V) – SysV	S		N	S	S	N
ipalloc(3R)	N		N	N	N	N
irint(3M)	N	Replaced by (int) rint().	N	N	N	N
isalnum(3V)	S		S	S	S	N
isalpha(3V)	S		S	S	S	N
isascii(3V)	S		S	S	S	N
isatty(3V)	S		S	S	S	N
iscntrl(3V)	S		S	S	S	N
isdigit(3V)	S		S	S	S	N
isendwin(3V) – SysV	S		N	S	S	N
isgraph(3V)	S		S	S	S	N
isinf(3M)	N		N	N	N	N
islower(3V)	S		S	S	S	N
isnan(3M)	S		S	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
isnormal(3M)	N		N	N	N	N
isprint(3V)	S		S	S	S	N
ispunct(3V)	S		S	S	S	N
issecure(3)	N		N	N	N	N
isspace(3V)	S		S	S	S	N
issubnormal(3M)	N		N	N	N	N
isupper(3V)	S		S	S	S	N
isxdigit(3V)	S		S	S	S	N
iszero(3M)	N		N	N	N	N
itom(3X)	S		N	N	N	N
j0(3M)	C	In the SunOS release 4.x software, `j0(HUGE)`, `j1(HUGE)`, and `jn(4,HUGE)` will return `zero` with no error indication. In the SunOS release 5.6, SVID, or SVR4 software these routines will return `zero`, set errno to `ERANGE`, and print a message indicating a TLOSS math error on the standard error output.	N	C	C	N
j1(3M)	C		N	C	C	N
jn(3M)	C		N	C	C	N
jrand48(3)	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
key_decryptsession(3N)	S		S	S	S	N
key_encryptsession(3N)	S		S	S	S	N
key_gendes(3N)	S		S	S	S	N
key_setsecret(3N)	S		S	S	S	N
keyname(3V) – SysV	S		N	S	S	N
keypad(3V) – SysV	S		N	S	S	N
killchar(3V) – SysV	S		N	S	S	N
klm_prot(3R)	S		N	N	N	N
kvm_close(3K)	S		N	N	N	N
kvm_getcmd(3K)	S		N	N	N	N
kvm_getproc(3K)	S		N	N	N	N
kvm_getu(3K)	S		N	N	N	N
kvm_nextproc(3K)	S		N	N	N	N
kvm_nlist(3K)	S		N	N	N	N
kvm_open(3K)	S		N	N	N	N
kvm_read(3K)	S		N	N	N	N
kvm_setproc(3K)	S		N	N	S	N
kvm_write(3K)	S		N	N	N	N
l3tol(3C)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
l64a(3)	S		S	S	S	N
label(3X)	S		N	N	N	N
lcong48(3)	S		N	S	S	N
ldaclose(3X)	N		N	N	N	N
ldahread(3X)	N		N	N	N	N
ldaopen(3X)	N		N	N	N	N
ldclose(3X)	N		N	N	N	N
ldexp(3M)	C	The SunOS 4.x version of ldexp() differs from the SunOS release 5.6, ABI, SVID, or SVR4 version only in the case of overflow. The SunOS release 4.x ldexp() returns (+/-) 1.0e999 if the correct value would overflow, while the SunOS release 5.6, ABI, SVID, or SVR4 ldexp() returns (+/-) HUGE (according to the sign of value). Both versions set errno to ERANGE.	C	C	C	S
ldfcn(3)	N		N	N	N	N
ldfhread(3X)	N		N	N	N	N
ldgetname(3X)	N		N	N	N	N
ldlinit(3X)	N		N	N	N	N
ldlitem(3X)	N		N	N	N	N
ldlread(3X)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
ldlseek(3X)	N		N	N	N	N
ldnlseek(3X)	N		N	N	N	N
ldnrseek(3X)	N		N	N	N	N
ldnshread(3X)	N		N	N	N	N
ldnsseek(3X)	N		N	N	N	N
ldohseek(3X)	N		N	N	N	N
ldopen(3X)	N		N	N	N	N
ldrseek(3X)	N		N	N	N	N
ldshread(3X)	N		N	N	N	N
ldsseek(3X)	N		N	N	N	N
ldtbindex(3X)	N		N	N	N	N
ldtbread(3X)	N		N	N	N	N
ldtbseek(3X)	N		N	N	N	N
leaveok(3V) – SysV	S		N	S	S	S
lfind(3)	S		S	S	S	N
lgamma(3M)	S		N	S	S	N
line(3X)	S		N	N	N	N
linemod(3X)	S		N	N	N	N
localdtconv(3)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`localeconv`(3)	S		S	S	S	N
`localtime`(3V)	C	See `ctime`(3V).	C	C	C	N
`lockf`(3)	S		S	S	S	N
`log`(3M)	C	In the SunOS release 4.x software, when `log`() produces undefined results (for example, `log(-1.0)`), it returns `NaN`, with an `EDOM` error and a `DOMAIN` matherr. In the SunOS release 5.6, SVID or SVR4 version, it returns `-HUGE` with an `EDOM` error and `DOMAIN` matherr.	N	C	C	N
`log10`(3M)	C	In the SunOS release 4.x software, when `log10`() produces undefined results (for example, `log10(0)`) it returns `NaN`, with an `EDOM` error and a `DOMAIN` matherr. In the SunOS release 5.6, SVID, or SVR4 version, it returns `-HUGE` with an `EDOM` error and `DOMAIN` matherr.	N	C	C	N
`log1p`(3M)	N		N	N	N	N
`log2`(3M)	N		N	N	N	N
`logb`(3M)	S		N	C	C	N
`longjmp`(3V)	S		S	S	S	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
longname(3V)	C	The SunOS 4.x version of longname() requires two arguments, *termbuf* and *name*, which do not exist in the SunOS release 5.6, SVID, or SVR4 version. *termbuf* is a pointer to the terminal entry from termcap, which is replaced by terminfo. *name* is a pointer to a buffer to hold the result. Since both versions return the same information, simply remove the two arguments from the SunOS 4.x call to port to the SunOS release 5.6, SVID, or SVR4 environment.	N	C	C	S
longname(3V) – SysV	S		N	S	S	N
lrand48(3)	S		N	S	S	N
lsearch(3)	S		S	S	S	N
ltol3(3C)	N		N	N	N	N
lwp_checkstkset(3L)	N		N	N	N	N
lwp_create(3L)	N		N	N	N	N
lwp_ctxinit(3L)	N		N	N	N	N
lwp_ctxmemget(3L)	N		N	N	N	N
lwp_ctxmemset(3L)	N		N	N	N	N
lwp_ctxremove(3L)	N		N	N	N	N
lwp_ctxset(3L)	N		N	N	N	N
lwp_datastk(3L)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`lwp_destroy`(3L)	N		N	N	N	N
`lwp_enumerate`(3L)	N		N	N	N	N
`lwp_errstr`(3L)	N		N	N	N	N
`lwp_fpset`(3L)	N		N	N	N	N
`lwp_geterr`(3L)	N		N	N	N	N
`lwp_getregs`(3L)	N		N	N	N	N
`lwp_getstate`(3L)	N		N	N	N	N
`lwp_join`(3L)	N		N	N	N	N
`lwp_libcset`(3L)	N		N	N	N	N
`lwp_newstk`(3L)	N		N	N	N	N
`lwp_perror`(3L)	N		N	N	N	N
`lwp_ping`(3L)	N		N	N	N	N
`lwp_resched`(3L)	N		N	N	N	N
`lwp_resume`(3L)	N		N	N	N	N
`lwp_self`(3L)	N		N	N	N	N
`lwp_setpri`(3L)	N		N	N	N	N
`lwp_setregs`(3L)	N		N	N	N	N
`lwp_setstkcache`(3L)	N		N	N	N	N
`lwp_sleep`(3L)	N		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
lwp_stkcswset(3L)	N		N	N	N	N
lwp_suspend(3L)	N		N	N	N	N
lwp_yield(3L)	N		N	N	N	N
madd(3X)	S		N	N	N	N
madvise(3)	S		N	N	N	N
malloc(3)	S		S	S	S	N
malloc_debug(3)	S		N	N	N	N
malloc_verify(3)	S		N	N	N	N
mallocmap(3)	S		N	N	N	N
matherr(3M)	S		N	S	S	N
max_normal(3M)	N		N	N	N	N
max_subnormal(3M)	N		N	N	N	N
mblen(3)	S		S	S	S	N
mbstowcs(3)	S		S	S	S	N
mbtowc(3)	S		S	S	S	N
mcmp(3X)	S		N	N	N	N
mdiv(3X)	S		N	N	N	N
memalign(3)	S		N	N	S	N
memccpy(3)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
memchr(3)	S		S	S	S	N
memcmp(3)	S		S	S	S	N
memcpy(3)	S		S	S	S	N
memset(3)	S		S	S	S	N
meta(3V) – SysV	S		N	S	S	N
mfree(3X)	S		N	N	N	N
min(3X)	S		N	N	N	N
min_normal(3M)	N		N	N	N	N
min_subnormal(3M)	N		N	N	N	N
mkstemp(3)	S	The mktemp(3C) routine provides similar functionality.	A	A	A	N
mktemp(3)	C	The SunOS 4.x mktemp() routine replaces the trailing X characters of template with a letter and the current process ID. The SunOS release 5.6, ABI, SVID, or SVR4 version only specifies that it will replace the six trailing Xs with a character string that can be used to create a unique file name. If the application does not depend on the specific file name (that is, the application only cares that the name is unique), the SunOS 4.x and SunOS release 5.6, ABI, SVID, or SVR4 versions of mktemp() are compatible.	C	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`mlock`(3)	S		S	S	S	N
`mlockall`(3)	S		S	S	S	N
`modf`(3M)	S		N	S	S	N
`mon_break`(3L)	N		N	N	N	N
`mon_cond_enter`(3L)	N		N	N	N	N
`mon_create`(3L)	N		N	N	N	N
`mon_destroy`(3L)	N		N	N	N	N
`mon_enter`(3L)	N		N	N	N	N
`mon_enumerate`(3L)	N		N	N	N	N
`mon_exit`(3L)	N		N	N	N	N
`mon_waiters`(3L)	N		N	N	N	N
`moncontrol`(3)	A	This routine is replaced by `profil`(2).	A	A	A	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD

`monitor`(3)	C	The SunOS 4.x `monitor`() routine differs from the SunOS release 5.6, ABI, SVID, or SVR4 version in the following respects: In the SunOS release 4.x software, to profile the entire program it is sufficient to use:
		`extern etext;`
		`monitor(N_TXTOFF(0),etext, buf, bufsize, nfunc);`
		While with the SunOS release 5.6, ABI, SVID, or SVR4 `monitor`() routine, it is sufficient to use:
		`extern int etext;`
		`monitor((int(*)())2,etext, buf, bufsize, nfunc);`
		In the SunOS release 4.x software, to stop execution monitoring and write the results to the buf defined previously, use:
		`monitor(0);`
		While with the SunOS release 5.6, ABI, SVID, or SVR4 `monitor`() routine, use:
		`monitor((int(*)())0,(int(*)())0,(WORD*) 0, 0,0);`
		The `prof`(1) command can then be used to examine the results.
			C	C	C	N

`monstartup`(3)	A	This routine is replaced by `profil`(2).	A	A	A	N
`mout`(3X)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
move(3V) – SysV	S		N	S	S	S
mrand48(3)	S		N	S	S	N
msg_enumrecv(3L)	N		N	N	N	N
msg_enumsend(3L)	N		N	N	N	N
msg_recv(3L)	N		N	N	N	N
msg_reply(3L)	N		N	N	N	N
msg_send(3L)	N		N	N	N	N
msub(3X)	S		N	N	N	N
msync(3)	C	The following errno flag is valid for the SunOS 4.x version of this system call but is not valid in the SunOS release 5.6, ABI, SVID, or SVR4 version: EIO. In the SunOS 4.x version errno flag is set to EPERM if MS_INVALIDATE was specified and one or more of the pages is locked in memory, while in the SunOS release 5.6, ABI, SVID, or SVR4 version, errno is set to EBUSY instead.	C	C	C	N
mtox(3X)	S		N	N	N	N
mult(3X)	S		N	N	N	N
munlock(3)	S		S	S	S	N
munlockall(3)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
mvaddch(3V) – SysV	S		N	S	S	N
mvaddstr(3V) – SysV	S		N	S	S	N
mvcur(3V)	C	The return value of the SunOS 4.x version of mvcur() is undefined, while the SunOS release 5.6, SVID, or SVR4 version returns OK upon success; otherwise, it returns ERR.	N	C	C	S
mvcur(3V) – SysV	S		N	S	S	N
mvdelch(3V) – SysV	S		N	S	S	N
mvgetch(3V) – SysV	C	In the SunOS release 5.6, SVID, or SVR4 version, if the window is not a pad, and it has been moved or modified since the last call to wrefresh(), wrefresh() will be called before another character is read. In the SunOS release 4.x software, wrefresh() will not be called under these circumstances.	N	C	C	N
mvgetstr(3V) – SysV	C	See getstr(3V) — Sys V.	N	C	C	N
mvinch(3V) – SysV	S		N	S	S	N
mvinsch(3V) – SysV	S		N	S	S	N
mvprintw(3V)	C	See wprintw(3V).	N	C	C	S
mvprintw(3V) – SysV	S		N	S	S	N
mvscanw(3V)	C	See wscanw(3V).	N	C	C	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`mvscanw`(3V) – SysV	S		N	S	S	N
`mvwaddch`(3V) – SysV	S		N	S	S	N
`mvwaddstr`(3V – SysV	S		N	S	S	N
`mvwdelch`(3V) – SysV	S		N	S	S	N
`mvwgetch`(3V) – SysV	C	In the SunOS release 5.6, SVID, or SVR4 version, if the window is not a pad and it has been moved or modified since the last call to `wrefresh`(), `wrefresh`() will be called before another character is read. In the SunOS release 4.x software, `wrefresh`() will not be called under these circumstances.	N	C	C	N
`mvwgetstr`(3V) – SysV	C	See `getstr`(3V) — Sys V.	N	C	C	N
`mvwin`(3V)	C	The SunOS 4.x version of `mvwin`() can be used to move subwindows, while the SunOS release 5.6, SVID, or SVR4 `mvderwin`() should be used to move subwindows (or derived windows) inside their parent windows.	N	C	C	S
`mvwin`(3V) – SysV	S		N	S	S	N
`mvwinch`(3V) – SysV	S		N	S	S	N
`mvwinsch`(3V) – SysV	S		N	S	S	N
`mvwprintw`(3V)	C		N	C	C	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
mvwprintw(3V) – SysV	S		N	S	S	N
mvwscanw(3V)	C	See wscanw(3V).	N	C	C	S
mvwscanw(3V) – SysV	S		N	S	S	N
napms(3V) – SysV	S		N	S	S	N
net_addr(3N)	S		N	N	S	N
netname2host(3N)	S		S	S	S	N
netname2user(3N)	S		S	S	S	N
newpad(3V) – SysV	S		N	S	S	N
newterm(3V) – SysV	S		N	S	S	N
newwin(3V) – SysV	S		N	S	S	S
nextafter(3M)	S		N	S	S	N
nextkey(3X)	A	This routine is replaced by dbm_nextkey(3).	N	N	N	S
nice(3V)	S		S	S	S	S
nint(3M)	N		N	N	N	N
nl(3V) – SysV	S		N	S	S	S
nl_init(3C)	N		N	N	N	N
nl_langinfo(3C)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
nlist(3V)	C	The SunOS 4.x version of nlist() returns the number of symbols not found, or -1 on error. The SunOS release 5.6, SVID, or SVR4 version returns 0 on success, and -1 on error. Note that the SunOS release 5.6 nlist() assumes an ELF format file and the 4.1 nlist() works only on a.out format files.	N	C	C	S
nlm_prot(3R)	S		N	N	N	N
nocbreak(3V) – SysV	S		N	S	S	S
nocrmode(3X)	S		N	N	S	N
nodelay(3V) – SysV	S		N	S	S	N
noecho(3V) – SysV	S		N	S	S	S
nonl(3V) – SysV	S		N	S	S	S
nonstandard_arithmetic(3M)		N	N	N	N	N
noraw(3V) – SysV	S		N	S	S	S
notimeout(3V) – SysV	S		N	S	S	N
nrand48(3)	S		N	S	S	N
ntohl(3N)	S		N	N	S	N
ntohs(3N)	S		N	N	S	N
on_exit(3)	A	This routine is replaced by atexit(). Note that functions registered using atexit() are called without arguments.	A	A	A	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`opendir`(3V)	C	The SunOS release 5.6, ABI, SVID, or SVR4 `DIR` structure does not have the `dd_bsize` and `dd_off` fields. Also, the SunOS release 5.6, ABI, SVID, or SVR4 `dd_loc` and `dd_size` fields are int rather than long. The SunOS release 5.6, ABI, SVID, or SVR4 version includes `<sys/types.h>` while the SunOS 4.x version does not. The SunOS release 5.6, ABI, SVID, or SVR4 version sets errno to `ENOENT` when the directory name argument points to an empty string.	C	C	C	N
`opendir`(3V) – SysV	S		S	S	S	N
`openlog`(3)	S		N	N	N	N
`openpl`(3X)	N		N	N	N	N
`optarg`(3)	S		N	N	N	N
`optind`(3)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`overlay`(3V)	C	The SunOS 4.x `overlay`() is a function while the SunOS release 5.6, SVID, or SVR4 version of `overlay(srcwin,dstwin)` is a macro that calls `_overlap((srcwin), (dstwin), TRUE)`. The SunOS 4.x version of `overlay`() return value is undefined, while SunOS release 5.6, or the SVID or SVR4 version returns `OK` upon success otherwise it returns `ERR`.	N	C	C	S
`overlay`(3V) – SysV	S		N	S	S	N
`overwrite`(3V)	C	The SunOS 4.x version of `overwrite`() is a function while the SunOS release 5.6, SVID, or SVR4 version of `overwrite(srcwin, dstwin)` is a macro that calls `_overlap((srcwin), (dstwin), FALSE)`. The SunOS 4.x `overwrite`() return value is undefined, while the SunOS release 5.6, SVID, or SVR4 version returns `OK` upon success; otherwise, it returns `ERR`.	N	C	C	S
`overwrite`(3V) – SysV	S		N	S	S	N
`passwd2des`(3R)	S		N	N	N	N
`pause`(3V)	S		S	S	S	N
`pclose`(3S)	S		S	S	S	N
`pechochar`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
perror(3)	S		S	S	S	N
plock(3)	S		S	S	S	N
plot(3X)	S		N	N	N	N
pmap_getmaps(3N)	S	This routine is still available, but is superseded by rpcb_getmaps(3N).	A	A	S	N
pmap_getport(3N)	S	This routine is still available, but is superseded by rpcb_getaddr(3N).	A	A	S	N
pmap_rmtcall(3N)	S	This routine is still available, but is superseded by rpcb_rmtcall(3N).	A	A	S	N
pmap_set(3N)	S	This routine is still available, but is superseded by rpcb_set(3N).	A	A	S	N
pmap_unset(3N)	S	This routine is still available, but is superseded by rpcb_unset(3N).	A	A	S	N
pnoutrefresh(3V) – SysV	S		N	S	S	N
pnp(3R)	N		N	N	N	N
pod_getexit(3L)	N		N	N	N	N
pod_getmaxpri(3L)	N		N	N	N	N
pod_getmaxsize(3L)	N		N	N	N	N
pod_setexit(3L)	N		N	N	N	N
pod_setmaxpri(3L)	N		N	N	N	N
point(3X)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
popen(3S)	S		S	S	S	N
pow(3M)	C	In the SunOS release 5.6, SVID, or SVR4 version, the routine returns `0` when *x* == 0 and *y* is non-positive or when *x*<0 and *y* is not integral. For overflow or underflow, `pow`() returns `+/-HUGE` or `0`, respectively. In both cases, errno is set. In the SunOS 4.x version, `pow(x,0.0)` is 1 (which is not mentioned in the SunOS release 5.6, SVID, or SVR4 version); it returns `NaN` when x < 0 and y not integral, returns `+/-infinity` when *x*==0 and *y* < 0. On overflow and underflow, it returns IEEE implementation-dependent values. In the SunOS release 4.x version, since HUGE is defined as +Infinity, `pow(10.0,HUGE)` and `pow(10.0,-HUGE)` do not underflow or overflow and therefore no errno is produced. In the SunOS release 5.6, SVID, or SVR4 software, these functions set errno to `ERANGE`.	N	C	C	N
prefresh(3V) – SysV	S		N	S	S	N
printf(3V)	S	See `fprintf`(3).	S	S	S	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
printw(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of printw() returns the integer ERR upon failure and an integer value other than ERR upon successful completion. The SunOS 4.x version returns void. The SunOS release 5.6, SVID, or SVR4 headers <curses.h> automatically include the headers <stdio.h> and <unctrl.h> and if CURS_PERFORMANCE is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
printw(3V) – SysV	S		N	S	S	N
prof(3)	A	The profil(2) routine provides similar functionality.	A	A	A	N
psignal(3)	C	The *sig* argument is defined as an unsigned int in the SunOS 4.x versionbut is defined as an int in the SVR4 and SunOS release 5.6 versions.	N	N	C	S
putc(3S)	S		S	S	S	N
putchar(3S)	S		S	S	S	N
putenv(3)	S		S	S	S	N
putp(3V) – SysV	S		N	S	S	N
putpwent(3)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
puts(3S)	S		S	S	S	N
putw(3S)	S		S	S	S	N
pwdauth(3)	N		N	N	N	N
qsort(3)	S		S	S	S	N
quiet_nan(3M)	N		N	N	N	N
rand(3V)	S		S	S	S	S
random(3)	A	Now random(3C). The drand48(3C) (for SunOS release 5.6, SVID, or SVR4 software) or rand(3C) routines provide similar functionality.	A	A	A	S
raw(3V) – SysV	S		N	S	S	S
rcmd(3N)	S		N	N	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`readdir`(3V)	C	The `dirent` structures of SunOS 4.x `readdir`() and the ABI and SVID versions only have the `d_name` field in common. The SunOS 4.x `readdir`() supports an obsolete data structure `direct` defined in `<sys/dir.h>`, which is no longer supported by the SunOS release 5.6, ABI, SVID or SVR4 softwqre. Applications must migrate to the `dirent` structure defined in `<dirent.h>`. SunOS release 5.6, ABI, SVID, or SVR4 `readdir`() updates the directories last accessed time. The `dirent` structures of SunOS 4.x, SVR4, and SunOS release 5.6only have the `d_name` and `d_reclen` fields in common. Also, SunOS release 5.6 `dd_loc` and `dd_size` fields are type int rather than type long as in SunOS 4.x.	C	C	C	S
`readdir`(3V) – SysV	C	The SunOS 4.x, SVR4, and SunOS release 5.6 `dirent` structures only have the `d_name` and `d_reclen` fields in common. Also, the SunOS release 5.6 `dd_loc` and `dd_size` fields are type int rather than type long as in the SunOS 4.x software. The SunOS release 5.6, ABI, SVID, or SVR4 `readdir`() updates the directory's last accessed time. The `dirent` structures of SunOS 4.x `readdir`() and the ABI and SVID versions only have the `d_name` field in common.	C	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
realloc(3)	C	The SunOS release 4.x realloc() accepts a pointer to a block freed since the most recent call to malloc(), calloc(), and realloc(). The SunOS release 5.6, ABI, SVID, or SVR4 realloc() does not accept such a pointer.	C	C	C	N
realpath(3)	S		N	N	S	N
re_comp(3)	A	Now re_comp(3C). For the ABI and SVID version, the regexp(3) general-purpose regular expression matching routines provide similar functionality. This routine is replaced by recomp(3G).	A	A	A	S
re_exec(3)	A	Now re_exec(3C). For the ABI and SVID version, the regexp(3) general-purpose regular expression matching routines provide similar functionality. This routine is replaced by regex(3G).	A	A	A	S
refresh(3V) – SysV	S		N	S	S	S
registerrpc(3N)	S	This routine is still available, but is superseded by rpc_reg(3C).	N	N	S	N
remainder(3M)	S		S	S	S	N
remexportent(3)	N	The /etc/dfs/sharetab file replaces /etc/exports. Refer to share(1M), unshare(1M), and sharetab(4) man pages for more information.	N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
remque(3)	S		N	N	S	N
resetterm(3V)	A	This routine is replaced by reset_shell_mode(3).	N	A	A	N
res_init(3)	S		N	N	S	N
res_mkquery(3)	S		N	N	S	N
res_send(3)	S		N	N	S	N
reset_prog_mode(3V) – SysV	S		N	S	S	N
reset_shell_mode(3V) – SysV	S		N	S	S	N
resetty(3V) – SysV	S		N	S	S	S
restartterm(3V) – SysV	S		N	S	S	N
rewind(3S)	S		S	S	S	N
rewinddir(3V)	S		S	S	S	N
rex(3R)	S		N	N	N	N
rexec(3N)	S		N	N	S	N
rindex(3)	S	Now rindex(3C).	A	A	A	S
rint(3M)	S		N	N	S	N
ripoffline(3V) – SysV	S		N	S	S	N
rnusers(3R)	N		N	N	N	N
rpc_createerr(3N)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
rpow(3X)	S		N	N	N	N
rquota(3R)	N		N	N	N	N
rresvport(3N)	S		N	N	S	N
rstat(3R)	N		N	N	N	N
rtime(3N)	N		N	N	N	S
ruserok(3N)	S		N	N	N	N
rusers(3R)	S		N	N	S	N
rwall(3R)	S		N	N	S	N
saveterm(3V)	A	This routine is replaced by def_prog_mode(3X).	N	A	A	N
savetty(3V) – SysV	S		N	S	S	S
scalb(3M)	C	In the SunOS release 5.6, SVID, or SVR4 version, the routine computes the value *x* * (*r****n*) where *r* is the radix of the machine's floating point arithmetic. When *r* == 2, scalb() is the same as ldexp(3M) routine. On overflow, the routine returns +/- HUGE (depending on the sign of *x*). On underflow, it returns 0 and sets the errno. In the SunOS 4.x version, the routine computes the value *x* * (2***n*) at all times; scalb() is not defined when *y* is not integral.	N	C	C	N
scalbn(3M)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`scandir`(3)	N		N	N	N	S
`scanf`(3V)	S		S	S	S	N
`scanw`(3V)	C	In the SunOS release 5.6, SVID, or SVR4 software the header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>` and if `CURS_PERFORMANCE` is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
`scanw`(3V) – SysV	S		N	S	S	N
`scr_dump`(3V) – SysV	S		N	S	S	N
`scr_init`(3V) – SysV	S		N	S	S	N
`scr_restore`(3V) – SysV	S		N	S	S	N
`scroll`(3V)	C	`scroll`() returns `ERR` on failure and an indeterminate value for success. The SunOS 4.x version returns `ERR` on failure and `OK (0)` on success. In the SunOS release 5.6, SVID, or SVR4 version the header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>` and if `CURS_PERFORMANCE` is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
`scroll`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
scrollok(3V) – SysV	S		N	S	S	S
seconvert(3)	S		N	N	N	N
seed48(3)	S		N	S	S	N
seekdir(3V)	S		S	S	S	N
setac(3)	N		N	N	N	N
setbuf(3V)	S		S	S	S	S
setbuffer(3V)	S		N	N	N	S
set_curterm(3V) – SysV	S		N	S	S	N
setegid(3V)	S		N	N	N	N
seteuid(3V)	S		N	N	N	N
setexportent(3)	A	The /etc/dfs/sharetab file replaces /etc/exports. Refer to share(1M), unshare(1M), and sharetab(4) man pages for more information.	N	N	N	N
setfsent(3)	A	This routine is replaced by fopen(3).	A	A	A	N
setgid(3V)	S		S	S	S	N
setgraent(3)	N		N	N	N	N
setgrent(3V)	S		S	S	S	N
sethostent(3N)	S		N	N	S	N
setjmp(3V)	S		S	S	S	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
setkey(3)	S		N	S	S	N
setlinebuf(3V)	S		N	N	N	S
setlocale(3V)	C		S	S	S	N
setlogmask(3)	S		N	N	N	N
setmntent(3)	A	The fopen(3) followed by the lockf(3) routines provide similar functionality.	A	A	A	N
setnetent(3N)	S		N	N	S	N
setnetgrent(3N)	S		N	N	N	N
setprotoent(3N)	S		N	N	S	N
setpwaent(3)	N		N	N	N	N
setpwent(3V)	S		S	S	S	N
setpwfile(3V)	N		N	N	N	N
setrgid(3V)	A	This routine is replaced by setgid(2).	A	A	A	N
setrpcent(3N)	S		N	N	S	N
setruid(3V)	A	This routine is replaced by setuid(2).	A	A	A	N
setscrreg(3V) – SysV	S		N	S	S	N
setservent(3N)	S		N	N	S	N
setstate(3)	S	Now setstate(3C).	N	A	A	S
setsyx(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`set_term`(3V) – SysV	S		N	S	S	N
`setterm`(3V)	C	This is an obsolete call that is replaced by `setupterm`() in both the SunOS 4.x and SunOS release 5.6 software. See `curs_terminfo`(3X). The call: `setupterm(term, 1, (int *) 0)` provides the same functionality as `setterm(term)`.	N	C	C	S
`setterm`(3V) – SysV	S		N	S	S	N
`setttyent`(3)	N	Refer to `ttymon`(1) and `ttydefs`(4) man pages for information about SunOS release 5.6 tty system.	N	N	N	N
`setuid`(3V)	S		S	S	S	N
`setupterm`(3V) – SysV	S		N	S	S	N
`setusershell`(3)	S		N	N	N	N
`setvbuf`(3V)	S		S	S	S	S
`sfconvert`(3)	S		N	N	N	N
`sgconvert`(3)	S		N	N	N	N
`sigaction`(3V)	S		S	S	S	N
`sigaddset`(3V)	S		S	S	S	N
`sigdelset`(3V)	S		S	S	S	N
`sigemptyset`(3V)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`sigfillset`(3V)	S		S	S	S	N
`sigfpe`(3)	S		N	N	N	N
`siginterrupt`(3V)	A	The `sigaction`(2) routine provides similar functionality.	A	A	A	S
`sigismember`(3V)	S		S	S	S	N
`siglongjmp`(3V)	S		S	S	S	N
`signal`(3V)	C	The following SunOS 4.x signal is not defined in the SVR4 and SunOS release 5.6 `signal`(2) routine: SIGLOST. The following SunOS 4.x signals are not defined in the ABI and SVID `signal` routine: SIGIO, SIGURG, SIGVTALRM, SIGPROF, SIGLOST.	C	C	C	S
`ssignal`(3V)	C		C	C	C	N
`signaling_nan`(3M)	N		N	N	N	N
`signbit`(3M)	N		N	N	N	N
`significand`(3M)	N		N	N	N	N
`sigsetjmp`(3V)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`sin`(3M)	C	For arguments that are much lower than zero, the SunOS release 5.6, SVID, or SVR4 version of these routines return zero because of the loss of significance. In this case, a message indicating TLOSS (see `matherr`(3M)) appears on the standard output. For cases of partial loss of significance, a PLOSS error is generated but no error is printed. In both cases, errno is set to `ERANGE`. In the SunOS 4.x version, an argument reduction takes place for values exceeding PI/4 in magnitude. The reduction could happen in software or hardware. The variable *fp_pi* defined in `<math.h>` allows changing of the precision at runtime. The error exceptions occur in the IEEE 754 spirit for both versions.	N	C	C	N
`sinh`(3M)	S		N	S	S	N
`single_precision`(3M)	N		N	N	N	N
`single_to_decimal`(3)	S		N	N	N	N
`sleep`(3V)	S		S	S	S	S
`slk_clear`(3V) – SysV	S		N	S	S	N
`slk_init`(3V) – SysV	S		N	S	S	N
`slk_label`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`slk_noutrefresh`(3V) – SysV	S		N	S	S	N
`slk_refresh`(3V) – SysV	S		N	S	S	N
`slk_restore`(3V) – SysV	S		N	S	S	N
`slk_set`(3V) – SysV	S		N	S	S	N
`slk_touch`(3V) – SysV	S		N	S	S	N
`sm_inter`(3R)	S		N	N	N	N
`space`(3X)	S		N	N	N	N
`spray`(3R)	S		N	N	S	N
`sprintf`(3V)	S	See `fprintf`(3).	S	S	S	S
`sqrt`(3M)	C	In the SunOS release 4.x software, when `sqrt`() produces undefined results (for example, `sqrt (-3.0)`) it returns `NaN`, with an `EDOM` error and a `DOMAIN` matherr. The SunOS release 5.6, SVID, or SVR4 version returns `0` with an EDOM error and a `DOMAIN` matherr.	N	C	C	N
`srand`(3V)	C	In the SunOS release 4.x software, argument *seed* is defined as int while in the SunOS release 5.6, ABI, SVID, or SVR4 software it is defined as unsigned int.	C	C	C	S
`srand48`(3)	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
srandom(3)	S	Now srandom(3C). The srand48(3C) (in the SunOS release 5.6, SVID, or SVR4 software) or srand(3C) routines provide similar functionality.	A	A	A	S
sscanf(3V)	S		S	S	S	N
ssignal(3)	S		N	N	S	N
standard_arithmetic(3M)	N		N	N	N	N
standend(3V) – SysV	S		N	S	S	S
standout(3V) – SysV	S		N	S	S	S
store(3X)	A	This routine is replaced by dbm_store(3).	N	N	N	S
strcasecmp(3)	S		N	N	N	N
strcat(3)	S		S	S	S	N
strchr(3)	S		S	S	S	N
strcmp(3)	S		S	S	S	N
strcoll(3)	S		S	S	S	N
strcpy(3)	S		S	S	S	N
strcspn(3)	S		S	S	S	N
strdup(3)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`strftime`(3V)	C	There are some differences in the directives specified in the following formats: `%k` and `%l` - Not supported in the SunOS release 5.6 software. `%S`- the SunOS 4.x software specifies seconds to be in the range of 0-59, while the SunOS release 5.6 software defines seconds to be in the range of 0-61 (allows for leap seconds). `%V`,`%W`- Under the SunOS release 4.x software , week number 01 is the first week in January with four or more days in it, while in the SunOS release 5.6 software, week number 01 is the first week in January starting with a Sunday for `%U` or a Monday for `%W`. The SunOS 4.1 `tm` structure contains two fields not present in the SunOS release 5.6 `tm` structure: `tm_zone` and `tm_gmtoff`. Instead, the SunOS release 5.6 version uses the external variable *timezone* to contain the difference (in seconds) between GMT and local standard time, and the external variable *daylight* to indicate if daylight savings should be applied. Additionally, the SunOS release 5.6 version uses an external variable *tzname* to store standard and summer time-zone names. These external variables (*timezone*, *daylight*, and *tzname*) are supported by the SunOS 4.x System V installation option `ctime`(3V) library routines.	C	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`string_to_decimal`(3)	N		N	N	N	N
`strlen`(3)	S		S	S	S	N
`strncasecmp`(3)	S		N	N	N	N
`strncat`(3)	S		S	S	S	N
`strncmp`(3)	S		S	S	S	N
`strncpy`(3)	S		S	S	S	N
`strpbrk`(3)	S		S	S	S	N
`strptime`(3V)	S		A	A	A	N
`strrchr`(3)	S		S	S	S	N
`strspn`(3)	S		S	S	S	N
`strtod`(3)	C	The SunOS 4.x `strtod`() and `atof`() routines accept *inf_form*, *infinity_form*, *nan_form*, and *nanstring_form*, while the SunOS release 5.6, ABI, SVID, or SVR4 `strtod`() and `atof`() routines do not accept these forms.	C	C	C	N
`strtok`(3)	S		S	S	S	N
`strtol`(3)	S		S	S	S	N
`strxfrm`(3)	S		S	S	S	N
`stty`(3C)	A	The `termio`(7) interface provides similar functionality.	A	A	A	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
subpad(3V) – SysV	S		N	S	S	N
subwin(3V)	C	The SunOS release 5.6, SVID, or SVR4 routine returns a null pointer if a failure occurs. The SunOS release 5.6, SVID, or SVR4 header file <curses.h> automatically includes the header files <stdio.h> and <unctrl.h> and if CURS_PERFORMANCE is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
subwin(3V) – SysV	S		N	S	S	N
svc_destroy(3N)	S		S	S	S	N
svc_fds(3N)	S	This routine is still available, but is superseded by svc_fdset(3N).	N	S	S	N
svc_fdset(3N)	S		S	S	S	N
svc_freeargs(3N)	S		S	S	S	N
svc_getargs(3N)	S		S	S	S	N
svc_getcaller(3N)	S	This routine is still available, but is superseded by svc_getrpccaller(3N).	A	A	A	N
svc_getreq(3N)	S	This routine is still available, but is superseded by svc_getreqset(3N).	S	S	S	N
svc_getreqset(3N)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
svc_register(3N)	A	This routine is still available, but it superseded by svc_reg(3N).	A	A	A	N
svc_run(3N)	S		S	S	S	N
svc_sendreply(3N)	S		S	S	S	N
svc_unregister(3N)	A	This routine is still available, but is superseded by svc_unreg(3N).	A	A	A	N
svcerr_auth(3N)	S		S	S	S	N
svcerr_decode(3N)	S		S	S	S	N
svcerr_noproc(3N)	S		S	S	S	N
svcerr_noprog(3N)	S		S	S	S	N
svcerr_progvers(3N)	S		S	S	S	N
svcerr_systemerr(3N)	S		S	S	S	N
svcerr_weakauth(3N)	S		S	S	S	N
svcfd_create(3N)	A	This routine is still available, but is superseded by svc_fd_create(3N).	A	A	A	S
svcraw_create(3N)	S	This routine is still available, but is superseded by svc_raw_create(3N).	N	N	S	N
svctcp_create(3N)	S	This routine is still available, but is superseded by svc_create(3N), svc_tli_create(3N), and svc_vc_create(3N).	N	N	S	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
svcudp_bufcreate(3N)	S	This routine is still available, but is superseded by the svc_tli_create(3N), and svc_dg_create(3N) routines.	N	N	S	S
svcudp_create(3N)	S	This routine is still available, but is superseded by svc_create(3N), svc_tli_create(3N), and svc_dg_create(3N).	N	N	S	S
swab(3)	S		S	S	S	N
sys_siglist(3)	N	Use psignal(3C).	N	N	N	S
syslog(3)	S		N	N	S	N
system(3)	S		S	S	S	N
t_accept(3N)	S		S	S	S	N
t_alloc(3N)	S		S	S	S	N
t_bind(3N)	S		S	S	S	N
t_close(3N)	S		S	S	S	N
t_connect(3N)	S		S	S	S	N
t_error(3N)	S		S	S	S	N
t_free(3N)	S		S	S	S	N
t_getinfo(3N)	S		S	S	S	N
t_getstate(3N)	S		S	S	S	N
t_listen(3N)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
t_look(3N)	S		S	S	S	N
t_open(3N)	S		S	S	S	N
t_optmgmt(3N)	S		S	S	S	N
t_rcv(3N)	S		S	S	S	N
t_rcvconnect(3N)	S		S	S	S	N
t_rcvdis(3N)	S		S	S	S	N
t_rcvrel(3N)	S		S	S	S	N
t_rcvudata(3N)	S		S	S	S	N
t_rcvuderr(3N)	S		S	S	S	N
t_snd(3N)	S		S	S	S	N
t_snddis(3N)	S		S	S	S	N
t_sndrel(3N)	S		S	S	S	N
t_sndudata(3N)	S		S	S	S	N
t_sync(3N)	S		S	S	S	N
t_unbind(3N)	S		S	S	S	N
tan(3M)	S		N	S	S	N
tanh(3M)	S		N	S	S	N
tcdrain(3V)	S		S	S	S	N
tcflow(3V)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
tcflush(3V)	S		S	S	S	N
tcgetattr(3V)	S		S	S	S	N
tcgetpgrp(3V)	S		S	S	S	N
tcsendbreak(3V)	S		S	S	S	N
tcsetattr(3V)	S		S	S	S	N
tcsetpgrp(3V)	S		S	S	S	N
tdelete(3)	S		S	S	S	N
telldir(3V)	S		S	S	S	N
tempnam(3S)	S		S	S	S	N
textdomain(3)	N		N	N	N	N
tfind(3)	S		S	S	S	N
tgetent(3X)	C	The SunOS release 5.6, SVID, or SVR4 software is supporting this routine as a conversion aid and it should not be used in new applications. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
tgetent(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`tgetflag`(3X)	C	The SunOS release 5.6, SVID, or SVR4 software is supporting this routine as a conversion aid and it should not be used in new applications. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
`tgetflag`(3V) – SysV	S		N	S	S	N
`tgetnum`(3X)	C	The SunOS release 5.6, SVID, or SVR4 software is supporting this routine as a conversion aid and it should not be used in new applications. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
`tgetnum`(3V) – SysV	S		N	S	S	N
`tgetstr`(3X)	C	The SunOS release 5.6, SVID, or SVR4 software is supporting this routine as a conversion aid and it should not be used in new applications. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
`tgetstr`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`tgoto`(3X)	C	The SunOS release 5.6, SVID, or SVR4 software is supporting this routine as a conversion aid and it should not be used in new applications. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
`tgoto`(3V) – SysV	S		N	S	S	N
`tigetflag`(3V) – SysV	S		N	S	S	N
`tigetnum`(3V) – SysV	S		N	S	S	N
`tigetstr`(3V) – SysV	S		N	S	S	N
`time`(3V)	S		S	S	S	N
`timegm`(3V)	A	This routine is replaced by `mktime`(3C).	A	A	A	N
`timelocal`(3V)	S	This routine is the inverse of `localtime`(3C).	A	A	A	N
`times`(3V)	C	The SunOS 4.x `times`() routine returns time values in units of 1/HZ seconds, where HZ is 60. The SunOS release 5.6, ABI, SVID, or SVR4 `times`() routine returns time values in units of 1/CLK_TCK of a second.	C	C	C	S
`timezone`(3C)	S		N	N	N	N
`tmpfile`(3S)	C		C	C	C	N
`tmpnam`(3S)	S		S	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
toascii(3V)	S		S	S	S	N
toascii(3V) – SysV	S		S	S	S	N
tolower(3V)	S		S	S	S	N
tolower(3V) – SysV	C	The SunOS release 5.6, ABI, SVID, or SVR4 version of this routine is affected by the program's locale as specified by LC_CTYPE, while the SunOS 4.x version is not.	C	C	C	N
touchline(3V)	C	The SunOS release 5.6, ABI, SVID, or SVR4 version of this routine returns ERR on failure and an integer other than ERR on success.	N	C	C	S
touchline(3V) – SysV	S		N	S	S	N
touchoverlap(3X)	N		N	N	N	S
touchwin(3V)	C	The SunOS release 5.6, ABI, SVID, or SVR4 version of this routine returns ERR on failure and an integer other than ERR on success.	N	C	C	S
touchwin(3V) – SysV	S		N	S	S	N
toupper(3V)	S		S	S	S	N
toupper(3V) – SysV	C	The SunOS release 5.6, ABI, SVID, or SVR4 version of this routine is affected by the program's locale as specified by LC_CTYPE, while the SunOS 4.x version is not.	C	C	C	N
tparm(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`tputs`(3V)	C	The SunOS release 5.6, SVID, or SVR4 software supports this routine as a conversion aid. It should not be used in new applications. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
`tputs`(3V) – SysV	S		N	S	S	N
`traceoff`(3V) – SysV	S		N	N	S	N
`traceon`(3V) – SysV	S		N	N	S	N
`tsearch`(3)	S		S	S	S	N
`ttyname`(3V)	S		S	S	S	N
`ttyslot`(3V)	S		N	N	S	N
`twalk`(3)	S		S	S	S	N
`typeahead`(3V) – SysV	S		N	S	S	N
`tzset`(3V)	C	See `ctime`(3V).	C	C	C	N
`tzsetwall`(3V)	A	This routine is replaced by `tzset`(3C) .	A	A	A	N
`ualarm`(3)	S	Now `ualarm`(3C). The `setitimer`(2) system call with the *which* argument set to ITIMER_REAL provides similar functionality.	N	A	A	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`ulimit`(3C)	S	The SVR4 and SunOS release 5.6 `ulimit`() is compatible with the SunOS 4.x `ulimit`(). The SunOS 4.x version of `ulimit`() routine's integer *cmd* values 1 and 2 may not be compatible with the equivalent SVID `ulimit`() routines' symbolic constant *cmd* values UL_GETFSIZE and UL_SETFSIZE. Also, the SVID `ulimit`() routine does not support the functionality of 3 (get the maximum possible break value) and 4 (get the size of the process's file descriptor table).	C	C	S	N
`unctrl`(3V) – SysV	S		N	S	S	S
`ungetc`(3S)	S	The SVR4 and SunOS release 5.6 `ungetc`() guarantees to push back four characters, so it is compatible with the SunOS 4.x `ungetc`(). In the SunOS release 4.x software, `ungetc`() is guaranteed to push back one character on the standard input without a previous read statement, while the ABI and SVID `ungetc`() does not support this attribute.	C	C	S	N
`ungetch`(3V) – SysV	S		N	S	S	N
`user2netname`(3N)	S		S	S	S	N
`usleep`(3)	S	Now `usleep`(3C). The `setitimer`(2) or `select`(3C) routines provide similar functionality.	N	A	A	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`utime`(3V)	C	The SunOS 4.x `utime`() and SunOS release 5.6, ABI, SVID, or SVR4 `utime`() differ in the type of the second argument. In the SunOS release 4.x software, argument *timep* points to an array of two *time_t* values, while in theSunOS release 5.6, ABI, SVID, or SVR4 version, argument *times* points to the `utimbuf` structure (which contains two `time_t` members).	C	C	C	N
`valloc`(3)	S		N	N	S	N
`varargs`(3)	S		N	N	N	N
`vfprintf`(3V)	C	See `vprintf`(3V).	C	C	C	S
`vidattr`(3V) – SysV	S		N	S	S	N
`vidputs`(3V) – SysV	S		N	S	S	N
`vlimit`(3C)	A	This routine is replaced by `getrlimit`(2).	A	A	A	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
vprintf(3V)	C	The SunOS 4.x vprintf(), vfprintf(), and vsprintf() routines are incompatible with the SunOS release 5.6, ABI, SVID, or SVR4 version of these routines because of variable format list differences. In the SunOS release 4.x software, (*va_alist* (defined in <varargs.h>) is used in a function header to declare a variable argument list (for example, void function(*va_alist*)). In theSunOS release 5.6, ABI, SVID, or SVR4 version the definition from <stdarg.h> is used in a function header to declare a variable argument list (for example, void function (int arg1,...)).	C	C	C	S
vsprintf(3V)	C	See vprintf(3V).	C	C	C	S
vsyslog(3)	S	This routine is replaced by syslog(3).	N	N	N	N
vtimes(3C)	A	This routine is replaced by getrusage(2).	N	N	N	N
vwprintw(3V) – SysV	S		N	S	S	N
vwscanw(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
waddch(3V)	C	The characters in the SunOS release 5.6 software are chtype (long) if CHTYPE is not defined differently for <curses.h>. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion. The SunOS release 5.6, SVID, or SVR4 header file <curses.h> automatically includes the headers <stdio.h> and <unctrl.h> and if CURS_PERFORMANCE is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
waddch(3V) – SysV	S		N	S	S	N
waddstr(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of waddstr(3V) returns ERR (-1) on failure. The SunOS release 5.6, SVID, or SVR4 header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h> and if CURS_PERFORMANCE is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
waddstr(3V) – SysV	S		N	S	S	N
wattroff(3V) – SysV	S		N	S	S	N
wattron(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`wattrset`(3V) – SysV	S		N	S	S	N
`wclear`(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of `wclear`() routine always returns (`OK = 0`) upon success while the SunOS 4.x software returns `void`. The SunOS release 5.6, SVID, or SVR4 header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>` and if `CURS_PERFORMANCE` is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
`wclear`(3V) – SysV	S		N	S	S	N
`wclrtobot`(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of `wclrtobot`(3V) routine always returns (`OK = 0`) upon success while the SunOS 4.x software returns `void`. The SunOS release 5.6, SVID, or SVR4 header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>` and if `CURS_PERFORMANCE` is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
`wclrtobot`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`wclrtoeol`(3V)	C	The SunOS release 5.6, SVID, or SVR4 routine always returns (`OK = 0`) upon success while the SunOS 4.x software returns `void`. The SunOS release 5.6, SVID, or SVR4 header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>` and if `CURS_PERFORMANCE` is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
`wclrtoeol`(3V) – SysV	S		N	S	S	N
`wcstombs`(3)	S	The size of wchar_t is short in the SunOS 4.x software and long in the SunOS release 5.6 software.	S	S	S	N
`wctomb`(3)	S	The size of wchar_t is short in the SunOS 4.x software and long in the SunOS release 5.6 software.	S	S	S	N
`wdelch`(3V)	C	In the SunOS release 5.6, SVID, or SVR4 software this routine may be a macro, while it always is in the SunOS 4.x software. The SunOS release 5.6, SVID, or SVR4 version returns `ERR` on failure and an integer value other than `ERR` upon successful completion.	N	C	C	S
`wdelch`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
wdeleteln(3V)	C	In the SunOS release 5.6, SVID, or SVR4 software this routine may be a macro, while it always is in the SunOS 4.x software. The SunOS release 5.6, SVID, or SVR4 version returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
wdeleteln(3V) – SysV	S		N	S	S	N
wechochar(3V) – SysV	S		N	S	S	N
werase(3V)	C	In the SunOS release 5.6, SVID, or SVR4 software this routine returns OK(0) or a non-negative integer if *immedok* is set. The SunOS release 5.6, SVID, or SVR4 header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h> and if CURS_PERFORMANCE is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
werase(3V) – SysV	S		N	S	S	N
wgetch(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of wgetch() returns ERR on failure and an integer value other than ERR upon successful completion. The SunOS release 5.6, SVID, or SVR4 version also has additional support for function keys.	N	C	C	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
wgetch(3V) – SysV	C	In the SunOS release 5.6, SVID, or SVR4 software, if the window is not a pad, and it has been moved or modified since the last call to wrefresh(), wrefresh() will be called before another character is read. In the SunOS release 4.x software, wrefresh() will not be called under these circumstances.	N	C	C	N
wgetstr(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of wgetstr() returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
wgetstr(3V) – SysV	C	See getstr(3V) — Sys V.	N	C	C	N
winch(3V) – SysV	S		N	S	S	S
winsch(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of winsch() returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
winsch(3V) – SysV	S		N	S	S	N
winsertln(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of winsertln() returns ERR on failure and an integer value other than ERR upon successful completion. This can be a macro in SunOS release 5.6, or the SVID or SVR4.	N	C	C	S

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
winsertln(3V) – SysV	S		N	S	S	N
wmove(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of wmove() returns ERR on failure and an integer value other than ERR upon successful completion.	N	C	C	S
wmove(3V) – SysV	S		N	S	S	N
wnoutrefresh(3V) – SysV	S		N	S	S	N
wprintw(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of wprintw() returns ERR on failure and an integer value other than ERR upon successful completion. The SunOS 4.x version returns void. SunOS release 5.6, or the SVID or SVR4 header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h> and if CURS_PERFORMANCE is defined it defines most commonly used routines as macros for increased performance.	N	C	C	S
wprintw(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`wrefresh`(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of `wrefresh`() returns (`ERR = -1`) on failure and some other integer on success while SunOS 4.x returns `void`. SunOS release 5.6, or the SVID or SVR4 header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>` and if `CURS_PERFORMANCE` is defined it defines most commonly used routines as macros for increased performance.	N	C	C	S
`wrefresh`(3V) – SysV	S		N	S	S	N
`wscanw`(3V)	C	The SunOS release 5.6, SVID, or SVR4 version of `wscanw`() returns an `int` containing the number of fields mapped by the call while the SunOS 4.x version returns `void`. The SunOS release 5.6, SVID, or SVR4 header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>` and if `CURS_PERFORMANCE` is defined, it defines most commonly used routines as macros for increased performance.	N	C	C	S
`wscanw`(3V) – SysV	S		N	S	S	N
`wsetscrreg`(3V) – SysV	S		N	S	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`wstandend`(3V)	C	This is a `curses`(3V) function that clears all window attributes using `attrset(0)`. The SunOS 4.x version always returns `undefined` while the SunOS release 5.6, SVID, or SVR4 `standout`() routine always returns `1` (success).	N	C	C	S
`wstandend`(3V) – SysV	S		N	S	S	N
`wstandout`(3V)	C	This is a `curses`(3V) function that sets the A_STANDOUT attribute to enable the terminals best standout mode for a window. The SunOS 4.x version uses `attron(A_STANDOUT)` for this function and returns `undefined`. The SunOS release 5.6, SVID, or SVR4 `standout`() routine is the same as: `attron(A_STANDOUT)` and always returns `1` (success).	N	C	C	S
`wstandout`(3V) – SysV	S		N	S	S	N
`xcrypt`(3R)	N		N	N	N	N
`xdecrypt`(3R)	N		N	N	N	N
`xdr_accepted_reply`(3N)	S		S	S	S	N
`xdr_array`(3N)	S		S	S	S	N
`xdr_authunix_parms`(3N)	S	This routine is still available but is superseded by `xdr_authsys_parms`(3N).	A	A	A	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
xdr_bool(3N)	S		S	S	S	N
xdr_bytes(3N)	S		S	S	S	N
xdr_callhdr(3N)	S		S	S	S	N
xdr_callmsg(3N)	S		S	S	S	N
xdr_enum(3N)	S		S	S	S	N
xdr_float(3N)	S		S	S	S	N
xdr_free(3N)	S		S	S	S	N
xdr_getpos(3N)	S		S	S	S	N
xdr_inline(3N)	S		S	S	S	N
xdr_int(3N)	S		S	S	S	N
xdr_long(3N)	S		S	S	S	N
xdr_opaque(3N)	S		S	S	S	N
xdr_pointer(3N)	S		S	S	S	N
xdr_reference(3N)	S		S	S	S	N
xdr_setpos(3N)	S		S	S	S	N
xdr_short(3N)	S		S	S	S	N
xdr_string(3N)	S		S	S	S	N
xdr_u_char(3N)	S		S	S	S	N
xdr_u_int(3N)	S		S	N	S	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
xdr_u_long(3N)	S		S	S	S	N
xdr_u_short(3N)	S		S	S	S	N
xdr_union(3N)	S		S	S	S	N
xdr_vector(3N)	S		S	S	S	N
xdr_void(3N)	S		S	S	S	N
xdr_wrapstring(3N)	S		S	S	S	N
xdrmem_create(3N)	S		S	S	S	N
xdrrec_create(3N)	S		S	S	S	N
xdrrec_endofrecord(3N)	S		S	N	S	N
xdrrec_eof(3N)	S		S	S	S	N
xdrrec_skiprecord(3N)	S		S	N	S	N
xdrstdio_create(3N)	S		S	S	S	N
xtom(3X)	S		N	N	N	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
`y0`(3M)	C	In the SunOS release 4.x software, when these routines have undefined results they return `NaN`, with an `EDOM` error and a `DOMAIN` matherr. In the SunOS release 5.6, SVID, or SVR4 software, they return `-HUGE` with an `EDOM` error and a `DOMAIN` matherr. In the SunOS release 4.x software, `y0(HUGE)`, `y1(HUGE)`, `yn(9,HUGE)` will return `zero` with no error indication.	N	C	C	N

TABLE C–1 Library Routines Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6 Status	Notes	ABI	SVID	SVR4	BSD
y1(3M)	C	In the SunOS release 4.x software, when these routines have undefined results they return NaN, with an EDOM error and a DOMAIN matherr. In the SunOS release 5.6, SVID, or SVR4 software, they return -HUGE with an EDOM error and a DOMAIN matherr. In the SunOS release 4.x software, y0(HUGE), y1(HUGE), yn(9,HUGE) will return zero with no error indication.	N	C	C	N
yn(3M)	C	In the SunOS release 4.x software, when these routines have undefined results they return NaN, with an EDOM error and a DOMAIN matherr. In the SunOS release 5.6, SVID, or SVR4 software, they return -HUGE with an EDOM error and a DOMAIN matherr. In the SunOS release 4.x software, y0(HUGE), y1(HUGE), yn(9,HUGE) will return zero with no error indication.	N	C	C	N

APPENDIX **D**

System Files Reference Table

This appendix contains the System Files reference table. This table lists all SunOS 4.x system files, and shows their status in the Solaris 2.6 environment.

Using the Reference Table

- If an interface is listed as “changed” (C), a brief description of differences between the SunOS 4.x and Solaris 2.6 file is provided.
- If an interface is listed as “the same” (S), the Solaris 2.6 interface supports all features of the SunOS 4.x interface. In some cases the interface has been enhanced, but can be considered a complete superset of the SunOS 4.x interface.
- If an interface is listed as “not available” (N), check the Notes section for information about its replacement.

For complete information on all Solaris 2.6 interfaces, see the man Pages(4): File Formats.

System Files

TABLE D–1 File Formats Reference Table

SunOS release 4.x	SunOS release 5.6Status	Notes
`a.out`(5)	C	Assembler and link editor output format
`acct`(5)	S	Execution accounting file
`aliases`(5)	S	Addresses and aliases for sendmail
`ar`(5)	S	Archive (library) file format
`audit.log`(5)	N	Security audit trail file
`audit_control`(5)	N	Control information for system audit daemon
`audit_data`(5)	N	Current information on audit daemon
`auto.home`(5)	C	Automount map for home directories
`auto.master`(5)	C	Automount map for home directories
`bar`(5)	N	Tape archive file format
`boards.pc`(5)	N	ATN and XTN compatible boards for DOS windows
`bootparams`(5)	S	Boot parameter database
`cpio`(5)	S	Format of `cpio` archive
`crontab`(5)	S	Table of times to run periodic jobs
`dir`(5)	A	Format of directories
`dump`(5)	C	Incremental dump format
`environ`(5V)	C	User environment
`ethers`(5)	S	Ethernet address to *hostname* database or NIS domain
`exports`(5)	A	Directories to export to NFS clients
`fbtab`(5)	C	Frame buffer table

TABLE D–1 File Formats Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6Status	Notes
`fcntl`(5)	C	File control options
`fs`(5)	C	Format of a 4.2 (`ufs`) file system volume
`fspec`(5)	S	Format specification in text files
`fstab`(5)	A	Static file system mounting table, mounted file systems table
`ftpusers`(5)	S	List of users prohibited by FTP
`gettytab`(5)	N	Terminal configuration database
`group`(5)	S	Group file
`group.adjunct`(5)	N	Group security data file
`holidays`(5)	C	Prime/non-prime table for System V accounting
`hosts`(5)	S	Host-name database
`hosts.equiv`(5)	S	Trusted hosts by system and by user
`indent.pro`(5)	N	Default options for indent
`inetd.conf`(5)	S	Internet servers database
`internat`(5)	N	Key mapping table for internationalization
`keytables`(5)	S	Keyboard table descriptions for loadkeys and dumpkeys
`link`(5)	N	Link editor interfaces
`locale`(5)		Locale database
`magic`(5)	S	File command's magic number file
`mtab`(5)	A	Mounted file-system table

TABLE D–1 File Formats Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6Status	Notes
`netgroup`(5)	S	List of network groups
`netmasks`(5)	S	Network mask database
`netrc`(5)	S	File for ftp remote login data
`networks`(5)	S	Network name database
`passwd`(5)	C	Password file
`passwd.adjunct`(5)	N	User security data file. See `shadow`(4).
`phones`(5)	S	Remote-host phone number database
`plot`(5)	N	Graphics interface
`printcap`(5)	A	Printer capability database
`proto`(5)	S	Prototype job file for at
`protocols`(5)	S	Protocol name data base
`publickey`(5)	S	Public key database
`queuedefs`(5)	S	Queue description file for at, batch, and cron
`rasterfile`(5)	S	Sun's file format for raster images
`remote`(5)	S	Remote host description file
`resolv.conf`(5)	S	Configuration file for domain-name system resolver
`rfmaster`(5)	N	
`rgb`(5)	N	Available colors (by name) for coloredit
`rhosts`(5)	S	Trusted hosts by system and by user
`rmtab`(5)	S	Remote mounted file-system table

TABLE D–1 File Formats Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6Status	Notes
`rootmenu`(5)	A	Root menu specification for SunView
`rpc`(5)	S	RPC program number database
`sccsfile`(5)	S	Format of an SCCS history file
`services`(5)	S	Internet services and aliases
`sm`(5)	S	`in.statd` directory and file structures
`statmon`(5)	S	`statd` directories and file structures
`sunview`(5)	A	Initialization file for SunView
`svdtab`(5)	N	SunView device table
`syslog.conf`(5)	S	Configuration file for syslogd system log daemon
`systems`(5)	C	NIS systems file
`tar`(5)	S	Tape archive file format
`term`(5)	S	Terminal driving tables for nroff
`term`(5V)	S	Format of compiled term file
`termcap`(5)	S	Terminal capability database
`terminfo`(5V)	S	Terminal capability database
`toc`(5)	N	Table of contents of optional clusters
`translate`(5)	N	Input and output files for system message translation
`ttytab`(5)	N	Terminal initialization data
`types`(5)	S	Primitive system data types
`tzfile`(5)	S	Time zone information

TABLE D–1 File Formats Reference Table *(continued)*

SunOS release 4.x	SunOS release 5.6Status	Notes
`updaters`(5)	S	Configuration file for NIS updating
`utmp`(5V)	C	Login records
`uuencode`(5)	S	Format of an encoded uuencode file
`vfont`(5)	N	Font formats
`vgrindefs`(5)	N	`vgrind`'s language definition database
`xtab`(5)	N	Directories to export to NFS clients
`ypaliases`(5)	N	NIS aliases for sendmail
`ypfiles`(5)	S	NIS database and directory structure
`ypgroup`(5)	N	NIS group file
`yppasswd`(5)	N	NIS password file
`ypprintcap`(5)	N	NIS printer capability database

APPENDIX **E**

/ and /usr File Systems Changes

This Appendix shows the layout of directories in the / and /usr file systems. Appendix A, explains differences in commands within these directories.

Layout of the / File System

Table E–1 shows the layout of the SunOS release 5.6 / file system, which contains directories that are unique to each system.

TABLE E–1 Directories in the / File System

Directory	Description
/	Root of the overall file-system name space
/dev	Primary location for special files
/dev/dsk	Block disk devices
/dev/rdsk	Raw disk devices
/dev/pts	Pseudo terminal slave devices
/dev/rmt	Raw tape devices
/dev/sad	Entry points for the STREAMS Administrative Driver

TABLE E–1 Directories in the / File System *(continued)*

Directory	Description
/dev/term	Terminal devices
/etc	Host-specific system administrative configuration files and databases
/etc/acct	Accounting system configuration information
/etc/cron.d	Configuration information and FIFO for cron
/etc/default	Default information for various programs
/etc/dfs	Configuration information for exported file systems
/etc/fs	Binaries organized by file-system types for operations required before /usr is mounted
/etc/inet	Configuration files for Internet services
/etc/init.d	Scripts for transitioning among run levels
/etc/lib	Shared libraries needed during booting
/etc/lp	Configuration information for the printer subsystem
/etc/mail	Mail subsystem configuration
/etc/net	Configuration information for ti (transport independent) network services
/etc/opt	Configuration information for optional packages
/etc/rc0.d	Scripts for entering or leaving run level 0
/etc/rc1.d	Scripts for entering or leaving run level 1
/etc/rc2.d	Scripts for entering or leaving run level 2
/etc/rc3.d	Scripts for entering or leaving run level 3
/etc/rcS.d	Scripts for entering or leaving run level S

TABLE E–1 Directories in the / File System *(continued)*

Directory	Description
/etc/saf	Service Access Facility (SAF) files, including FIFOs
/etc/skel	Default profile scripts for new user accounts
/etc/sm	Status monitor information
/etc/sm.bak	Backup copy of status monitor information
/etc/tm	Trademark files; contents displayed at boot time
/etc/uucp	Configuration information for uucp
/export	Default root of the exported file-system tree
/home	Default root of a subtree for user directories
/kernel	Subtree of loadable kernel modules, including the base kernel itself as /kernel/unix
/mnt	Temporary mount point for file systems
/opt	Root of a subtree for add-on application packages
/opt/SUNWspro	Mount/installation point for unbundled language products
/platform	Subtree of loadable kernel modules
/sbin	Essential executables used in the booting process and in manual system failure recovery
/tmp	Temporary files; cleared during boot sequence
/usr	Mount point for the /usr file system
/var	Root of a subtree of various files
/var/adm	System logging and accounting files
/var/crash	Default depository for kernel crash dumps
/var/cron	Log file for cron

TABLE E–1 Directories in the / File System *(continued)*

Directory	Description
/var/lp	Line printer subsystem logging information
/var/mail	Directory where users' mail is kept
/var/news	Community service messages (not to be confused with USENET-style news)
/var/nis	NIS+ databases
/var/opt	Root of a subtree for various files associated with optional software packages
/var/options	Provides package compatibility with pre-SunOS 5.0 packages
/var/preserve	Backup files for vi and ex editors
/var/sadm	Databases maintained by the software package management utilities
/var/saf	System Access Facility (SAF) logging and accounting files
/var/spool	Directories for spooled temporary files
/var/spool/cron	Spool files for cron and at
/var/spool/locks	Spooling lock files
/var/spool/lp	Line printer spool files
/var/spool/mqueue	Mail queued for delivery
/var/spool/pkg	Spooled packages
/var/spool/uucp	Queued uucp jobs
/var/spool/uucppublic	Files deposited by uucp
/var/tmp	Directory for temporary files not cleared during boot sequence

TABLE E–1 Directories in the / File System *(continued)*

Directory	Description
/var/uucp	Log and status files for uucp
/var/yp	Databases for yp (for backward compatibility with NIS and ypbind)

Layout of the /usr File System

Table E–2 shows the layout of the /usr file system, which contains architecture-dependent and architecture-independent sharable files.

TABLE E–2 Directories in the /usr File System

Directory	Description
/usr/4lib	Libraries for the binary compatibility a.out package (BCP)
/usr/bin	Location for standard system commands
/usr/bin/sunview1	SunView executables, part of BCP
/usr/ccs	The C compilation system
/usr/ccs/bin	Binaries
/usr/ccs/lib	Libraries and auxiliary files
/usr/demo	Demo programs and data
/usr/games	Game binaries and data
/usr/include	Include header files (for C programs, and the like)
/usr/kernel	Additional modules

TABLE E–2 Directories in the /usr File System *(continued)*

Directory	Description
/usr/kvm	Implementation architecture-specific binaries and libraries
/usr/lib	Various program libraries, architecture-dependent databases, and binaries not invoked directly by the user
/usr/lib/acct	Accounting scripts and binaries
/usr/lib/dict	Database files for the spell command
/usr/lib/class	Scheduling class-specific directories containing executables for priocntl and dispadmin commands
/usr/lib/font	Font description files for troff
/usr/lib/fs	File system type dependent modules; not invoked directly by the user
/usr/lib/iconv	Conversion tables for iconv
/usr/lib/libp	Profiled libraries
/usr/lib/locale	Internationalization and localization databases
/usr/lib/localedef	Locale source file for localedef.
/usr/lib/lp	Line printer subsystem databases and back-end executables
/usr/lib/mail	Auxiliary programs for the mail subsystem
/usr/lib/netsvc	Internet network services
/usr/lib/nfs	Auxiliary NFS-related programs and daemons
/usr/lib/pics	PIC archives needed to build the runtime linker
/usr/lib/refer	Preprocessor for nroff/troff
/usr/lib/sa	Scripts and commands for the system activity report package
/usr/lib/saf	Auxiliary programs and daemons related to the Service Access Facility (SAF)

TABLE E–2 Directories in the `/usr` File System *(continued)*

Directory	Description
`/usr/lib/spell`	Auxiliary `spell`-related programs and databases
`/usr/lib/uucp`	Auxiliary `uucp`-related programs and daemons
`/usr/local`	Commands local to a site
`/usr/net/servers`	Entry points for foreign name-service requests related by the listener
`/usr/oasys`	Files pertaining to the optional FACE package
`/usr/old`	Programs that are being phased out
`/usr/openwin`	Mount or installation point for OpenWindows software
`/usr/sadm`	Various files and directories related to system administration
`/usr/sadm/bin`	Binaries for use by FMLI scripts
`/usr/sadm/install`	Executables and scripts for package management
`/usr/sbin`	Executables for system administration
`/usr/sbin/static`	Statically linked versions of selected programs from `/usr/bin` and `/usr/sbin`; used to recover from broken dynamic linking
`/usr/share`	Architecture-independent databases
`/usr/share/lib`	Architecture-independent databases
`/usr/share/lib/keytables`	Keyboard layout description tables
`/usr/share/lib/mailx`	Help files for `mailx`
`/usr/share/lib/nterm`	Terminal tables for `nroff`
`/usr/share/lib/pub`	Various data files
`/usr/share/lib/spell`	Auxiliary `spell`-related databases and scripts
`/usr/share/lib/tabset`	Tab-setting escape sequences

TABLE E–2 Directories in the `/usr` File System *(continued)*

Directory	Description
`/usr/share/lib/terminfo`	Terminal description files
`/usr/share/lib/tmac`	Macro packages for `nroff` and `troff`
`/usr/share/lib/zoneinfo`	Time zone information
`/usr/share/src`	Source code for kernel, libraries, and utilities
`/usr/snadm`	Files associated with Administration Tool (`admintool`)
`/usr/ucb`	Berkeley compatibility package binaries
`/usr/ucbinclude`	Berkeley compatibility package header files
`/usr/ucblib`	Berkeley compatibility package libraries
`/usr/vmsys`	Files pertaining to the optional FACE package

APPENDIX **F**

Quick Reference for Basic Changes

This appendix is a quick reference for changes in common commands, files and directories, and daemons and standard processes.

Summary Tables

TABLE F–1 Basic Commands

SunOS Release 4.x	Solaris 2.6	Comments
`lpr`	`lp`	Basic default print command
`lpr` –`P` *printer*	`lp` –`d` *printer*	Specifying a printer with the print command
`lpq`	`lpstat` –`o`	Check the print queue of the default printer
`lpq` –`P` *printer*	`lpstat` –`o` *printer*	Check the status of a specific printer and list print IDs
	`lpstat` –`a`	Determine which printers are available (in the SunOS release 4.x software, you would check the `/etc/printcap` file)
`lprm` *print job#*	`cancel` *request ID*	Cancel a print job
	`cancel` *printer*	Alternate method for canceling a currently active print job

TABLE F–1 Basic Commands *(continued)*

SunOS Release 4.x	Solaris 2.6	Comments
`ps –ax`	`ps –ef`	Process status is the same but some of the options have changed
`pstat –s`	`swap –s`	Prints information about swap space

TABLE F–2 Advanced Commands

SunOS Release 4.x	Solaris 2.6	Comments
`dump`	`ufsdump`	For backing up file systems or specified files
`exportfs`	`share` *resources*	Used to make specified resource listed in user's `/etc/dfs/dfstab` available for remote mount
`exportfs –a`	`shareall`	Option to make all resources listed in user's `/etc/dfs/dfstab` available for mounting
`exportfs –u`	`unshare` *resource*	Used to make resources unavailable
`mount –a`	`mountall`	Mount all file systems specified in `/etc/vfstab`, where the mountall option is set
`restore`	`ufsrestore`	For restoring files dumped to backup media
`showmount –d`	`dfmounts` *option*	Lists mounted NFS file systems where option specifies machine name
`showmount –e`	`dfshares` *option*	Lists shared (exported) NFS file systems
`umount –a`	`umountall`	Unmount all file systems in `/etc/vfstab`, other than root, `/proc`, `/var`, and `/usr`

TABLE F–3 Files and Directories

SunOS Release 4.x	Solaris 2.6	Comments
`/var/spool/mail`	`/var/mail`	Location for incoming mail
`/etc/fstab`	`/etc/vfstab`	File system mount table
`/etc/exports`	`/etc/dfs/dfstab`	Lists exported file systems
`/etc/mtab`	`/etc/mnttab`	List of currently mounted resources read by the `/etc/mount` command
`/etc/xtab`	`/etc/dfs/sharetab`	List of shareable resources
`/usr/bin`	`/usr/bin` and `/usr/sbin`	`/usr/sbin` is available with Solaris executables
`/etc/aliases`	`/etc/mail/aliases`	New location for local e-mail alias file
`/etc/printcap`	No longer exists	Capability replaced by `/usr/share/lib/terminfo` and files in `/etc/lp`
`/etc/passwd`	`/etc/passwd` / `etc/shadow`	Capability is shared with counterpart, the `/etc/shadow` file, which stores user's encrypted passwords and other information

TABLE F–4 Daemons and Standard Processes

SunOS Release 4.x	Solaris 2.6	Comments
`/usr/lib/lpd`	`/usr/lib/lp/lpsched`	Print daemon
`/usr/etc/rpc.lockd`	`/usr/lib/nfs/lockd`	Network lock daemon
`/usr/etc/rpc.mountd`	`/usr/lib/nfs/mountd`	NFS mount request server
`/usr/etc/ypbind`	`/usr/lib/netsvc/yp/ypbind`	NIS binder process

TABLE F–4 Daemons and Standard Processes *(continued)*

SunOS Release 4.x	Solaris 2.6	Comments
`/usr/etc/nfsd`	`/usr/lib/nfs/nfsd`	NFS daemon
`/usr/etc/biod`	No longer exists	Block I/O daemon Capability implemented in the kernel
`/etc/rc` and `/etc/rc.local`	`/etc/rc[012356S].d`	System initialization scripts

TABLE F–5 File and Command Differences

SunOS Release 4.x	Solaris 2.6
`ac`	`sar`
`add_services`	`pkgadd`
`arch`	`uname –m`
`bar`	Use `cpio –H` bar to retrieve
`biff –n`	`chmod –o-x /dev/tty`
`biff –y`	`chmod –o+x /dev/tty`
`cc`	Not available
`dbxtool`	`debugger`
`df`	`df –k`
`dketl`	Not available
`dkinfo`	`prtvtoc`
`du`	`du –k`

TABLE F–5 File and Command Differences *(continued)*

SunOS Release 4.x	Solaris 2.6
`dump`	`ufsdump`
`dumpfs`	Not available
`etherfind`	`snoop`
`exportfs`	`share`
`extract_files`	Not available
`extract_patch`	Not available
`extract_unbundled`	`pkgadd`
`fastboot`	`reboot` or `init –6`
`fasthalt`	`init –0`
`hostid`	`sysdef –h`
`hostname`	`uname –n`
`intr`	Not available
`leave`	Use `cron` and `at`
`lint`	Not available
`load`	`pkgadd`
`loadc`	`pkgadd`
`load_package`	Not available
`lpc`	`lpadmin`
`lpd`	`lpsched`

TABLE F–5 File and Command Differences *(continued)*

SunOS Release 4.x	Solaris 2.6
`lpq`	`lpstat`
`lpr`	`lp`
`lprm`	`cancel`
`lptest`	Not available
`mach`	`uname –p`
`modstat`	`mount –a`
`mount`	`mount –F` *fstype* [*options*]
`mountall`	`modinfo`
`mount_tfs`	`mount –F` *fstype*
`pax`	`cpio`
`paxcpio`	`cpio`
`portmap`	`rpcbind`
`printenv`	`env`
`ps –a`	`ps –e`
`ps –aux`	`ps –el`
`pstat`	`sar`
`pstat –s`	`swap –s`
`rdump`	`ufsdump`
`restore`	`ufsrestore`

TABLE F–5 File and Command Differences *(continued)*

SunOS Release 4.x	Solaris 2.6
`rm_client`	`admintool`
`rm_services`	Not available
`rpc.etherd`	Not available
`rpc.lockd`	`lockd`
`rpc.mountd`	`mountd`
`rpc.read`	Not available
`rpc.rquotad`	Not available
`rpc.showfhd`	`showfhd`
`rpc.statd`	`statd`
`rpc.user_agentd`	Not available
`rpc.yppasswdd`	Not available
`rpc.ypupdated`	`ypupdated`
`rrestore`	`ufsrestore`
`rusage`	Not available
`startup`	Not available
`swapon`	`swap –a`
`sys-config`	`admintool`
`umountall`	`umount –a`
`umount-tfs`	`umount –F` *fstype*

TABLE F–5 File and Command Differences *(continued)*

SunOS Release 4.x	Solaris 2.6
`unload`	`pkgrm`
`update`	`fsflush`
`uptime`	`who –b`
`users`	`who –q`
`vipw`	Not available
`wall`	Not available
`whereis`	Not available
`whoami`	`id`
`ypbatchupd`	Not available
`yppasswd`	Use `nispasswd` for NIS+
`ypserv`	Not available

Glossary

Architecture — The specific components of a computer system and the way they interact with one another. From a Solaris 2.6 kernel perspective, “architecture” refers to the type of CPU chip in the system. In this manual, the only architecture discussed is the kernel architecture (for example sun4, sun4c, or sun4m).

Binary Compatibility Package — An optional package that enables existing SunOS release 4.x applications, both statically and dynamically linked, to run under SunOS release 5.6 without modification or recompilation.

Client — A system that uses NIS, NFS, or other services provided by another system.

Cluster — A functional collection of software packages.

Configuration cluster — A default selection of clusters representing typical software selections.

Dataless — A system whose `/usr` and `/usr/kvm` file systems are provided by a file server, and whose root and swap disk partitions are on a directly connected disk.

DDI — Device Driver Interface. Facilitates both source and binary portability across successive releases of the operating system on a particular system.

DKI — Driver Kernel Interface. A defined service interface for the entry point routines and utility functions specified for communication between the driver and the kernel. It does not encompass the driver/hardware or the driver/boot software interface.

Disk partition — See *disk slice*.

Disk slice A discrete portion of a disk, configured during installation. Slices were referred to as partitions under the SunOS 4.1.x and System V Release 3 software.

Diskless A system whose `root`, `swap`, and `/usr` file systems (disk partitions) are provided by an NFS server (or file server) instead of a directly connected disk.

DNS Domain name system. The distributed name/address mechanism used in the Internet.

ELF Executable and linking format. The native object format of Solaris 2.6 executables.

Heterogeneous server A server of diskless clients that is a mix of its own architecture and other kernel architectures.

Homogeneous server A server of diskless clients that has only clients with the same kernel architecture.

Install server A machine that provides boot service and network access to the Solaris 2.6 distribution. This can be on either a local CD-ROM or a file system containing a copy of the distribution.

IP address A unique number that identifies each host in a network. The address is partitioned into two distinct parts: a network part and a host part.

Kernel architecture The hardware portion of a Solaris 2.6 kernel. Two systems have the same kernel architecture if the same Solaris 2.6 kernel runs on both of them. Not all Sun-4 systems have the same kernel architecture.

Multiple OS operation The operation that enables a SPARC server to continue serving SunOS 4.1.x clients while the server is running the Solaris 2.6 release. In this special case, a heterogeneous server could be serving clients of the same kernel architecture.

Netmask A number used by software to separate additional network information (called the “subnet”) from the host part of an IP address. The netmask is also referred to as the subnet mask.

NIS The network information service. NIS provides information about machines and services in a local area network.

NIS+ An enhanced version of the network information service software. These enhancements include secure updates, better performance, and hierarchical naming.

OLIT	Abbreviation for OPEN LOOK Intrinsics Toolkit.
Package	A functional grouping of software. All SunOS release 5.6 software is grouped and distributed in packages. Packages are also the standard way to deliver unbundled Sun and third-party software.
SAC	Services Access Facility. A SunOS release 5.6 tool for managing access to local and network system services, such as modems and terminals
SAF	Service Access Control. Commands used to set up and manage services.
Server	A system that provides services to the network. These services include NFS system and NIS database access.
Source Compatibility Package	An optional package that contains a collection of SunOS release 4.x and BSD commands, library routines, and header files otherwise not available with Solaris 2.6 software.
Standalone	A system that does not depend on a server for its `root`, `swap`, or `/usr` disk partitions.
Time zone	Any of the 24 longitudinal divisions of the earth's surface for which a standard time is kept.
Unbundled	Software products not delivered as part of SunOS™ release 5.6 software distribution: for example, the SunPro compilers.

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z