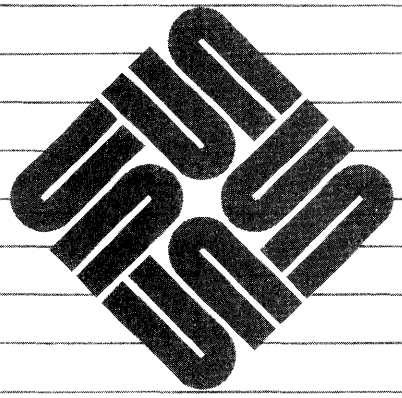
sun®
microsystems

SunOS 4.1 Release Manual



Part Number: 800-3802-10
Revision A of 6 March 1990

The use of this logo certifies SunOS 4.1 conformance with X/Open Portability Guide Issue 2 (XPG 2).
This logo is a trademark of the X/Open Company Limited in the UK and other countries, and its use is licensed to Sun Microsystems, Inc.

Contents

Tables

Figures

1

Introduction/Guide to SunOS Publications

1.1. The SunOS Release 4.1 Documentation - Where to Start

As with every SunOS™ release, the first manual you should read is the *Read This First* (Part Number 800-3845-10) packed in the box that the release 4.1 software CDs, tapes, or diskettes came in. This is especially important for subsequent SunOS releases which use this set of documentation as a base. The *Read This First* for each future release will inform you if new functionality will require modification or replacement of SunOS 4.1 documents.

After the *Read This First*, the *SunOS 4.1 Release Manual* (this manual) is the next one to read. Chapters 3 and 4 (*Special Notes* and *Known Problems*) are particularly important; you should read and understand those chapters before installing or using SunOS release 4.1. After this manual, there are many different routes you can take through the 4.1 documentation. The *Roadmap to the SunOS Release 4.1 Documentation* (Part Number 800-3873-10) is designed to help guide you through. The *Roadmap* is packed in the same box this manual came in.

1.2. Purpose of This Manual

Generally, this manual assumes that you are already familiar with the SunOS and want to find out what software and hardware has been added or improved since SunOS release 4.0. This manual provides concise descriptions of those changes and enhancements in SunOS release 4.1. It does *not* give full details on those enhancements, but describes each briefly and indicates which of the other SunOS release 4.1 documents provides complete details.

1.3. Guide to Publications

Sun Microsystems, Inc. ® provides an extensive collection of software manuals for use with release 4.1. The software manuals are shipped in the following binders (in general order of use):

Media Box: *Read This First*

The *Read This First* is discussed above.

Binder: *SunOS Release and Install*

This binder contains the *SunOS 4.1 Release Manual*, which you are reading now. The most important parts of this manual are the "Special Notes For SunOS Release 4.1" and "Known Problems" chapters. Those chapters *must* be read and understood before using SunOS release 4.1. In addition, this manual contains brief descriptions of the new release 4.1 features, and points to where details can be found.

The *Quick Install Guide* describes a fast new utility for booting a standalone system. The *Installing SunOS 4.1* manual contains detailed descriptions of how to install release 4.1 on all systems.

Books: *User's Guides*

These brief manuals are easy to read, written for the user with little or no UNIX† experience. *Getting Started* is for those new to the Sun environment. Other titles are *Doing More with SunOS, Basic Troubleshooting, SunView User's Guide,* and *Customizing Your Environment.* The *SunDiag User's Guide* is in the same box, but is written for users who will be testing memory, drives, boards, and board-level devices with SunDiag. It is the only User's Guide that is strictly for advanced users.

Binder: *System and Network Administration*

The *System and Network Administration* manual is written for system administrators, but is a valuable resource for *all* SunOS users. It deals with adding hardware, disk maintenance, networking and electronic mail service, and advanced UNIX administration.

Binders: *SunOS Reference Manual* **(3 Volumes)**

This set of binders contains the SunOS release version of the Berkeley UNIX "Manual Pages," alphabetically arranged descriptions of commands, functions and other aspects of SunOS release 4.1. These manual pages are divided by numbered tabs into a range of subjects listed below:

1. User Commands
2. System calls and error numbers
3. User-level library functions
4. Devices, drivers, protocols and network interfaces
5. File formats used or read by various programs
6. Games and demos
7. Public files, tables and TROFF macros
8. System maintenance and operations commands

Binder: *Global Index*

Provides an index to all SunOS release 4.1 software documentation (except this manual and the *Read This First*).

Binder: *SunOS Documentation Tools*

Editing Text Files covers the editors *vi, ex, ed,* and *sed. Formatting Documents* explains special formatting macros that work with TROFF such as *refer,* indexing, table formatting and equation setting. *Using TROFF and NROFF* explains the use of those text processing utilities.

† UNIX is a registered trademark of AT&T.

Binder: *Programmers Guides*

This group of manuals focuses on programming within the Sun environment, for both systems and applications level interests.

C Programmers Guide describes how to write C programs that interface with SunOS. *Assembly Language Reference Manual for Sun-3* and *Assembly Language Reference Manual for Sun-4* covers syntax and usage of the assembler for some of the microprocessors used in Sun workstations and servers. *A RISC Tutorial* looks at the aspects of RISC and open systems architectures as they pertain to SPARC architecture. *Porting Software to SPARC Systems* briefly describes machine level SPARC architecture, and porting C, FORTRAN, and Pascal programs to a SPARC system. The *Debugging Tools Manual* describes the debuggers `dbx`, `dbxtool`, and `adb` for experienced programmers.

Binder: *Programmer's Overview Utilities and Libraries*

The *System Services Overview* contains details of various specialized aspects of SunOS operating system, including internationalization, security features, networking, and UNIX standards compatibility. *Programming Utilities and Libraries* is written primarily for applications programmers to provide an overview of the Sun environment, and the system facilities, utilities, and libraries supported.

Binder: *Network Programming Guide*

The *Network Programming Guide* provides an overview of NFS, pipes, sockets, network commands, Sun on-line database service, and network managers and monitors.

Binder: *Writing Device Drivers/STREAMS Programming*

Writing Device Drivers is a guide to adding drivers for new hardware devices to the SunOS kernel. The *STREAMS Programming* manual covers the theory of STREAMS programming, the SunOS-specific implementation, and catalogs STREAMS functions and data structures.

Binder: *SunView Programmer's Guide*

SunView 1 Programmer's Guide is written for applications programmers to support interactive, graphics-based applications running within windows. The *SunView 1.80 Update Appendix* updates the *SunView 1 Programmer's Guide* with the latest information.

Binder: *SunView 1 System Programmer's Guide*

The *SunView 1 System Programmer's Guide* describes how SunView works from the inside, and how to structure applications. The *Pixrect Reference Manual* describes the *Pixrect* graphics library routines that manipulate arrays of pixel values, and *RasterOps* used by applications programs to manipulate bit-mapped displays.

NOTE: Documentation of security features has been taken out of the *Security Features Guide* and divided among several other manuals. See the next chapter for details.

2

Documentation Conventions

Sun Workstation Architecture Notation

The Sun family of workstations is divided into *Kernel Architectures* and *Application Architectures*. Both application and kernel architectures are returned by the `arch`(1) command in lower case, without a space between the "sun" and the architecture family number (for example `sun4`; see the table below). In this manual, application architectures are noted in the same font as most of the words in this manual (for example, "Sun-4"). The more specialized kernel architectures are noted as they would be returned by the `arch` command (for example, `sun4c`).

The table below lists all of the Sun Workstations and their kernel and application architectures. See the `arch`(1) manual page for details.

Application architecture	Kernel architecture	Current Sun System Models
sun2*	sun2	2/50, 2/120
sun3	sun3	3/50, 3/60, 3/75, 3/110, 3/140, 3/150, 3/160, 3/180, 3/260, 3/280
sun3	sun3x	3/80, 3/460, 3/470, 3/480
sun4	sun4	4/110, 4/150, 4/260, 4/280, 4/390, SPARCsystem 330/370 SPARCsystem 430/470
sun4	sun4c	SPARCstation 1
sun386*	sun386	386*i*/150, 386*i*/250
***Not Supported in Release 4.1**		

Other Documentation Conventions

- System commands, messages, and SunOS release filenames appear in `listing font like this`.
- Information you type in response to the system is shown in **`bold listing font like this`**.

- Information you type that differs between users and systems is shown in *italics like this*. For example:

```
hostname% chmod +w filename
```

  where *filename* is the name of a file you want to give write permission to. These variables often are in tables in the text, where you select the correct entry for your system.

- Titles of chapters of this document are listed in plain Roman font, inside quotation marks like this: "The SPARCstation 1 Desktop Workstation."
- Document titles are listed in plain, nonbold, *italic* font.
- Dialogues between you and the system are enclosed in gray boxes, like this command, to remove a file and the system request for confirmation:

```
hostname% rm -i filename
rm: remove filename? y
```

- Non-interactive sections of program code and system messages are shown in plain listing font, enclosed in clear boxes:

```
int test[100];
main()
{
    register int a, b, c, d, e, f;

    test[a] = b & test[c & 0x1] & test[d & 0x1];
}
```

- The following notation is used throughout this manual for reference to manual pages for commands and utilities:

  passwd(1)

  This notation indicates the `passwd` manual page in section 1 of the *SunOS Reference Manual* (Part Number 800-3827-10).

- Magnetic disks are referred to with the usual spelling of that term, "disk". CDs are referred to with the spelling of "disc" that is traditional for the optical disc industry.

3

Special Notes For SunOS Release 4.1

3.1. Purpose of This Chapter

This chapter contains notes on SunOS release 4.1 features that require special attention.

3.2. Encrypting Files in SunOS release 4.1

File encryption software is *only* available with the *Encryption Kit* option in SunOS release 4.1 (available in the U.S. only). The kits contain a single tape (or 3 diskettes) with the code necessary for encrypting files on all Sun architectures. It must be installed to encrypt files under SunOS release 4.1 **and to unencrypt files that have been encrypted under any other SunOS release.** See the *READ THIS FIRST SunOS 4.1 U.S. Encryption Kit* (Part Number 800-3458-10) for installation instructions.

3.3. Moving Disk Drives From SunOS 4.1 Systems to Systems Running Earlier Releases

The SunOS filesystem format has changed in release 4.1. This change raises an issue when moving a disk drive with a filesystem created under SunOS 4.1, to a system running an earlier SunOS release. The recommended method of avoiding a compatibility problem is to save the filesystem on the drive before disconnecting it using `dump`(8). Run `newfs`(8) on the partitions after the drive has been connected to the new system and then restore the files using `restore`(8).

Alternatively, the `-c` option to `fsck`(8) can be used to convert a filesystem to the older format, eliminating the need for `dump` and `restore`. `fsck -c` will not work for all 4.1 filesystems, however. For example, a release 4.1 filesystem can be created with more than 2048 inodes per cylinder group. A filesystem created with greater than 2048 inodes cannot be converted with `fsck -c`, since the maximum is 2048 in earlier releases. `dump`, `newfs`, and `restore` would have to be used.

`fsck` will alert you if it cannot convert the filesystem. There is no problem moving an older file system to 4.1 because the changes are upwardly compatible.

Problem Installing SunDials/SunButtons on a SPARCstation 1 Solved (bugids 1020212, 1022317)

In releases previous to SunOS release 4.1, installing the SunButtons or SunDials graphics manipulation devices on a SPARCstation 1 failed. `/usr/etc/dbconfig` of `/dev/dialbox` consistently caused the workstations to panic with a data fault. This problem has been fixed.

3.4. "Yellow Pages" Has Been Renamed "Network Information Service" (NIS)

The Network Information Service (NIS) was formerly known as Sun Yellow Pages. The functionality of the two remains the same, only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications (plc) and may not be used without permission.

3.5. Additional 386i SunOS Release 4.0.*X* Specific Information in the *SunOS Reference Manual*

The *SunOS Reference Manual* contains information specific to the 386*i* SunOS 4.0.*X* (or earlier) Releases. An example of this information is the `login-tool`(8) man page.

SunOS release 4.1 does not support the Sun 386*i* platform; these pages were included to provide users with a more convenient and centralized desktop reference set. This information is clearly marked in the **Availability** sections of the effected man pages as follows:

386i SunOS 4.0.X based releases only. Not a feature of SunOS 4.1

3.6. Unbundled Product Compatibility

Some of Sun's products which are *not* sold with SunOS release 4.1 ("unbundled products") require special instructions to to function seamlessly with this release. Those instructions are explained in appendix A of this manual

Use of NSE 1.2 on Release 4.1 is Not Supported

Installation of NSE on a system running 4.1 will cause some system files to be corrupted and may cause a system failure. It's use is not supported on release 4.1; the problem will be corrected in the next revision of NSE.

The Userid of `nobody` is Now `65534`

New in SunOS release 4.1, the `passwd`(1) userid of `nobody` and `nogroup` has changed from –2 to `65534`. The `nobody` entry in the `passwd` file is the following:

```
nobody:*:65534:65534::/:
```

This change was made to conform to the POSIX standard, which requires that the userid of `nobody` be an unsigned interger. The internal binary representation of –2 and `65534` is the same.

NOTE: This change *must* be made to pre-existing `passwd` and group files to maintain functionality.

3.7. Early Shipment of SPARCstation 1s May Not Work on Release 4.1

A few early SPARCstation 1s with CPU boards with numbers below 501-1382-10 will not be able to run SunOS release 4.1. If your SPARCstation 1 has a CPU board number below 501-1382-10, ran 4.0.3 - 4c without problem, but fails with a terminal error (`Bus Error` or `Alignment Error`) when attempting to run SunOS 4.1, you probably need the board replaced. If you workstation fits that criteria, call your local Sun sales representative and reference FCO number 807-0072.

3.8. Security Features Documentation Has Been Revised

Documentation of security features has been reorganized in SunOS release 4.1. Security feature discussions have been taken out of the *Security Features Guide* and incorporated into the appropriate sections of the *System and Network Administration, System Services Overview* and *Network Programming* manuals as explained below.

System and Network Administration

Chapter 7 - Administering Security
This chapter contains information for everyone interested in security issues. It starts with a discussion of file protection for users, and then explains system and network protection for system administrators.

Chapter 13 - The SunOS Network Environment
The **Security in a TCP/IP Environment** section provides a discussion of how to allow or deny remote access by other users.

Chapter 14 - The Sun Network File System Service
The **Secure NFS** and following sections explain what secure NFS is and how to administer it.

Chapter 19 - Administering C2 Security
Explains the C2 security standard, and how to set up and administer C2 security. This discussion highlights password safeguarding and auditing security-related events.

System Services Overview

Chapter 5 - Programmers Guide to Security Features
A discussion of system calls, C libraries, and general information on writing secure C programs.

Network Programming Guide

Chapter 1 - Network Services
This chapter talks about RFS security features. There is some discussion on programming, but the emphasis is on utilities.

Chapter 3 - RPC Programming Guide
Contains a brief discussion specifying several routines which provide better security than standard routines.

3.9. Window Security Changes

Starting in release 4.1, ownership of window device files has been given to individual users. Users now control those window device files as they do any other UNIX file. This includes the ability to grant or withhold access permission to their window device files at any time.

As shipped, device security is *disabled* to prevent compatibility problems with earlier SunOS releases and unbundled products. In order to reduce security risks however, Sun recommends that window security be enabled at the earliest opportunity and thereafter left enabled.

To enable window security, become superuser, and simply remove the comment ("#") symbols in front of the entries in the framebuffer table, `/etc/fbtab`, and the SunView Device table, `/etc/svdtab` (examples of those tables are below). Window security will be enabled the next time you log in.

See the `fbtab`(5) and `svdtab`(5) manual pages for details.

NOTE: Sunview releases ownership of the `win` devices upon normal termination. As such, users running two or more copies of SunView on two or more framebuffers (that is, users of systems with multiple monitors but only one keyboard) will release ownership of the `win` devices if they quit any copy of SunView. If the user wishes to work normally on any remaining copy of SunView, the `win` devices should be re-acquired. This is a simple matter of typing `sv_acquire` on any available command line. If no command line is available (that is, no `cmdtool` or `shelltool` is running), the user may alternatively restart any of the remaining copies of SunView (which will re-acquire the `win` devices).

Figure 3-1 *A Framebuffer Table File*

```
# @(#)fbtab 1.3 90/01/25 SMI
#
# Copyright (c) 1989 by Sun Microsystems, Inc.
#
# /etc/fbtab -- framebuffer table
#
# Description:
# If the user is logging in on a device specified in the "console" field
# of any entry in this file, the owner/group of the devices listed in the
# "device_list" field will be set to that of the user.  Similarly, the mode
# will be set to the mode specified in the "mode" field.
#
# Format:
# console       mode    device_list
#
# Notes:
# A "device_list" is a colon-separated list of device names.
# A '#' begins a comment and may appear anywhere in an entry.
#
# Example:
# /dev/console  0600    /dev/kbd:/dev/mouse
#
 (Uncomment the following lines to enable window security.)

#/dev/console    0600    /dev/kbd:/dev/mouse
#/dev/console    0600    /dev/fb:/dev/bwone0:/dev/bwtwo0
#/dev/console    0600    /dev/cgone0:/dev/cgtwo0:/dev/cgthree0:/dev/cgfour0
#/dev/console    0600    /dev/cgsix0:/dev/cgeight0:/dev/cgnine0
#/dev/console    0600    /dev/gpone0a:/dev/gpone0b:/dev/gpone0c:/dev/gpone0d
#
```

Figure 3-2 *A SunView Device Table file*

```
#
# @(#)svdtab 1.3 90/01/25 SMI
#
# Copyright (c) 1989 by Sun Microsystems, Inc.
#
# /etc/svdtab -- SunView device table
#
# Description:
# When the user starts up SunView, the owner/group/mode of win devices
# will be set to uid/gid/<mode in this file>.
#
# Format:
# mode
# A '#' begins a comment and may appear anywhere in an entry.
#
# Example:
    (Uncomment this line to enable window security.)
# 0600
#
0600
```

3.10. PROM Issues with SunOS Release 4.1

General PROM Issues

See the table 3-1 below for a listing of the minimum PROM levels necessary to run SunOS release 4.1. If your system does not have a 3.0 PROM, then you may not have the latest proms. The 3.0 release has been greatly enhanced, and is recommended for running SunOS release 4.1, but not required (see the table 3-1 below).

NOTE: PROM level 3.0 is not available on SPARCstation 1s or Sun-3/50s. See table 3-1 below for the correct revision levels for those machines.

If an upgrade is needed, you have several options.

a. You can have Sun install the PROM for you. (The upgrade is free if you have an On-Site Hardware or Comprehensive Support Contract.)

b. You can order an upgrade kit, and install the PROM yourself. The kit explains how to replace the boot PROM on your CPU board, a process which takes ten or fifteen minutes.

Call 800-USA-4SUN, request Field Service, and schedule PROM installation or order a Sun-3 PROM Upgrade Kit. (Outside the USA, call your local support office.)

Finding Your Present PROM Level

To find the present PROM level of your workstation (except SPARCstation 1s), type the following at the monitor prompt (">"):

```
> kb
```

If you have a SPARCstation 1, use the following commands to find your present PROM level:

```
> new mode
  ok .version
```

Table 3-1 *Minimum PROM Revision Levels*

Workstations supported in SunOS Release 4.1	Minimum PROM Level
Sun-3/50	1.8
Sun-3/60	1.6
Sun-3/75	1.8
Sun-3/80	2.9
Sun-3/110	1.8
Sun-3/140	1.8
Sun-3/150	1.8
Sun-3/160	1.8
Sun-3/180	1.8
Sun-3/260	1.8
Sun-3/280	1.8
Sun-3/460	3.0
Sun-3/470	2.9
Sun-3/480	2.9
Sun-4/110	2.8
Sun-4/150	2.8
Sun-4/260	2.8
Sun-4/280	2.8
Sun SPARCsystem 330	3.0
Sun SPARCsystem 370	3.0
Sun SPARCsystem 430	0.2.C
Sun SPARCsystem 470	0.2.C
Sun SPARCstation 1	1.0; version 100

NOTE Table 3-2 supersedes this table for some systems and hardware configurations.

Table 3-2 *Minimum PROM Revisions for Special Configurations*

PROM Level	*Hardware Configuration*
2.7	You use a 7053 VME/SMD disk controller with one of the following workstations: Sun-3/75, Sun-3/140, Sun-3/150, Sun-3/160, Sun-3/180, Sun-3/260, or Sun-3/280.
2.9	Your system has a P4-bus graphics board, such as the CG6 or the CG8.

Additional PROM Issues

CD-ROM and SunIPI Support

If SCSI CD-ROM or SunIPI support is needed on a Sun SPARCsystem, then the following PROM levels must be running:

SPARCsystem 330/370.....Use 3.0.3 or greater.

SPARCsystem 430/470..... Use 3.0 or greater.

Loading SunOS 4.1 (or later) on a Sun-3 from 1/4″ tape

A few older Sun-3 systems fail to boot or return errors when you try to load SunOS release 4.1 (or later) from a 1/4″ tape drive. If this happens, follow these steps.

1. To check the revision level of the PROM, enter **kb** at the PROM monitor prompt (>).

   In response, the system displays a series of messages, for example:

```
> kb
...

---------------------------------------------
    Sun Workstation, Model Sun-3/80 Series.
    ROM Rev 3.0, 12MB memory installed, Serial #128.
    Ethernet address 8:0:20:6:33:84, Host ID 42000080.
---------------------------------------------
and so on ...
```

2. Compare the PROM revision number (`3.0` in the example) with that listed for your architecture in Tables 2-2 and 2-3. If the number is less than the minimum level shown, upgrade your PROM as described in the "General PROM Issues" section above. If your PROM level is listed as adequate for running SunOS release in the tables above and you still cannot boot successfully, see the "Miscellaneous PROM Issues" below. See also the *PROM User's Guide.*

Miscellaneous PROM Issues

- See the "Problems with International Type-4 Keyboards" section in chapter 4 for limitations on using Type-4 Keyboards on SPARCstation 1s.
- If the `mt` boot device support is needed, use PROM revision levels below 3.0.
- If the revision of your PROM is between `1.8` and `2.6` and you still cannot boot from 1/4″ tape, the problem could be caused by one of the anomalies described here:
- If your system contains a Sysgen controller board and a Wangtek tape drive, and if the tape you are trying to load is *write-protected*, you will receive an error message with status *96A0*.

  To boot, remove the tape from the drive, write-enable the tape by turning the write-protect key so it points *away* from SAFE, mount the tape in the drive, and boot again. Once the initial boot completes, protect the tape again by turning the write-protect key to SAFE.

  If the boot still fails, contact Hardware Support at 1-800-USA-4SUN. (Outside the USA, call your local support office.)

  Reference FA 136 when you call.
- If you have a Sysgen controller and an Archive tape drive and your PROM revision level is between `1.8` and `2.6`, call Sun as described earlier to request the latest revision PROM.

  Reference FA 135 when you call.
- If you have a Sun-3 workstation with a Sun-2 Mass Storage Subsystem and receive a tape error, call Sun as described earlier.

  Reference FA 137 and 138 when you call.

4

Known Problems with SunOS Release 4.1

4.1. Purpose of This Chapter

This chapter contains information that you should know and understand before using SunOS release 4.1. It is divided into the following sections:

- *Known Problems With SunOS Release 4.1 Products*, and
- *Documentation Corrections and Omissions*

If a bug reference number exists and is known, it is added to the title of the section.

4.2. Known Problems With SunOS Release 4.1 Products

The following is a list of problems which affect SunOS release 4.1.

The `sun3` Kernel Architectures Will Not Support Four Ethernet Connections

The Sun workstations with `sun3` kernel architectures will not automatically support four Ethernet connections. Support for four connections can be added manually as follows:

1. Create `/etc/hostname.ie2` and `/etc/hostname.ie3` with the correct hostname for `ie2` and `ie3`.
2. Put the internet addresses for hostname for `ie2` and `ie3` in `/etc/hosts`.
3. Reconfigure a new kernel with the following changes in the kernel template:

Comment out the `ie0` line (with a "#") as has been done in the box below:

```
#
#Support for the Sun-3/E Intel Ethernet board for 3E cpu systems.
#
#device          ie0 at vme24d16 ? csr 0x31ff02 priority 3 vector ieintr 0x74
```

and remove the comment ("#") from the `ie2` and `ie3` lines as has been done in the box below:

```
#
# Support for additional "Sun-3/E SCSI/Ethernet" boards
#
device          ie2 at vme24d16 ? csr 0x31ff02 priority 3 vector ieintr 0x76
device          ie3 at vme24d16 ? csr 0x35ff02 priority 3 vector ieintr 0x77
```

`uucp` Password File Problem with `/etc/rc` (1032757)

The default `passwd` entry for `uucp` is as follows:

```
uucp:*:4:8::/var/spool/uucppublic:
```

When adding `uucp` logins to the `passwd` file, it is common to make the login shell be `uucico`, which makes that entry look like the following:

```
uucp:*:4:8::/var/spool/uucppublic:/usr/lib/uucp/uucico
```

That only becomes a problem when it interacts with `/etc/rc`. The default `/etc/rc` has the following line which is executed upon bootup:

```
su uucp -c /usr/lib/uucp/uusched &
```

Normally, the `-c` option of `su` is passed (along with the rest of the command line) to the shell. But in this special case the shell is no longer `/bin/sh` (which is the default if none is specified) but `uucico`. Since `uucico` doesn't have a `-c` option, it fails and you get the following error:

```
su: uucico: illegal option -- c
        usage: uucico [-xNUM] [-r[0|1]] -sSYSTEM -uUSERID -dSPOOL -iINTERFACE
```

If you get this error at bootup, you should remove the `uucico` field from the `uucp`'s `passwd` entry.

SPARC Assembler May Optimize The Test Out of a Loop (1031879)

The following C program illustrates a construct that causes the SPARC assembler to optimize the test out of a while loop when compiled with `cc -O` (same as `cc -O2`).

```
int boothowto = 1;

int
main()
{
    int unit;

    if (boothowto & 1) {
retry:
        unit = -1;
        while (unit == -1) {
            if (unit != -1) {
                printf("unit = %d when it should be -1!0,
                  unit);
                exit(1);
            }
            unit = 0;
            print_unit(&unit);
        }
    } else {
        unit = 0;
        goto retry;
    }
}

print_unit(unitp)
    int *unitp;
{
    printf("print_unit: unit = %d0, *unitp);
}
```

To avoid this problem, compile with `-O~M` specified to the assembler, ie.

as [normal options] -O~M

if running the assembler directly, or

cc [normal options] -Qoption as -O~M

if assembling as part of any high-level language compile.

SunCD Software `eject`(1) May Not Work on a Read Failure (1033102)

If the SunCD driver fails to read a particular CD (for example, if the disc is defective), the software `eject` command may also fail, returning the following error message:

```
caffene# eject cd
eject: Open fail on cd -> /dev/rsr0: I/O error
caffene#
```

Power-cycling the SunCD May Cause a Filesystem Mount Failure (1033299)

If the SunCD drive is powered on after the Sun workstation is booted, the CD-ROM file system *might* return the following message and deny the mount request.

```
caffene# mount -rt hsfs /dev/sr0 /cdrom
mount_hsfs: /dev/sr0 on /cdrom: File too long
mount: giving up on:
    /cdrom
caffene#
```

This will also deny the user access to files on the disc. Should this happen, reboot the system with the CD-ROM drive powered on. To avoid this potential problem, *never power cycle the drive while the system is running.*

Conditions to Avoid when Using the SunCD (1033100, 1032990)

The following conditions should be avoided when using a SunCD; they may cause a system hang:

- Accessing a defective, "bad" (with recoverable recording errors), or non-HSFS disc on the CD-ROM drive.
- The CD-ROM disc is ejected while the filesystem is still mounted.
- The SunCD drive is turned off or power-cycled while the filesystem is still mounted.

Systems With Both 1/2-inch and 1/4-inch Tape Drives Must Install Software With Only One (1029124)

If your system has both 1/2-inch and 1/4-inch tape drives, only install software during SunInstall from *one* of the drives. If you do use both, you *must* use the 1/2-inch drive last for the installation to be successful.

Sun-3/80 Users Must Reset the Date After Powering Up (1033299)

Every time a Sun-3/80 running SunOS release 4.1 is shut off and then powered up again, the date must be reset. For example

```
water% date 09210045
```

will set the date to September 21, 12:45 AM. See the `date`(1V) man page for details.

TFTPBOOT links incorrect for CLASS A Networks (1032448)

A server on a CLASS A network will have improper entries for clients in the `/tftpboot` directory. The leading `0` for the name of the links is omitted. To correct this, the same links have to be made with a leading `0` after the clients have been installed (either with SunInstall or `add_client`(8). All of the entries should have 8 character Internet addresses.

An example for a `sun3x` on a CLASS A network. The original entries:

```
190722A -> boot.sun3x.sunos.4.1
190722A.sun3x -> boot.sun3x.sunos.4.1
```

must be changed to

```
0190722A -> boot.sun3x.sunos.4.1
0190722A.sun3x -> boot.sun3x.sunos.4.1
```

Note that once the `0` has been added, the Internet address (`0190722A`) has 8 characters.

add_services(8) Requires 350 KB in /usr (1032894)

In order to add a new release with `add_services`(8), the `/usr` partition must have at least 350 KB of available space.

The Manual Software Category Could be Loaded Twice (1032987)

If the `Manual` software category (on-line manual pages) is selected for two different application architectures, SunInstall will load that category twice.

This is usually not a problem. If there is adequate space, SunInstall merely copies the same data twice. However, SunInstall does allocate space for the second manual set. If SunInstall does not think that there is enough space to hold the manual, there will be an error message. Ignore the message and hit the return key. Since this set of manuals overwrites what was loaded the first time it does not really need any additional space

add_client -i Will Not Create a Client if a Previous add_client -i has Failed (1033185)

After correcting the problems that caused `add_client -i` to fail, rerunning `add_client` with the `-i` option will not create the clients desired. After the original `add_client -i`, has been run, you must use `rm_client`(8) to delete the clients created. Only then can clients be successfully created.

SunInstall Only Accepts the dd/mm/yy Format for Setting the Date (1029073)

SunInstall will only accept two-digit day, month, and year abbreviations in this form: dd/mm/yy. SunInstall claims that anything other than this two-digit format is invalid.

`ypinit` on Slave Server Reports `RPC Program Not Registered` Messages (1029284)

This can be avoided with the following steps. Before running `ypinit -s` *<ypmaster>* on a `ypslave` server, start the `ypxfrd` daemon on the master by logging into the master as `root` and typing `ypxfrd`.

In either case the transfer of maps will occur, but it is less confusing with `ypxfrd`.

You can also start the `ypxfrd` automatically on your `ypmaster` at boot-time, by editing `/etc/rc.local` as follows:

```
if [ -f /usr/etc/ypserv -a -d /var/yp/`domainname` ]; then
    ypserv;                  echo -n ' ypserv'
    ypxfrd;          echo -n ' ypxfrd'
fi
```

RFS Mounts on Directories with Inodes Greater Than 64k Will Fail returning `getwd fails: cannot find`

RFS will truncate inode numbers of 64K or greater. If you plan on advertising RFS resources, be sure that the filesystem to be mounted is not capable of creating inode numbers of 65536 or greater. The maximum possible inode number for a filesystem can be found by executing `df -i` and adding the `iused` and `ifree` values for the filesystem.

Problems With International Type-4 Keyboards

Support for the international variations of the Type-4 keyboard is supplied in the SunOS release 4.1 kernel. The current versions of the monitor PROMs on all Sun workstations do not initially understand the layout of the non-U.S. keyboards. They are understood after the first bootup of the new SunOS release 4.1 kernel; the kernel manages the reprogramming of the EEPROM keytable layouts. Under certain circumstances, however, the international keyboards are treated as if they are U.S. Keyboards (some keys appear to be in the wrong position on the keyboard). This problem occurs in the following circumstances:

- *When a non-U.S. keyboard is connected to a diskless workstation prior to booting from the server.* When booting from a server (single user or fully), the kernel is what reprograms the monitor EEPROM to understand the international (variant) keyboard. Therefore the first time such a workstation is booted, the monitor will think it has a U.S keyboard layout. Once the system has booted single user, and for all subsequent power-ups and re-boots, the EEPROM keyboard table will understand all non-U.S. layouts. This does not apply to SPARCstation PROMs of rev1.2 or higher (see next item).
- *Prior to rev1.2 of the SPARCstation PROM, the PROM monitor cannot be automatically changed (by `loadkeys`(1)) to understand the layouts of any non-US keyboards.* Rev 1.2 of the SPARCstation PROM will understand all of the non-US keyboard layouts from power-up. To determine the rev number of a SPARCstation 1 PROM, enter the following from the monitor prompt:

```
> new mode
ok .version
```

- *Non-SPARCstation 1 monitor PROMs prior to revision 2.6 do not understand the international Type-4 keyboard layouts.* Once SunOS release 4.1 is booted, full support is obtained. Again, this will not be a problem during operation of the keyboard through the kernel. If you have a PROM with an earlier revision of SunOS and you wish your Sun-4 or Sun-3 monitor PROM to support the keyboards, you will need a PROM upgrade.

See *Appendix A* for information on using the Type-4 keyboard with Sun's unbundled products.

`si1` Does Not Work on SunOS Release 4.1 (1020374)

`si1` (a second SCSI-3 host adapter), does not work on SunOS release 4.1.

SPARCstation 1 Floppy `overrun/underrun` Errors Mean the System is Too Busy to Cope With the Floppy Disk

If the system is very heavily loaded and you attempt to read or write to the floppy disk, you may receive error messages telling you the floppy experienced `overrun` or `underrun`. This means the system is so busy it cannot service the floppy drive's read or write requests in time, and the operation failed. To avoid this problem, use the floppy with a lighter system load.

The floppy driver will automatically retry the failed operation, but will give up completely after a number of attempts, print an error message in the console, and return a failing status code to the calling process. If this happens, the read or write did not succeed and you must retry the original operation when the system is less heavily loaded.

Multiple `shmget(2)` ENOMEM Failures Cause System Panic

If `shmget()` fails with ENOMEM, it leaves behind a partially initialized data structure. A subsequent call to `shmget()` using the same key will return a shared memory identifier which refers to that data structure, and a subsequent call to `shmat()` or `shmctl()` using that identifier may panic the system. In SunOS release 4.1, the problem can be avoided only by not reusing a key for which `shmget()` returned ENOMEM (until the system is rebooted).

SunDiag 2.0 Generates Spurious `WARNING` Message During Startup on Systems Having a Mono FB

SunDiag has been enhanced with a new mono FB test which requires `/dev/bwtwo0` to run if mono FB is not the primary console (`/dev/fb`). If `bwtwo0` is not running, the following error message will be returned:

```
probe WARNING: no /dev/bwtwo0 file
```

If mono FB is the primary console, this message can be safely ignored. Otherwise, user should restart SunDiag using the `-m` option, which will instruct SunDiag to create missing device files. Alternatively, `MAKEDEV bwtwo0` can be run before starting the mono FB test.

RFS Domains With Secondary Name Servers (1028779)

If your RFS domain will have secondary name servers, there is a problem with starting the primary only. It will "time-out" waiting for a name server signal. A workaround for this is to start RFS in this sequence:

1. Run `dorfs start` on the primary first.
2. Run `dorfs start` on a secondary within 1 minute of starting the primary. They will both prompt for a password. Enter the password (if any) for the primary first.
3. Run `dorfs start` on any other machine in this domain.

4.3. Documentation Corrections and Omissions

Online `skyversion`(8) Manual Page Should be Ignored

The printed version of the `skyversion`(8) Reference Manual page is specific to Sun-2 systems, which are not supported in this SunOS release. The on-line manual page has been removed to reflect the correct state.

Incorrect Description in the *System and Network Administration* Manual

There is an incorrect error message and description in Table 21-6 on page 687 of the *System and Network Administration* (Part Number 800-3805-10). The incorrect message is as follows:

```
No uucp server        A TCP/IP call is attempted,
                      but there is no server for UUCP.
```

The description should read as follows:

```
No uucp service number  No entry for uucp/tcp can be found in
                        /etc/services, but there is a hyphen
                        in a Systems file entry port field
```

`rmount`(2) Discussion Omitted from the *System Services Overview*

The following discussion was inadvertently omitted from the *System Services Overview* manual.

AT&T System V Release 3 RFS has an rmount system call, `rmount`(2), that is separate from the mount system call used to mount local filesystems, `mount`(2). SunOS 4.1 does not contain the `rmount`(2) system call.

In Addition, System V has a shell script, `/etc/rmount`(1), which will mount all RFS resources in one command. SunOS release 4.1 does not have this utility either, but can duplicate it with the following command: `mount -vat rfs`.

5

Compatibility and Performance Issues

5.1. Purpose of This Chapter

The purpose of this chapter is threefold: First, to address the compatibility issues that must be considered when migrating to a new release. Second, it discusses ways of improving performance in SunOS release 4.1. Third, this chapter describes the international standards the SunOS operating system is now compatible with.

5.2. Compatibility Issues

Binary Compatibility

User executables (including shared libraries), sources, and object files from 4.0.*x* releases should move to SunOS release 4.1 with no change. This includes many Sun and 3rd party unbundled products. There are exceptions to this that should, in practice, be encountered rarely. These are programs (in any of source, object, or executable forms) that:

a) depend upon the internal implementation of the kernel and its data structures (and their size and shape);

b) depend upon the object file format in a way that can be confused by extra information now appearing at the end of some object files in support of `#ident` directives n C source programs;

c) rely upon implementation attributes, such as the location, existence, or format of certain files not documented as part of the system's programming interface (such as temporary files created by library functions or utilities).

Programs that use the `kvm` library are likely to be in class `a` as described above, although the programs may still `function`, the data they access through the library may have changed in ways that will affect the function of the program. Examples of such data include the `proc` and `user` structures of the system.

Further, programming that modifies the implementation of the system, in particular kernel extensions or modifications, may require some conversion. Most drivers should work simply through recompilation, although other kinds of changes should be inspected to verify that whatever aspect of the system implementation they rely upon for their operation has not been altered.

It is important to know that, as always, while it is possible to move programs (their sources, objects, and executables) forward to release 4.1, it is *not* possible to move 4.1 programs back to prior releases. In 4.1, this extends to newly created filesystems, which incorporate updates from the Berkeley 4.3BSD "Tahoe" release that change the on-disk format of the filesystem. Such filesystems can not be directly connected to pre-4.1 systems, though network access to them is unaffected by this change, and tapes made via `dump` and `tar` and similar utilities are also unaffected. See the "Moving Disk Drives From SunOS 4.1 Systems to Systems Running Earlier Releases" section of chapter 2 for a description of this change

Client-Server Relationships

In SunOS release 4.1, services operate correctly among all combinations of architectures for client and server. In many cases, however, a high-powered client can *overwhelm* a low-powered server. It is therefore recommended that the server be more capable than any of its clients. This means that the server should have a CPU with higher MIPS rating. Some consideration should also be given to the amount of memory, speed of disks, number of disks, number of disk controllers, and layout of data among disks on the server. Review the following list for information on particular client-server relationships:

- Diskless and Dataless Clients

  The client and server should both run SunOS release 4.1. It is *not* advisable for a server running SunOS release 4.1 to support a client running an earlier release.

- General NFS™ Service

  There are no restrictions on general NFS access among machines and releases.

- RFS Service

  RFS should operate between all systems configured for RFS, including systems not provided by Sun.

- NIS (was YP) Clients and Servers

  There are some minor differences in the administration of the `/etc/hosts` database. This will not be visible to applications, but could change the results seen by users unless the system has been administered to be compatible.

  See *System and Network Administration* manual (Part Number 800-3805-10) for details.

5.3. Performance Tuning in SunOS release 4.1

The interactive performance of this release is 17% *better* than in SunOS release 4.0, *and* there is functionality inherent in SunOS release 4.1 which allows you to improve your performance even more. Performance tuning efforts in SunOS release 4.1 have two foci:

- Improving interactive performance on 4MB configurations. Standard SunView applications such as `mailtool` and `textedit` now run faster than they did in SunOS release 4.0.
- Maintaining non-interactive performance. Even with its many functional enhancements, SunOS release 4.1 performs at least as well as 4.0.

A large performance tuning effort was undertaken to achieve these goals, encompassing most areas of our bundled software, including:

- Tuning algorithms and data structures in SunView applications like `mailtool`.
- Tuning algorithms and data structures in the SunView libraries, and ordering the text of these libraries to reduce the working set sizes.
- Tuning the UNIX file system buffer cache management code.
- Tuning the kernel directory name lookup cache management algorithm.
- Converting many kernel data structures from static to dynamic allocation to provide a net kernel size decrease.
- Downsizing kernel data structures to reduce kernel size.
- Improving the kernel swap algorithms.
- Tuning the pixrect library.
- Eliminating code that provided compatibility with obsolete hardware and software options.
- Tuning Ethernet drivers.
- Revising the u-area management code to improve performance on machines with virtual address caches.

Improving Performance with `tmpfs`

A `tmpfs` filesystem allows a system's virtual memory resources of to be used as a filesystem. Files and directories can be created and deleted with normal UNIX semantics. These `tmpfs` filesystems do not require any additional disk space and the data can be accessed quickly. `tmpfs`-mounted directories appear identical to standard UNIX filesystems to users and most UNIX utilities.

For details and limitations, see the *System and Network Administration* manual (Part Number 800-3805-10) and the `tmpfs`(4S) manual page in the *SunOS Reference Manual* (Part Number 800-3827-10).

Performance Tips

The recommendations in this chapter will not solve every performance problem, but many can be eliminated by following these suggestions.

Kernel Configuration

Configuring the kernel, or using one of the preconfigured kernels or small kernels supplied with SunOS release 4.1 is an easy method of significantly improving performance. See chapter 6 of this manual for more information, and the *System and Network Administration* (Part Number 800-3805-10) and *Installing SunOS 4.1* (Part Number 800-3803-10) manuals for details.

Routing

Workstations that have only one Ethernet interface and do not act as NFS servers do not need to do dynamic routing with `in.routed`. Instead, you can make them route statically by commenting out (add a number sign # in column 1) the following lines in `/etc/rc.local`:

```
#if [ -f /usr/etc/in.routed ]; then
#   in.routed; echo -n ' routed'
#fi
```

Routing table entries will still be added or modified by the kernel as a result of ICMP redirect messages.

Diskless clients have a route provided automatically by the server. On other workstations, a command of this form can be added to `rc.local` just after the `in.routed` information. (*router* is the hostname of an IP router—also called a gateway—on the local network):

```
/usr/etc/route add default router 1
```

This action frees up both the pages used by `in.routed` and most of the memory allocated for routing table entries.

Accounting

Do not enable process accounting. If accounting is not configured into the kernel (`options SYSACCT`), or the file `/var/adm/acct` does not exist at boot time, accounting is not enabled. See `rc`(8) in the *SunOS Reference Manual* (Part Number 800-3827-10).

Eliminating Unnecessary Server Processes

Workstations typically only require these server processes: `portmap`, `ypbind`, `biod` (there are four of them), `syslogd`, `update`, `inetd`, and `lpd` (also `keyserv` if you use secure NFS and `sendmail`). Eliminating additional server processes will increase performance.

Miscellaneous

Do not enable file-system quotas. (By default, quotas are not enabled.) Replacing `/usr/ucb/quota` with `/usr/bin/true` will prevent possible delays at login time due to calls to `rpc.rquotad` on each NFS server from which you have a file system mounted.

Use the default SunView background, and do not use retained windows.

Do not enable `in.rwhod`.

5.4. Standards Compatibility

A summary of the standards with which SunOS release 4.1 is now compatible is below; see the *System Services Overview* (Part Number 800-3846-10) for a detailed discussion of the subject.

System V Compatibility

System V Accounting

System V changes in SunOS release 4.1 include modifications to the kernel and programs/commands to provide the System V accounting record format. A compatibility issue that arises from these changes is that old files of accounting records cannot be preserved and used across an installation.

Compatibility Changes

The effort to make the SunOS system SVID compatible is not exclusive to SunOS release 4.1. Modifications and additions have occurred to provide SVID compatibility in every release since SunOS release 3.2, and have continued in SunOS release 4.1. The following list outlines the changes that have been made to achieve SVID 2 compatibility:

- System V style accounting has been implemented in SunOS release 4.1. This can cause possible client/server and system administration confusion, since old accounting files cannot be processed with the new accounting routines.
- Mandatory file and record locking has been added to SunOS release 4.1; it will have no impact on application programs. See *Mandatory File and Record Locking* in chapter 7 of this manual.
- Commands are following the same path; new flags have been added wherever possible to comply with SVID. If the syntax and semantics required by the SVID 2 were incompatible with those of the SunOS release 4.0, a new command was added in `/usr/5bin`.
- RFS is bundled. Therefore, there are more kernel configuration options.
- Library routines have been modified to offer SVID 2-compatible interface. `libsvidm.a` provides a SVID 2-compatible math library; `libm.a` supports IEEE standard floating point behavior; and `libmalloc.a` provides SVID 2-compatible versions of `malloc()`, `calloc()`, and `realloc()`.
- The "malloc" library provides SVID 2-compatible versions of `malloc()`, `calloc()`, and `realloc()`. The default versions of these routines in `libc.a` behave as in SunOS release 4.0. The `libmalloc.a` versions of these routines indicate an error when space of size zero is requested; the

`libc.a` version of these routines returns a pointer to an area that should not be used for storage of data.

- The system now supports equal access to NFS and RFS. This change results in differences in installation and administration for NFS. See the *System Services Overview* manual (Part Number 800-3846-10) for descriptions of the SunOS utilities that have changed, and the SVID 3 utilities that are available.
- STREAMS capability was added in SunOS release 4.0.
- `terminfo`(5V) capability was added in SunOS release 3.2.

NOTE: Some object files and links may change location causing a few binary applications, shell scripts, and makefiles with hardcoded path names to break. Where possible, symbolic links are provided for compatibility.

POSIX Conformance

The System V libraries have been updated to meet the requirements of IEEE Std. 1003.1-1988 (commonly known as POSIX.1) with Common Usage C Language-Dependent System Support.

Some minor interfaces have changed to conform to POSIX.1 requirements, but existing binaries should continue to work correctly.

- The types of several function return values and arguments are more specific, and `lint`(1V) will complain about the old usage. However, the underlying types are still compatible.
- Null pathnames are now always treated as an error when using the System V compatibility package C library.

X/Open Compatibility

The X/Open compatibility package allows programmers to write software that conforms to the base level of the X/Open 1987 standard. The System V versions of most required commands, system calls, library routines, and headers conform to the *X/Open Programmer's Guide* (1987) definition (XPG-2), For routines and headers that do not, Release 4.1 provides X/Open conforming versions in `/usr/xpg2lib`, and `/usr/xpg2include`.

To compile C programs that conform to the X/Open standard, you can use the `cc` executive script in `/usr/xpg2bin`. To use this as the preferred compiler, place `/usr/xpg2bin` ahead of `/usr/5bin` and `/usr/bin` in the shell's execution path.

See also *System V Compatibility Features*, in the *System Services Overview* (Part Number 800-3846-10) for more information about System V.

`pathconf`: New Interface

SunOS release 4.1 has a new interface, `pathconf()`, that answers questions about the file system. It is limited to SunOS release 4.1 versions of UFS and NFS file systems.

UFS file systems enable this interface by default. NFS filesystems must be mounted with the `-o posix` option in order to support this interface. The remote system must understand mount protocol version 2, which is new to SunOS release 4.1.

Mounts that specify POSIX fail if the server does not support the version 2 mount protocol.

If the root partition is an NFS partition (diskless client) the root partition is mounted without the POSIX option during the single user portion of the boot sequence. This insures that root can always be mounted. If a client changes from a server that understands version 2 mount protocol to one that does not, the multi user remount of / will fail. The machine may be booted to single user, `/etc/fstab` edited to remove `posix` from the mount options, and then the machine will come up multi user.

`lint`(1) Library Support for Some Environments

SunOS release 4.1 supports `lint` libraries for the following environments:

- ANSI C
- POSIX.1
- XPG2
- SVID 2
- SVID 3
- BSD 4.3

These libraries may be used with lint to check portability to the desired environment. For example, the following will check for portability to a POSIX.1 conforming system:

```
coffee% lint -n -lposix posix_src.c
```

6

Installation Changes and Additions to SunOS Release 4.1

6.1. Purpose of this Chapter

This chapter provides information about installation features and tools that are new in SunOS release 4.1. For a complete discussion of the subject and "how to" instructions on installing the SunOS release 4.1, see the *Quick Install Guide* (Part Number 800-3824-10), and *Installing SunOS 4.1* (Part No. 800-3803-10) manuals for details.

6.2. SunInstall™ Changes and Additions

The following changes and features have been added to SunInstall; see the *Installing SunOS 4.1* manual (Part Number 800-3803-10) and the manual pages referenced for complete descriptions and details:

- SunOS release 4.1 software is available for installation from CD (compact disc). Installation from a single CD is much more convenient than from multiple tapes or diskettes. All of the software is contained on the one disk and the files needed can be copied from it without the additional overhead of making sure the correct version of the correct tape is installed. Installation from CD is currently supported on the following Sun Workstations:
  - SPARCstation 1
  - SPARCsystem 330
- *Quick Install* eases simple standalone installations. It allows the user to install the MINIROOT in one simple step. *Quick Install* should be used by any user booting as standalone (independent of a network). It gives the standalone user four installation configuration levels of software support: Minimum Necessary to Function, General User, Programmer, and Support for Everything.
- `add_client`(8) and `rm_client`(8) allow system administrators to easily add and remove clients from a network.
- `add_services`(8) can be used to set up any system to act as a fileserver for another architecture release and/or add additional software from a release tape or diskette *after* installation. It can also be used to add services for diskless clients.
- Many bug fixes that improve overall performance.

6.3. Improving Performance with Kernel Configuration

Kernel configuration options are briefly covered below, and are discussed in detail in the *System and Network Administration* manual (Part Number 800-3805-10).

The Importance of Configuring the GENERIC Kernel

Installing SunOS release 4.1 will give you the large GENERIC kernel by default; it is highly recommended that you reconfigure this kernel as soon as possible after installation. The GENERIC kernel contains code instructions needed by all hardware and software applications that Sun supports, including many you will not need. Tailoring this kernel to individual system needs significantly reduces memory requirements, which improves performance.

NOTE: You must have the `Sys` software category installed on your system before attempting to configure or replace a kernel. See the *System and Network Administration* manual for details.

Alternatives

To streamline the GENERIC kernel and improve performance, you have the following alternatives:

- Install one of the five Sun-supplied preconfigured small kernels (`GENERIC_SMALL`), that you will not need to build. This is the easiest way to configure the kernel, but saves the least amount of memory.
- Build a custom kernel using one of the Sun-supplied kernel configuration files (which simplify the process). This will save more memory than using one of the preconfigured kernels.
- Build a completely customized kernel by editing the GENERIC configuration file yourself. This is the most complex of the three alternatives outlined here, but will increase performance the most.

Using `install_small_kernel(8)` to Install Pre-configured Kernels

The `install_small_kernel` script can be used to easily install one of the four Sun-supplied pre-configured kernels below. The script can be run from the miniroot. Standalone systems can run the script themselves, and servers can use it to install small kernels on their clients.

NOTE: In all cases the script will ask for user confirmation before actually doing the install.

Pre-configured kernels for the four architectures are listed below.

`sun3` A Sun-3/50 or 3/60 (68020-based systems) diskless, or with up to two SCSI disks and one SCSI tape,

`sun3x` A Sun-3/80 (68030-based system) diskless, or with up to four SCSI disks, and one SCSI tape.

`sun4` A Sun-4/110, Sun-4/150, and SPARCsystem 330 with up to two SCSI disks and one SCSI tape,

`sun4c` A SPARCstation 1 with up to four SCSI disks, one CD-ROM, one floppy disk, and two SCSI tapes.

Sun-Supplied Kernel Configuration Files

These files are for common systems, and allow you to configure (build) a custom kernel more easily. These files found in `/usr/kvm/sys/`*ARCH*`/conf`, where *ARCH* is *sun2*, *sun3*, *sun3x*, *sun4* or *sun4c*, for, Sun-3 (68020-based), Sun-3x (68030-based), Sun-4 (SPARC-based) and Sun-4c (SPARC-based) systems, respectively. See `arch`(1): *Sub-Architecture Concept* in chapter 7 of this manual for details. See the *Installing SunOS 4.1* manual (Part Number 800-3803-10) for instructions.

In `/usr/kvm/sys/`*ARCH*`/conf`, file names beginning with `DL` are for diskless systems. Names beginning with `SDST` are for SCSI disk and tape, and `XD` and `XY` files support Xylogics disk controllers.

NOTE: The "`DL`" configuration files (with no numeric suffix) in these tables will result in the largest (slowest) kernels. They contain support for all of the controllers used by all of the platforms listed.

Table 6-1 *Sun-Supplied Kernel Configuration Files for* `sun3` *Architectures*

Configuration File Name	Supported Architecture
GENERIC_SMALL	3/50 or 3/60 With Up to 2 SCSI Disks and 1 SCSI Tape Drive
DL	Diskless Sun-3/160, 3/50, 3/260, or 3/110
DL50	Diskless 3/50
DL60	Diskless Sun-3/60
DL75	Diskless Sun-3/75
DL110	Diskless Sun-3/110
SDST50	Sun-3/50 With 1 SCSI Disks and 1 SCSI Tape
SDST60	Sun-3/60 With 1 SCSI Disks and 1 SCSI Tape
SDST110	Sun-3/110 With 1 SCSI Disks and 1 SCSI Tape
SDST160	Sun-3/160 With 1 SCSI Disks and 1 SCSI Tape
SDST260	Sun-3/260 With Up to 2 SCSI Disks and 1 SCSI Tape
XDMT160	Sun-3/160 With Up to 4 SMD-4 Disk Controllers, 2 Xylogics 1/2" Tape Drives, 2 SCSI Disks, and 1 SCSI Tape
XDMT260	Sun-3/260 With Up to 4 SMD-4 Disk Controllers, 2 Xylogics 1/2" Tape Drives, 2 SCSI Disks, and 1 SCSI Tape
XYMT160	Sun-3/160 With Up to 2 451 Disk Controllers, 2 Xylogics or Tapemaster Tape Drives, 2 SCSI Disks, and 1 SCSI Tape
XYMT260	Sun-3/260 With Up to 2 451 Disk Controllers, 2 Xylogics or Tapemaster Tape Drives, 2 SCSI Disks, and 1 SCSI Tape

Table 6-2 *Sun-Supplied Kernel Configuration Files for* `sun3x` *Architectures*

Configuration File Name	Supported Architecture
GENERIC_SMALL	Sun-3/80 With Up to 2 SCSI Tapes and 4 SCSI Disks
DL80	Diskless 3/80
DL470	Diskless Sun-3/470
SDST80	Sun-3/80 With Support for All Possible Standard Devices and Software Options
XDXT470	Sun-3/470 With Up to 4 SMD-4 Disk Controllers, 2 Xylogics 1/2" Tape Drives, 4 SCSI Disks, and 4 SCSI Tape
XYXT470	Sun-3/470 With Up to 2 Xylogics 450/451 Controllers, 2 Tapemaster or Xylogics 1/2" Tape Drives, 4 SCSI Disks, and 1 SCSI Tape

Table 6-3 *Sun-Supplied Kernel Configuration Files for* `sun4c` *Architectures*

Configuration File Name	Supported Architecture
GENERIC_SMALL	SPARCstation 1 With Up to 4 SCSI Disks and 2 SCSI Tapes
DL60	Diskless SPARCstation 1 (Does not support SCSI Devices)
NFS60	SPARCstation 1 With Up to 4 SCSI Disks and 2 SCSI Tapes, Configured to Boot From NFS Filesystems
SDST60	SPARCstation 1 With Up to 4 SCSI Disks and 2 SCSI Tapes

Table 6-4 *Sun-Supplied Kernel Configuration Files for* `sun4` *Architectures*

Configuration File Name	Supported Architecture
GENERIC_SMALL	Sun-4/110 or 4/330 With Up to 4 SCSI Disks and 2 SCSI Tapes
DL	Diskless Sun-4/260, 4/280, 4/110, or 4/330
DL110	Diskless Sun-4/110
DL330	Diskless Sun-4/330
SDST110	Sun-4/110 With Up to 4 SCSI Disks and 2 SCSI Tapes
SDST330	SPARCstation 330 With Up to 4 SCSI Disks and 2 SCSI Tapes
XDXT260	Sun-4/260 With Up to 2 SMD-4 Controllers, 2 Xylogics 1/2" Tape Drives, 2 SCSI Disks, and 1 SCSI Tape
XYXT260	Sun-4/260 With Up to 2 Xylogics 540/451 Controllers, 2 Xylogics 1/2" Tape Drives, 2 SCSI Disks, and 1 SCSI tape

6.4. Pre-Loaded Disk (SPARCstation 1 and Sun-3/80 only)

Pre-Loaded Software

In order to simplify getting Sun-3/80 and SPARCstation 1 systems up and running, systems delivered with at least one hard disk will have a useful subset of the SunOS release 4.1 operating system loaded at the factory. This subset will enable program and demo execution, SunView use, and System V functionality. Only one disk, (`/dev/sd0` for a SPARCstation 1; `/dev/sd6` for a Sun-3/80), in each system will have the software pre-loaded.

The following packages are installed on a pre-loaded disk:

```
root
usr
Kvm
Sys
Networking
SunView_Users
SunView_Demo
Text
Install
System_V
Demo
```

These packages use approximately 44 MB of disk space in the `/usr` partition for a SPARCstation 1 with `sd0g` or `sd6g` mounted on `/usr`.

A SPARCstation 1 with the software pre-loaded has approximately 4.0 MB of free space available in the root partition with `sd0a` mounted on `/`, and approximately 23 MB of free space available in the `/usr` partition with `sd0g` mounted

on `/usr`. A Sun-3/80 will have approximately 4.8 MB of free space available with `sd6a` mounted on `/`, and approximately 24.5 MB of free space available in the `/usr` partition `sd6g` mounted on `/usr`. It will also have 14 MB of swap space. See the *Installing SunOS 4.1* (Part Number 80-3803-10) and *System and Network Administration* (Part Number 800-3805-10) manuals for complete information on default disk partitioning.

A system delivered with pre-loaded software does not need to have `suninstall` run on it to make it usable. It just needs to be configured with a name and certain networking and NIS name service information. The first time the system boots, a short series of questions will appear on the screen. Answering these questions is all that is necessary to configure the system for use. Once these questions are answered, the boot procedure will continue and the system will be usable as soon as the boot procedure completes. (See the *Sun System and Network Manager's Guide* in the SPARCstation 1 Owner's Set for a complete description of the configuration procedure.)

While many needs can be satisfied with the pre-loaded software setup, some can not. If you need more free space than is available, need additional packages installed, or need different disk partitioning, you will have to do more than simply configure your system.

For information on installing additional software packages after your system is up and running, see the *System and Network Administration* (Part Number 800-3805-10)

If you need more free space after your system is configured and up and running, you may be able to remove unneeded files. If this is not possible, you may customize your installation using `suninstall`.

If you need to alter the default disk partitioning, you must run `suninstall`.

The `sys-config`(8) Utility for Pre-Loaded Disks

The `sys-config` utility configures systems with pre-loaded software. It runs automatically on systems which do not need software installation in order to boot. `sys-config` will configure a standalone system, a NIS client or a non-networked system, but will not completely configure a server. You must be superuser to run `sys-config`; it can be found in `/usr/etc/install`.

`sys-config` runs automatically when a system with pre-loaded software is booted for the first time. It also runs during the first boot after `sys-unconfig` has been run. If the file `/etc/.UNCONFIGURED` exists, `sys-config` will run when the system is booted.

Since configuring a system consists of assigning a hostname, selecting a time zone, specifying an IP address (if appropriate), and specifying an NIS domain (if appropriate), `sys-config` can also be run *after* installation to change the hostname, time zone, NIS domain information or network information, instead of manually editing those files.

Files Affected by `sys-config(8)`

The following is a list of files and descriptions of how they are affected by `sys-config`:

`/etc/hosts`
Comments out any entry in `/etc/hosts` with the same hostname or IP address as that being assigned.

Adds an entry with the hostname and IP address being assigned. The default IP address of 192.9.200.1 is for illustrative purposes *only*. It must be changed to the user's actual IP address. This does not affect non-networked Standalone systems, but all others must get an actual IP address from their network administrator.

Makes `loghost` an alias of the `hostname` entry.

`/usr/lib/zoneinfo/localtime`
Moves the current `/usr/lib/zoneinfo/localtime` file to `/usr/lib/zoneinfo/localtime-`.

Creates a hard link from the file corresponding to the time zone specified by the user to `/usr/lib/zoneinfo/localtime`.

`/etc/.UNCONFIGURED`
Removes this file.

`/etc/install/sys_info`
Updates this file with the information collected from the user.

The `sys-unconfig` Utility

The `sys-unconfig` utility undoes what `sys-config` does. You must also be superuser to run `sys-unconfig`.

The following files are affected by `sys-unconfig`:

`/etc/hosts`†
The current `/etc/hosts` file is concatenated to the end of `/etc/hosts.saved`, and `/etc/hosts` is created and becomes the default.

`/usr/lib/zoneinfo/localtime`

Removes `/usr/lib/zoneinfo/localtime`.

Copies the default time zone file, `/usr/lib/zoneinfo/PST8PDT`, to `/usr/lib/zoneinfo/localtime`.

`/usr/lib/zoneinfo/localtime-`
Removes this file.

†The `/etc/hosts` file is a list of workstations and their IP addresses. If your system is using NIS (was YP) then this file is only consulted when booting; otherwise the NIS maps are consulted.

`/usr/etc/install/files/sys_info`
Restores the values that it had on a pre-loaded disk.

`/etc/.UNCONFIGURED`
Creates this file.

6.5. Installation Changes since SunOS 4.0.3 (in 4.0.3 - 4c)

- Setting the time zone in `suninstall`(8) is done by walking through a series of menu selections. It is no longer necessary to remember and type in the name of the time zone file. If your time zone is not listed, go to the main time zone screen and select `Greenwich Mean Time`. You will then be able to specify the number of hours east or west of GMT.
- `SunInstall` supports 3-1/2" diskettes as a release media. The support for diskettes is fully integrated throughout `suninstall`.
- After `suninstall` has been run in the MINIROOT, the `/etc/install` directories are preserved in the permanent filesystem. When the system is booted after installation, the directory contents accurately reflect the installation.
- A new disk partitioning scheme uses available space more efficiently on 104 MB hard disks. All 104 MB hard disks shipped by Sun are formatted and labeled as follows:

```
partition a - starting cyl      0, # blocks    16170 (77/0/0)
partition b - starting cyl     77, # blocks    28140 (134/0/0)
partition c - starting cyl      0, # blocks   204540 (974/0/0)
partition d - starting cyl      0, # blocks        0 (0/0/0)
partition e - starting cyl      0, # blocks        0 (0/0/0)
partition f - starting cyl      0, # blocks        0 (0/0/0)
partition g - starting cyl    211, # blocks   160230 (763/0/0)
partition h - starting cyl      0, # blocks        0 (0/0/0)
```

This reflects the allocation of space as follows:

use	size(MB)	partition	mounted on
root	8	a	/
swap	14	b	N/A
usr	remaining space	g	/usr

or

use	size(MB)	partition	mounted on
user defined	104	c	N/A

This results in approximately 72 MB of usable space in `/usr`. In a system that is pre-loaded at the factory, there is approximately 35 MB of usable space in `/usr` after the software has been loaded.

There is approximately 4.5 MB of usable storage in the root / partition.

7

New Software: Changes and Additions to SunOS Release 4.1

7.1. General Software Additions

arch(1): Sub-Architecture Concept

The arch(1) command was modified in SunOS 4.0.3 - 4c to include a –k option. Executing `arch –k` prints the kernel architecture type, such as `sun4` or `sun4c`. This indicates the architecture of the machine, and is important only for programs explicitly depending on the hardware architecture.

The *archname* option is also new to the arch(1) command. The command `arch` *archname* returns `0` (true) when the machine can run the application binaries of the *archname* machine architecture, and returns `exit status 1` (false) when it cannot run those binaries.

For example, `arch sun4` returns `exit status 0` (true) on both `sun4` and `sun4c` kernel architectures because these machines are binary compatible for application software.

The path for application architecture-dependent executables is called the *execpath* and is unchanged from SunOS 4.0. The default *execpath* for a standalone or server machine is `/usr`, and for a diskless client is

`/export/exec/`*ARCH*

The path for the kernel architecture dependent executables is called the *kvmpath* was introduced in SunOS release 4.0.3 - 4c. The default kvmpath for a standalone or server machine is `/usr/kvm`, and for a diskless client is

`/export/exec/kvm/`*kernel_ARCH*

Every diskless client must mount the execpath on `/usr`, and the kvmpath on `/usr/kvm`. You define both the execpath and the kvmpath on the `HOST FORM`, `SOFTWARE FORM`, and `CLIENT FORM` menus in *suninstall*.

Executing arch(1) will display the application architecture of a machine. In the case of a SPARCstation 1, `arch` would return `sun4`. Adding the –k argument to `arch` will show the kernel architecture of the machine. In the case of a SPARCstation 1, `arch –k` would return `sun4c`.

Application architecture	Kernel architecture	Current Sun System Models
sun2*	sun2	2/50, 2/120
sun3	sun3	3/50, 3/60, 3/75, 3/110, 3/140, 3/150, 3/160, 3/180, 3/260, 3/280
sun3	sun3x	3/80, 3/460, 3/470, 3/480
sun4	sun4	4/110, 4/150, 4/260, 4/280, 4/390, SPARCsystem 330/370 SPARCsystem 430/470
sun4	sun4c	SPARCstation 1
sun386*	sun386	386*i*/150, 386*i*/250
***Not Supported in Release 4.1**		

See `arch`(1) in the *SunOS Reference Manual* (Part Number 800-3827-10) for details.

Requesting Hardware Flow Control Capability

In SunOS release 4.1, hardware flow control capability can now be requested for use with `tip`, `uucp`, and `getty`. It is documented in `gettytab`(5), `remote`(5) and some `uucp` documentation. The corresponding tip variable `hardwareflow` has been added to `tip`(1C). See those manual pages in the *SunOS Reference Manual* (Part Number 800-3827-10) for details.

Enhanced `dump`(8) and `restore`(8)

The `dump` and `restore` commands are enhanced with a new option that allows keeping an on-line index of dumps. This index allows you to more easily locate the tape or diskette on which a file has been dumped, without the need to mount many dump tapes.

For more details, see the `-a` option in the `dump`(8) and `restore`(8) manual pages, and the *System and Network Administration* manual (Part Number 800-3805-10).

The `dump` command has been enhanced with two additional features. The new `-v` (verify) option instructs `dump` to reread a dump tape and verify that it matches the contents of the disk being dumped. `dump` also contains a new feature which allows a specified set of files and directories on a disk partition to be dumped. Previous versions only allowed dumps of a complete disk partition.

`setrlimit`(2): Sets Process Resource Limit

SunOS release 4.1 includes a new resource limit, `RLIMIT_NOFILE`, which affects file descriptor creation.

Processes set resource limits with the `setrlimit(2)` system call. For each process, resource limits are set for file size, data size, core file size, CPU time, and other parameters. The new `RLIMIT_NOFILE` resource limit constrains the maximum value allowed for newly created file descriptors in a process. For example, when `RLIMIT_NOFILE` equals 64, a process can only create file descriptors 0 through 63.

See the `setrlimit`(2) manual page for a full explanation of setting resource limits.

Hard and Soft Limits

As with all resource limits, the `RLIMIT_NOFILE` limit has two components: a hard limit and a soft limit.

The soft limit determines whether a system call that would create a file descriptor of a particular value succeeds or fails: if the value is less than the soft limit, the operation proceeds; if not, it fails.

The hard limit places a boundary on the values allowed for the soft limit. For example, if the hard limit is 32, the soft limit is restricted to the range 0 through 32.

As with all resource limits, only processes with superuser powers can increase the hard `RLIMIT_NOFILE` limit. Any process can change the soft limit, up to the maximum allowed by the hard limit. Changes affect only the process and any child processes it subsequently creates.

Default `RLIMIT_NOFILE` Values

When the system boots (and default values have not been changed by the system administrator), the hard limit is set to 256 and the soft limit is set to 64.

Note that the default soft limit value is compatible with SunOS releases previous to 4.1. If the default value is not changed, file descriptor creation succeeds and fails just as in SunOS release 4.0.

Using `getdtablesize`(2) to Set Values

Before release 4.1, the `getdtablesize`(2) system call returned a constant value, and the size of the table was equal to the number of file descriptors that a process could open. In SunOS release 4.0, this value was 64.

In SunOS release 4.1, the `getdtablesize` call returns the soft limit value of `RLIMIT_NOFILE` and so is no longer a constant value. Programs that assume a constant value for `getdtablesize` will break. However, note that the `getdtablesize` value that a process sees changes only when the process itself alters the `RLIMIT_NOFILE` value.

Caution

It might seem that the soft limit is equal to the number of open file descriptors a process can have. But, since file descriptors open when the `RLIMIT_NOFILE` soft limit changes are unaffected by the change, these values are not necessarily the same. For example, if the soft limit is 64, a process can open file descriptors 53 through 63, then change the soft limit to 32 and open file descriptors 0 through 32. Although the soft limit is now set to 32, the process can have 44 open file descriptors.

For most situations, no file descriptors with values over the new soft limit are likely to be open when the soft limit is lowered; in this case, the maximum number of file descriptors equals the soft limit value.

A common programming technique is to use `getdtablesize()` as the upper bound of a loop intended to iterate over all file descriptors. This technique is sometimes used to close all open file descriptors.

In SunOS release 4.1, this technique is not guaranteed to find all open file descriptors.

To examine all possible open file descriptors, the program should loop from 0 to the configuration constant `NOFILE`, defined in `param.h`. `NOFILE` contains the maximum hard limit value ever possible on the system, and is set to 256 in SunOS release 4.1.

`poll`(2) System Call Extension

The `poll`(2) system call has been extended to work on all types of file descriptors, not just `STREAMS`. This can help when converting code from SunOS systems to an AT&T UNIX System V Release 4 environment; `poll` is native to SVR4, whereas `select`(2) is emulated in SVR4. This functionality is not compatible with pre-System V Release 4 AT&T environments. See the `poll`(2) manual page in the *SunOS Reference Manual* (Part Number 800-3827-10) manual for details.

`rpcgen`(1) Improvements

`-I` Option Added for Use With `inetd`

The `-I` option has been added to `rpcgen` for use with `inetd`. Now the server programs can be started by `inetd` also.

`-L` Option Added for Use With `syslog()`

`rpcgen -L` generates errors code to `syslog()`, instead of using `stderr` to report them. This is useful for the server daemons.

`-T` Option Added to Generate Indexed-by-Procedure Table

With `-T` option, `rpcgen` generates an indexed-by-procedure table. There are applications in which a table (with each entry containing the handler or zero, and the `size` and `xdr` routines for the argument and result) indexed by procedure number are very useful for both client and server programs. For example, the client side use is that of determining the `xdr` routines to use, while the server side use is for dispatching as well as `xdr()`-ing.

`rpcgen` Now Accepts `-Ddefines`

`rpcgen` accepts `-Ddefines`. This enables users to specify their own application specific options via `#defines`.

RPC Library Improvements

Two options have been added to `clnt_control()`:

`CLGET_FD` allows a user to get the associated file descriptor value of a `CLIENT` handle. This is useful when a user wants to optimize the usage of file-descriptors and use the same one.

`CLSET_FD_CLOSE` and `CLSET_FD_NCLOSE` allow users to specify whether to close or not close the file descriptor while destroying the client handle.

`portmap`(8C) Improvements

`portmap`(8C) now handles `PMAPPROC_CALLIT` requests (forwarding requests) with sockets instead of forking off the process. This will significantly increase performance, and solve the problem of `portmap` failing and exiting without explanation. See the `socket`(2) manual page in the *SunOS Reference Manual* (Part Number 800-3827-10) manual for details on sockets.

`portmap` now *only* accepts `PMAP_SET` and `PMAP_UNSET` calls from the clients of its own machines. It also solves the problem of non-root users accidentally unmapping ports.

Asynchronous I/O

Asynchronous I/O allows the user to initiate several I/O requests before inquiring about their completion. From the user's point of view, the advantage is that CPU activity continues concurrently with I/O operations. In the synchronous case, the user must wait for the I/O to complete before issuing the next request.

Four new library functions have been created to support this utility:

```
aioread(3)
aiowrite(3)
aiowait(3)
aiocancel(3)
```

Mandatory File and Record Locking (MFRL)

Mandatory File and Record Locking is a synchronization mechanism that restricts access by programs accessing the same files simultaneously. When a file is mandatorily locked, access to that data by any other process is restricted according to the type of lock on the file. The standard I/O subroutines and I/O system calls enforce the locking protocol. Control over records should still be performed explicitly by requesting an appropriate record lock before I/O operations, but an additional check is made by the system before each I/O operation to ensure the record locking protocol is being honored.

NOTE: Use of MFRL is not recommended. When used, mandatory locks should be implemented with extreme care. If a runaway or otherwise out-of-control process should hold a mandatory lock on a file critical to the system and fail to release that lock, the entire system could hang or crash. In addition, Mandatory locks will not work on NFS files except in restricted cases, and will not work for mapped files.

See the File and Record Locking Chapter in the System Services Overview (Part Number 800-3846-10), Chapter 14 of the *System and Network Administration* manual (Part Number 800-3805-10), and the `fcntl`(2V), `lockf`(3), and `fcntl(5)` manual pages in the SunOS Reference Manual (Part Number 800-3827-10) for details.

`libkvm` Changes

`libkvm` now recognizes user addresses and more types of kernel addresses; this makes debugging kernel crash dumps easier. In particular, use of the `$<calltrace` macro to produce a trace-back will, if possible, continue the traceback from kernel space into user space.

`mlock`(3): Lock Down Memory in a Process

SunOS release 4.1 supports the page and process memory locking operations `mlock`(3), `mlockall`(3), and `plock`(3). These operations can be used to force portions of files and processes into physical memory and hold them there. These functions can improve the response time of processes that use the locked memory, since delays for paging are eliminated.

The `mlock`(3) function (and its converse, `munlock`(3)) locks (unlocks) a range of process addresses. `mlockall`(3) (and its converse, `munlockall`(3)) locks (unlocks) an entire address space. `plock`(3) is a SVID-compatible interface for memory locking. See the appropriate manual pages in the *SunOS Reference Manual* (Part Number 800-3827-10) for detailed information on the use of these functions.

Memory locking can improve the performance of a given process, but overall system performance may suffer because the memory pool available to other processes is smaller. Because of this potential impact on system resources and performance, use of memory locking operations is restricted to the super-user.

Program Controlled Binding

The `dlopen`, `dlsym`, `dlerror`, and `dlclose` functions provide a simple programmatic interface to the services of the dynamic link-editor. The following operations are provided:

- Adding a new shared object to the program address space.
- Obtaining the address bindings of symbols defined by such objects.
- Removing such objects when they are no longer required.

See the `ldopen`(3) manual page in the *SunOS Reference Manual* (Part Number 800-3827-10) for more information about these functions.

`gettytab`(5): New Capabilities Added

Four options to set `tty` modes have been added to `gettytab`(5): `ms`, `m0`, `m1`, and `m2`. They are all string valued and can be used to specify any mode supported by `stty`(1v). This permits modes not supported by the older terminal interface described in `ttcompat`(4M) to be set or cleared.

For example, the following option can be used in `/etc/gettytab` to enable hardware (RTS/CTS) flow control on a particular line:

```
:ms=crtscts:
```

`ms` sets modes that continually apply, whereas `m0`, `m1`, and `m2` set modes which apply concurrently with those set by `f0`, `f1`, and `f2`. The modes specified by `ms`, `m0`, `m1`, and `m2` are applied *after* the modes specified by other existing capabilities. See `gettytab`(5) in the *SunOS Reference Manual* (Part Number 800-3827-10) for details.

`sundiag`(8): Enhanced Diagnostic Software

SunOS release 4.1 includes the new `sundiag` diagnostic system. This is a SunView-based user interface that tests system devices and peripherals. It replaces the `sysdiag` program, which is no longer supported. See the *SunDiag User's Guide* for details.

`make`(1): Enhancements

The `make`(1) command has been enhanced to allow variables to be set to the output of shell commands. This makes it easier to use the new `arch`(1) features in makefiles.

NOTE: Pre-4.1 versions of `make` cannot use this new feature. Trying to use a release 4.1 makefile with an old version of `make` will fail, returning a cryptic error message.

See `make`(1) in the *SunOS Reference Manual* (Part Number 8003827-10) for details.

`eject`(1): New Utility for Ejecting Diskettes

The `eject`(1) utility (introduced in SunOS release 4.0.3 - 4c) supports software ejects for floppy diskettes and CD-ROM discs on the Sun-3/80 and SPARCstation 1 workstations. See the `eject`(1) manual page for a full description of `eject`(1) features.

New Devices

`makedev`(8) now creates the following devices:

- `fd` is the floppy drive for the SPARCstation 1 and Sun-3/80.
- `pp` is the Sun-3/80 printer port.
- `ppdiag` is a diagnostics special device file.
- `audio`, the SPARCstation 1 audio device,
- `openprom`, the Openprom eeprom options device,
- `sbus0 - sbus3`, the SBus virtual address space special devices,
- `vd`, the loadable driver control device,
- `sr0`, the SunCD driver.

The GENERIC Configuration File

The standard GENERIC configuration file has been changed to support new devices. See the *System and Network Administration* manual (Part Number 800-3805-10) for details.

Write Check Functionality: New Ioctl, Manual Page (`dkctl`(8S))

This feature allows the verification of writes on the disks of Sun workstations (write check). When this function is enabled, and toggled "on", the device driver verifies each write to the disk partition for which the function is enabled. The verification is done by hardware if possible (for example the `scsi verify` command), or by a readback and compare of just-written data.

`dkctl`(8S) Manual Page Created

This feature is not enabled by default; it must be enabled by editing the `/etc/rc.local` file, then using the `wchk` command to `dkctl`(8) (control special disk operations). For example, adding the following line to the end of `/etc/rc.local`:

```
dkctl /dev/rXXNP wchk
```

will enable write checking for device *XX*, unit *N*, partition *P*.

DKIOCWCHK Ioctl Added to dkio(4S)

Once enabled, write check can be toggled "on" and "off" using the DKIOCWCHK ioctl to dkio(4S).

Performance is impacted by this feature. See the dkio(4S) and dkctl(8) manual pages for details.

intr(8): New Boot Sequence Interrupt Command

Interrupting commands during the boot sequence is done differently in 4.1. Subshells that do output redirection to /dev/console are no longer necessary. Old rc files work, but no commands in the old rc files are interruptible during boot sequence.

In earlier releases, the shell that ran the /etc/rc* shell scripts had no associated controlling terminal. Output was sent to /dev/console via redirection. This redirection had the side effect of making the command interruptible.

In SunOS release 4.1, the shell is given /dev/console as its controlling terminal. All output generated from the rc files (and the commands contained therein) will go to the console by default.

The shell is started in a state that ignores keyboard signals (the tty's process group is set to be different from that of the shell). In order for commands to be interruptible, they must be prefixed by the intr(8) wrapper:

```
coffee% intr fsck -p -w / /usr
```

intr changes the state of the console and the command such that the command and its children are interruptible, but previous and subsequent commands are still protected. See intr(8) in the *SunOS Reference Manual* (Part Number 800-3827-10) for details.

dbx(1): New modules Commands for Selective Debugging

The modules, modules select, and modules append commands have been added to dbx(1) in SunOS release 4.1 to facilitate debugging very large programs; they allows users to select and use just the specific parts of debugging information needed.

modules controls and displays the amount of source level debugging information available to dbx(1).

Usage:

```
modules
modules select [all|objname] [objname]...
modules append [objname]...
  ...
{debug file}
```

Displaying the Debugging Object Files

`modules` with no arguments displays the set of object files and associated source file pathnames for which `dbx`(1) currently has source level debugging information.

Setting the Module Selection List of Object Files

`modules select` controls or displays the modules selection list. When the module selection list is set, subsequent `debug` commands will restrict the collection of source level debugging information to the object files in the module selection list. Object files will be treated as if they were compiled without the `-g` switch if they are not in the module selection list. Setting the module selection list allows users to control the size of `dbx`'s internal symbol tables when debugging large programs.

The command `modules select` without any arguments displays the current module selection list.

The command `modules select` *objname* [*objname*]... sets the modules selection list to include *only* the named object files.

The command `modules append` *objname* [`objname`]... adds the named files to the modules selection list.

Once a modules selection list is set, all subsequent `debug` commands will interrogate it.

Disabling the Selection List

The command `modules select all` disables (`unsets`) the module selection list. Subsequent `debug` commands will process all files and symbols. This is the default state when `dbx` is started.

Example:

```
/usr/ucb/dbx
(dbx) debug a.out
Reading symbolic information...
Read 1600 symbols
(dbx) modules
    object file(a.out)      source files
     a.o                  ../src/a.c ../src/a.y
     b.o                  ../src/b.c
(dbx) modules select b.o
(dbx) debug a.out
Reading symbolic information...
Read 1600 symbols (1 of 2 files selected)
(dbx) modules
    object file(a.out)      source files
     b.o                  ../src/b.c
```

See the `modules`(1) man page and the *Debugging Tools* manual (part number 800-3849-10) for details.

`setsid`(2V/8): Controlling Terminal Assignment

Controlling Terminals

A controlling terminal is the mechanism by which keyboard signals are dispatched to user processes. If a process is running without a controlling terminal, interrupts (`^C`) and job control functions (`^Z`) will not work.

In SunOS release 4.1, a process must be a session leader in order to acquire a controlling terminal. A process becomes a session leader by calling `setsid`(2V).

`setsid()` did not exist in earlier releases; it was created to support POSIX job control. This section of the document outlines the steps that have been taken to ensure that a process is a session leader at the appropriate moments. We also discuss programs that are not helped by these steps.

New Requirements for Controlling Terminals

Controlling terminals are established as a side effect of `open`(2V). In SunOS release 4.0 you received a controlling terminal if your process group was `0`. Alternatively, you received a controlling terminal if the terminal was not in use as such, you did not have a controlling terminal, and you were a process group leader (session leader precursor).

In SunOS release 4.1, controlling terminals are still established as a side effect of `open`(2V). However, the requirements are that you are a session leader, you did not have a controlling terminal, and the `tty` is not in use as a controlling terminal.

Since `setsid`(2V) is new to SunOS release 4.1, the process environment has been modified so as to have a process be a session leader whenever that process wants a controlling terminal. Processes are given sessions when they:

- are spawned from `init`(8), `inetd`(8), or `rlogind`(8),
- disassociate themselves from their controlling terminal via an `ioctl(cttyfd, TIOCNOTTY, 0)`, or
- protect themselves from `tty` signals via a `setpgrp(mypid, 0)`.

This catches almost all of the existing programs. A small class of programs are created without a controlling terminal and assume their process group is `0` (as was the case under SunOS release 4.0 and BSD release 4.3). Under earlier releases, having a process group of `0` was sufficient to acquire a controlling terminal as part of `open`(2V). Since no process is ever spawned with a process group of `0` any more, programs that assume that state are in trouble. We have provided a wrapper for these programs that fixes their problem. See the `setsid`(8V) and `setsid`(2V) manual pages in the *SunOS Reference Manual* (Part Number 800-3827-10) for details.

The typical problem cycle is:

- A complaint comes in that someone is seeing `TIOCSPGRP-interrupted system call` messages on the console, and/or their shell is not responding to keyboard signals, and/or their csh/ksh/emacs is hung.
- The user does a `ps -jax` on the system and sees something like:

```
PPID   PID  PGID   SID TT TPGID  STAT   UID  TIME COMMAND
 ...
 394   399   399   399  ?    -1   SOE   812  0:00 emacs foo.c
 ...
```

The emacs does not have a controlling terminal.

- The user wraps `setsid`(8) around the offending program and all is well.

The following code shows canonical forms for getting a controlling terminal; these methods were and still are supported:

```
/*
 * System V (Release 3 and earlier)
 */
(void) setpgrp();
fd = open("/dev/ttya", O_RDWR);
```

```
/*
 * BSD 4.3 and earlier
 * Very careful programmer
 */
if ((fd = open("/dev/tty", O_WRONLY)) != -1) {
     (void) ioctl(fd, TIOCNOTTY, 0);
     (void) close(fd);
}
(void) setpgrp(0, 0);
fd = open("/dev/ttya", O_RDWR);
```

```
/*
 * BSD 4.3 and earlier
 * Lazy but correct programming
 */
(void) setpgrp(0, 0);
fd = open("/dev/ttya", O_RDWR);
```

The following code is the POSIX version of the same:

```
/*
 * POSIX.  The fork is necessary since a setsid(2)
 * may be successfully called only once per process.
 */
if (fork() != 0)
     exit(0);
(void) setsid();
fd = open("/dev/ttya", O_RDWR);
```

The following code is no longer supported:

```
/*
 * BSD 4.3 and earlier
 * assumes pgrp == 0 if there is no ctty,
 * bad assumption in SunOS 4.1.
 */
if ((fd = open("/dev/tty", O_WRONLY)) != -1) {
     (void) ioctl(fd, TIOCNOTTY, 0);
     (void) close(fd);
}
fd = open("/dev/ttya", O_RDWR);
```

Color Enhancements

Bit True Color

To support the **CG8** and **CG9** there is a new attribute:

```
CANVAS_COLOR24 , TRUE
```

The use of this attribute is summarized below.

Table 7-1 *Color Attribute Usage Summary*

Effect	SunView Attributes
mono	window_create()
8-bit indexed emulation	window_create() pw_putcolormap
8-bit indexed emulation	window_create(CANVAS_COLOR24, TRUE) pw_putcolormap
24-bit	window_create(CANVAS_COLOR24, TRUE)

Text subwindows in SunView tools such as `shelltool`, `cmdtool`, and `textedit` have command line arguments that allow you to specify a foreground and background color for a window. These command line options are as follows:

```
-Wf r g b -Wb r g b -Wg
```

See `sunview`(1) in the *SunOS Reference Manual* (Part Number 800-3827-10) for a definition of these options. The **CG9** supports all of these options, but Sun recommends only limited use of the `-Wg` option for interactive performance reasons. When `-Wg` is used, every pixel of every character in the window requires 32-bit operation instead of the 1-bit operation required if the window remained in the overlay.

Colored Panel Text Items

SunView 1.80 offers the PANEL_ITEM_COLOR attribute to support colored panel items. Its use is simple:

```
PANEL_ITEM_COLOR, color,
```

The *color* should be given as an index into a colormap, such as `sunwindow/cms_rainbow.h`.

Changes to Defaults Database

The following items have been removed from the defaults database maintained by the `Defaults Editor`.

- `Stack_Menus`
- `Menu_Prompt`

The following items have been added. The items are shown as `Category/Item`, where `Category` is selected from the Defaults Editor `Category` cycle menu at the top left of the control panel. `Item` is the menu item within `Category` and appears in the text window below the control panel.

- `SunView/Selection_Timeout`, allows the user to adjust the time limit for establishing selection client status. This is used by SunView tools when trying to become selection clients. If the selection service does not respond within the time specified (for example, because the system is heavily loaded), the tool will abort the request.
- `Help/Directory` is the directory containing the help text files used by Spot Help and More Help.
- `Help/Server` is the server program for More Help. By default the item in listed as `help_viewer` , but this program only exists on the Sun386i. For other architectures, users must write their own server and list it in the defaults database.

SunView User Features

Editable Panel Text Items

Panel text items, such as the name of a mail folder in `mailtool`, can now be edited directly. You can put the cursor in the middle of an item to insert or delete text, instead of having to delete from the end of the item. Note that since long panel text items will scroll horizontally, this editing capability is limited to the portion of the text item that is visible.

Locking Sliders

New in SunOS release 4.1, locked window sliders are the default. The difference is that locked sliders stay engaged as long as the mouse is moving inside the window or panel, even if the pointer slips off the slider itself. In previous releases, a slider was engaged only as long as the mouse pointer was on top of it.

7.2. New Software: Graphics

GPSI Enhancements

This section lists the Graphics software libraries that have been changed in this release. These changes include functionality enhancements or bug fixes. The functionality enhancements are described below. of this manual.

Pixrect Library

The Pixrect library has been upgraded; pixrect bugs have been fixed, and the library has been augmented to handle lookup tables.

Lookup Tables

The CG8 and CG9 frame buffers have introduced true color lookup table capability. The Pixrect function `pr_putcolormap` is currently used for 8-bit indexed colormaps. Since the lookup table differs from a colormap, the new commands `pr_putlut` and `pr_getlut` are added to Pixrects.

No error is generated when a `colormap` function is applied to a true color planegroup: the function is ignored by the hardware. You must use the `pr_*lut()` functions.

Because of the special encoding used by the CG9's overlay and enable planegroups, the `pr_*lut()` functions must be used to modify their pixel color values as well.

See the *4.0.3 - GFX Rev. 1 Pixrect Manual* for details.

GPSI

The GPSI microcode has undergone a number of bug fixes and functionality enhancements to support accelerated true color and improve interactive graphics applications. The enhanced microcode will run on the following configurations of graphics processor hardware:

- GP2 / CG5
- GP2 / CG9

The new features added are listed below:

1. Additional Markers — 8 additional predefined markers, making twelve total, have been taken from the set of defined SunPHIGS markers. They have been included to accelerate SunPHIGS applications.
2. Spotlights — Spotlights are similar to positional lights. Positional lights are a directional light source, whose intensity diminishes as function of the lights angle from the light's direction. Spotlights are positional lights whose output intensity is clamped to zero beyond a defined spread angle.
3. Performance improvement for triangles — the triangle rendering algorithm has been optimized to improve performance.
4. Nop command — a nop command, `GP2_NOP`, has been added to the GPSI command set.
5. Colormap flashing can be avoided on —CXP's `GP2_SET_CMAP_OFFSET` command can be used to set the colormap size to a value that is not a power of two. This can be used to avoid excessive colormap flashing in some cases.
6. New options for `GP1_SET_ZBUF` — the `GP1_SET_ZBUF` command now has an additional flag. The command can now set the Z-buffer, clear the screen, or do both simultaneously.
7. Depth cued polygons — depth cued polygons have been added to the GPSI command set.

The new functionality will only work on GP2-based hardware. The commands will be ignored on GP and GP+ based systems.

Documentation

For more information on these new GPSI commands, see the *GPSI Programmer's Guide Addendum.*

SunView

This section describes any enhancements to the SunView interface added by the SunOS release 4.0.3 - GFX release. Enhancements are briefly described here.

Pixwins

The Pixwin library has a number of enhancements. They include support for non-power-of-two colormaps, lookup table support, and two new plane groups described in the above Pixrect Library section.

Colormaps

There is a new SunView flag to support colormaps with sizes that are not an exact power of 2. The `PWCD_SET_CMAP_SIZE` flag is defined in `/usr/include/sunwindow/pixwin.h`. Setting this flag in the `pw_clipdata->pwcd_flags` structure of a canvas defines a new definition for the default plane mask. The previous action was to set the planemask to the colormap size minus one (`cms.cms_size - 1`). This action is still followed if the `PWCD_SET_CMAP_SIZE` flag is not set, thus insuring complete backward compatibility. However, if the new flag is set, then the plane mask is set to the next power of 2, greater than or equal to the size of the colormap, and then one is subtracted.

To set the flag, do a logical "or" of `PWCD_SET_CMAP_SIZE` with the value of `pw_clipdata->pwcd_flags`.

NOTE *The application writer using this new flag could accidently render with colors that are undefined for the canvas. The window system software will* **not** *check for this error when the* `PWCD_SET_CMAP_SIZE` *flag is set.*

This flag is primarily introduced in an attempt to reduce the amount of colormap flashing that can occur. In conjunction with this new flag, the `/usr/include/sunwindow/cms_colorcube.h` header file has had some new macro definitions added to it.

```
#define CMS_COLORCUBE_SHIFT      "colorcubeshift"
#define CMS_COLORCUBE_SHIFT_SIZE          225
#define cms_colorcubeshiftsetup(r,g,b)
#define cms_colorcubesetupshift_gamma(r,g,b,gamma)
```

The `colorcubeshift` definitions only use the 5-9-5 portion of the colorcube definitions, excluding the black-white portions and the grayscale ramp. The resulting colormap is 227, instead of 256, entries long. This allows some overlap of colormap segments before flashing begins.

To get accelerated support (on GP2-based machines) for the new colormap size, the new GPSI command, `GP2_SET_CMAP_OFFSET` must be used in conjunction with the SunView flag mentioned above. For more information, see the above *GPSI* section, or refer to the *GPSI Programmer's Guide Addendum.*

7.3. New Software: Network Changes

RFS (Remote File Sharing) for SunOS release 4.1

RFS is a distributed filesystem developed by AT&T that allows workstations running the UNIX operating system to transparently share files over a network. To make files available to other systems over RFS, a machine advertises a local directory using the RFS name service. A client may then mount the advertised RFS directory as it would other filesystems. Once mounted, files in the RFS filesystem can be manipulated in any way that a local UNIX file would be.

In SunOS release 4.1, RFS is an optional software installation category on the release tape. Selection of this category during installation provides the capability to act as both a server and a client of RFS filesystems. Use of RFS does not affect the use of other filesystem types; for example, a machine can simultaneously access and serve RFS, NFS, and UFS filesystems.

As with any network service, RFS must run on an underlying protocol. For SunOS release 4.1, RFS is supported by a version of the TCP protocol that conforms to TLI. This protocol support is provided as part of the RFS installation option.

Any system running the SunOS release with RFS installed can transparently share files with a system running an AT&T System V Release 3.2-compatible version of RFS (assuming the system running 3.2 is using the TCP/IP protocol).

See the *System and Network Administration* (Part Number 800-3805-10) manual for details.

TCP/IP Configuration Control

The following table lists the parameters that can be changed in the SunOS release 4.1 TCP/IP software. For a detailed discussion of TCP/IP, and a detailed discussion of the new TCP/IP functionality, see the *System and Network Administration* manual (Part Number 800-3805-10).

All parameters are stored in the file `netinet/in_proto.c` in the kernel build directory (usually `/usr/share/sys`).

The first column of the table gives the variable name, and the second column lists the default value. Entries in the third column, when present, are options that can be set in the kernel configuration file.

Table 7-2 *TCP/IP Default Parameters*

Variable Name	Default Value	Options
`ip_forwarding`	0	`IPFORWARDING`
`ip_subnetslocal`	1	`SUBNETSARELOCAL`
`ip_sendredirects`	1	`IPSENDREDIRECTS`
`ip_dirbroadcast`	1	`DIRECTED_BROADCAST`
`ip_printfs`	0	
`tcp_default_mss`	512	
`tcp_sendspace`	4096	
`tcp_recvspace`	4096	
`tcp_keeplen`	1	
`tcp_ttl`	30	
`tcp_nodelack`	0	
`tcp_keepidle`	14400	
`tcp_keepintvl`	150	
`udp_cksum`	0	
`udp_ttl`	30	
`udp_sendspace`	9000	
`udp_recvspace`	18032	

`uucp` Upgrade to Honey/DanBer

uucp has been upgraded to a more modern version based on System V Release 3 (the Honey DanBer uucp). System administration and control files related to uucp will need to be converted or replaced with new versions. See the *System and Network Administration* manual (Part Number 800-3805-10) for details of upgrading to the new version of uucp.

`showfh`(8C), `rpc.showfhd`: New Diagnostics

`showfh` and `rpc.showfhd` are new NFS diagnostics that allow users to map NFS file handles to file names. `rpc.showfhd` is the version that runs on servers. See the `showfh`(8C) and `showfhd`(8C) manual pages for more information.

Changes for Network Performance

- The TCP implementation has had several performance improvements. Larger segment sizes are used, such as 1460 bytes on Ethernet instead of 1024. The handling of networks with large Maximum Transmission Units (MTU) has been improved.
- The FTP program uses larger buffer and TCP window sizes. The kernel has been tuned to improve performance with larger buffers and window sizes.
- The Internet checksum calculation on the SPARC architecture is now about twice as fast as in previous releases. This makes the use of TCP and enabling UDP checksums more practical.
- The Ethernet drivers will no longer send "trailer" packets. The `trailer` and `-trailer` options on the `ifconfig`(8) command are still supported for compatibility, but the flag they control no longer has any effect. This

makes the Ethernet drivers slightly smaller and faster. Trailer packets are still accepted as input, so there should be no compatibility problems.

- The LANCE Ethernet (le) driver has been rewritten to make it smaller and faster. It can now chain multiple Ethernet packets for transmission at once.
- NFS clients now dynamically determine timeout values, by estimating the average and deviation of measured response times. This results in fewer retransmissions when mounting from a slow or overloaded server. When routing across networks, the transfer sizes are also adjusted automatically, to handle congestion better on wide-area networks.
- The user level RPC clients, which use UDP, now back off exponentially. This results in fewer retransmissions when making a call to a slow or overloaded server.
- Address Resolution Protocol (ARP) packets are limited to one every second, to reduce unnecessary traffic when servers are temporarily down. Ethernet error messages are limited to one every two seconds, so that useful work can be done even when the network has serious problems. When an output queue overflows, a random packet is dropped instead of always dropping the last one. This makes congestion control more fair.

Network Management Changes

- The `ifconfig`(8) command now has the `-a` option to operate on all interfaces at once. It will print out the Ethernet address if available (super-user only).
- The IP header can be sent by specifying a protocol number of zero on a raw IP socket. This can be used by network management tools.
- The `etherfind`(8) program now includes more protocol knowledge, such as NFS, NIS name server, Domain Name Service, IP options, and others. These are available through the `-r` option.
- The `ping`(8) program now includes options to use loose source route and record route IP options, `-l` and `-R`. In verbose mode, it prints more information on ICMP packets that it receives.

See the appropriate manual pages in the *SunOS Reference Manual* (Part Number 800-3827-10) for details.

NIS (YP) Improvements

Note: As explained in Chapter 2 of this manual, *Network Information Service* (*NIS*) has replaced "YP." The names of `yp` commands however, such as `ypbind` and `ypinit` will remain unchanged. The following improvements have been made to this service since SunOS release 4.0:

- NIS server is now implemented with `ndbm`(3) library routines, resulting in faster times for all data access routines.
- NIS binder is more secure. `ypbind` must be invoked with a flag in order to allow the binding to be explicitly set.

- The ability to use the Domain Name System resolver from NIS (through the `makedbm -b` option) has been improved. The NIS no longer forks on each request, and returns soft error codes† and multiple addresses to NIS clients.
- The NIS client library and `ypbind` have been streamlined to use about half as many system calls in many operations. This allows higher loads to be handled much better.
- A new daemon, `ypxfrd`, allows for faster propagation of NIS maps from the master servers to the slave servers (the "push" process), by transferring the maps in larger segments.
- The internet Domain Name System support has been upgraded to BIND version 4.8, plus many bug fixes. The name server now invokes the `/usr/etc/in.named-xfer` program to do zone transfers, resulting in higher availability. See *Chapter 22, Administering Domain Name Service,* in the *System and Network Administration* manual (Part Number 800-3805-10) for details.
- `passwd`(1) now changes the NIS `passwd` if there is no local entry. `ypchsh`(1) and `ypchfn`(1) have been added to allow changing the shell and finger entries in the NIS `passwd` database.
- The `revnetgroup` program used by the NIS Makefile for building the netgroup NIS maps was changed to read from standard input instead of being hardcoded to reading `/etc/netgroup`.

Networking Improvements for Small-Memory Machines

The following improve performance on small-memory (4 MB) machines:

- The NFS client code now dynamically allocates client handles, and returns them to the kernel memory pool when not in use.
- One kind of network buffers (cluster mbufs) are returned to the kernel memory pool when not in use.
- The number of Ethernet controllers of each type, and the number of buffers that they use, can now be easily changed when kernels are built. For example, instead of always supporting exactly two "ie" drivers (`ie0` and `ie1`), any number can be configured in software (the limit depends on your hardware configuration). In the normal small-memory configuration (only one interface; the NFS server not configured into the kernel), the default is to dedicate many fewer buffers to the Ethernet driver.

7.4. TFS (Translucent File Service) for NSE

The Translucent File Service is a new copy-on-write filesystem that allows users to share file hierarchies while providing each user with a private hierarchy into which files are copied as they are modified. Because it appears to be a standard filesystem, a program does not need special knowledge to use the TFS.

†"soft error codes" indicate that the DNS servers are unavailable or overloaded and the process should be retried later.

The TFS allows users to mount a private, writable filesystem in front of any number of public, read-only filesystems in such a way that the contents of the public filesystems remain visible behind the contents of the private filesystem. This is useful when a user wants to install a local version of a binary in a directory that is mounted from another machine.

See the *System and Network Administration* manual (Part Number 800-3805-10) for a complete description of TFS, and the `tfsd`(8), `mount_tfs`(8), `umount_tfs`(8), `lsw`(1), and `unwhiteout`(1) manual pages for details of these new commands.

7.5. Compiler Modifications

`libm` Support for 4.1 C Compiler Changes

Assembly language subroutines and inline expansion template files have been revised for SunOS release 4.1:

Instruction Scheduling

Assembly language functions and inline expansion templates have been rewritten to take better advantage of the MC68882 (`sun3x` -f68881) and the TI 8847 (`sun3` `-ffpa` and `sun4`).

Fortran COMPLEX Code Generation

Fortran COMPLEX code generation, implemented as part of the C compiler, now passes COMPLEX function parameters that result as pointers to structures in memory.

The new run-time library entry points have been renamed.

`libm` includes new trigonometric functions for SunOS release 4.1 that express their arguments as multiples of pi. This enhances fast and accurate reduction of arguments of all sizes. See the `sin(3M)` manual page for more information.

`Sun3 cc -O` Default to `-O2`

The `sun3` C compiler now treats the `-O` option as equivalent to `-O2`. Thus, `-O` now invokes a global optimizer on both Sun-3 and Sun-4 machines.

Global Optimizer Improvements

The SunOS release 4.1 global optimizer uses less time and space than its 4.0 equivalent, and no longer aborts when optimizing programs that require more than 2MB of stack space at compile time.

NOTE: Programs with large procedures (greater than about 1,000 lines) may cause the optimizer to run out of memory if compiled at the `-O3` level. It is recommended that large programs be compiled at an optimization level of `-O2` or lower.

In SunOS release 4.1, Functions Without Return Statements May Yield Different Results

Functions that end without executing an explicit return statement may not produce the same results under SunOS release 4.1 that they did under SunOS release 4.0. *This is intentional*; C does not define the value of a function that ends without an explicit return statement.

NOTE: If `main()` ends without either returning a value or passing a status to `exit()`, an undefined result will be returned to the invoking environment (such as the shell). Consequently, if the same program is used in a Makefile, `make` may halt with a nonzero error status.

New `-dalign` Option For Better Access to Double-precision Floating-point Data

The new `-dalign` option forces generation of `ldd/std` instructions for efficient access to double-precision floating-point data, even in situations where actual alignment of the referenced storage is unknown. Performance of most floating-point-intensive programs can be improved considerably by use of `-dalign`.

NOTE: `-dalign` is not the default. ANSI FORTRAN 77's alignment rules do not permit compilers to force double word alignment of double-precision data; REAL*4 and REAL*8 use the same alignment rules. To remain FORTRAN compatible, the default code generation rules assume only 32 bit alignment for C data of type `double`.

The C compiler guarantees double-word alignment for all auto, static, and external variables of type `double`.

NOTE: Function arguments of type `double` are *not* guaranteed to have double-word alignment.

Loop Unrolling at `-O3` and `-O4` Optimization Levels

The compiler will unroll some C `for` and `while` loops at the `-O3` and `-O4` optimization levels. Unrolling will occur for small floating point intensive loops on machines with schedulable FPU's, that is, SPARC, Sun-3 FPA, 68882. Sufficiently small memory-intensive loops are unrolled at `-O3` and `-O4` on all Sun architectures.

The compilers do not unroll loops at `-O2` (the default optimization level when `cc -O` is specified).

Improved Floating Point Instruction

The SunOS release 4.1 compilers do improved floating point instruction scheduling for SPARC, Sun-3 FPA, and 68882/68881.

There are numerous code generation improvements and bug fixes for both the Sun-3 and Sun-4 compilers.

7.6. New Software: Using and Writing Device Drivers

New DVMA Allocation

SunOS release 4.1 contains a new set of routines that device drivers can use to allocate DVMA space for I/O transfers. These routines are a move toward separating the allocation and maintenance of DVMA resources from the complex framework of the mainbus (`mb`) structures; they also simplify matters in the case when no DVMA space can be allocated. The older `mbsetup()` and `mballoc()` interfaces are retained for compatibility with current drivers, so use of the new routines is entirely optional. See the *Writing Device Drivers* manual (Part Number 800-3851-10) for more information.

`mt`(1): New Options

The `mt`(1) command supports these new options. For complete information, see the updated manual page.

bsf Back space over *count* file marks. The tape is positioned on the beginning-of-tape side of the file mark.

nbsf Back space *count* files. The tape is positioned on the first block of the file. This is equivalent to *count* +1 `bsf`'s followed by one `fsf`. **nbsf** replaces **bsf** (from release 4.0.3) for the `st` driver; it is new for the `xt` driver.

asf Absolute space to *count* file number.

eom Space to the end of recorded media on the tape. This is useful for appending files onto previously written tapes.

erase Erase the entire tape; this now supports 1/2-inch tape drives.

See the `mtio`(4), `st`(4S), and `xt`(4S) manual pages in the *SunOS Reference Manual* (Part Number 800-3827-10) for details.

7.7. New Software: Kernel Use and Development

`savecore`(8): Abbreviated Kernel Crash Dumps

SunOS release 4.1 utilizes a new condensed crash dump scheme. On a system panic, only the relevant pages are written to the dump area, instead of the entire physical memory. "Relevant pages" are defined as pages in use by the kernel, pages belonging to the active process at the time of the crash, and stack pages for all the other user processes. This results in crash dumps of greatly reduced size, compared to those without the condensed `savecore` dump scheme. Typically dumps under `savecore` are 3 to 4 MB in size, independent of the amount of physical memory present on the system. Thus even a 16MB workstation can usually successfully dump to a 14MB swap partition.

At times it may be desirable to force a dump of all memory, whether "relevant" or not. Patching the kernel variable `dump_allpages` to `1` will force the dumping of all pages on a crash, subject to the size of the dump area. All relevant pages will be dumped, and the remaining pages dumped as space permits.

`savecore`(8) is a program that copies a dump image from the dump area into the filesystem. It now understands the new condensed dump files exclusively, and no longer supports SunOS release 4.0.3 and earlier non-condensed versions of crash dump files.

When `savecore` processes a condensed crash dump, the `vmcore.N` file it creates is a condensed file. `libkvm` will recognize that the `vmcore.N` file is condensed and do the necessary relocation of physical addresses. No changes to `adb`, `ps`, or any other utility that uses `libkvm` to examine crash dumps were necessary. User programs that use `libkvm` to examine crash dumps only require relinking with the new `libkvm`. A read of an unsaved page will return zeros; a write will be ignored. `libkvm` continues to work with non-condensed core files; `adb -k /vmunix /dev/mem`, for example, still works.

NOTE: As shipped, `savecore` must be invoked manually shortly after reboot to save the crash dump from being overwritten by the paging system. The instructions that automatically invoke `savecore` during reboot are commented out of `/etc/rc.local`. These lines would store the crash dump in the `root` partition and this may not be desirable. See "Handling System Crash Dumps" in the *Sun System & Network Manager's Guide* (SPARCstation 1 only) and the *Administering Workstations* section of Chapter 8 in the *System and Network Administration* manual (Part Number 800-3805-10), and `savecore`(8) for instructions on enabling and tailoring your system to save the crash dump in an appropriate partition.

`crash`(8): Interpreting Kernel Data

`/usr/kvm/crash` has been ported from System V to SunOS release 4.1; it is a friendly, interactive, and relatively comprehensive program used to interpret kernel data structures on running systems. `crash` will display system and process memory information in suitable formats.

This program will be of use mainly to system administrators and kernel developers.

For more information, see the `crash`(8) manual page and the *System and Network Administration* manual (Part Number 800-3805-10).

`modload`(8): Loading Software Modules On a Running Kernel

In SunOS release 4.1, the ability to load and unload software modules from the kernel of a running system has been added to all Sun architectures. The `modload`(8) and `modunload`(8) programs provide this function. This functionality is now available for all Sun architectures; in earlier releases, they were available on SPARCstation 1 and Sun386*i* configurations only.

This new functionality allows software developers to distribute software that can be linked into the kernel with a single `modload` command. Device driver

writers, for example, can now add the software interface for a new device to the kernel without the usual complex, tedious, and time-consuming procedure of building a new kernel. This feature is useful to anyone who wants to load, test, and use functionality on a system without having to reboot. This procedure will shorten and simplify the process of adding software modules to the kernel, but if the added module breaks the existing software, the familiar reboot will be required.

The `-exec` *exec_file* option to `modload` specifies a shell script to be executed after the module is successfully loaded. A common use for this script is to create entries in `/dev` for a newly loaded driver.

The `-conf` option specifies the configuration information (device address, interrupt vector, and priority, and optionally the device number) needed to install a driver.

See the *Writing Device Drivers* (Part Number 800-3851-10) and *System and Network Administration* (Part Number 800-3805-10) manuals and the `modload`(8), `modstat`(8), and `modunload`(8) manual pages for details.

7.8. Internationalization Features

This section is an overview of the Internationalization features of SunOS release 4.1. For a detailed treatment of the subject see the *System Services Overview* manual (Part Number 800-3846-10).

8-bit Cleanup

The following table lists commands that have been modified in release 4.1 to make them 8-bit-clean.

Table 7-3 *8-bit Clean Commands*

Commands Modified in Release 4.1			
`automount`	`cat`(sysV)	`cat`(ucb)	`cmdtool`
`col`(sysV)	`col`(ucb)	`csh`	`diff`
`dd`	`ed`	`ex`	`file`
`fmt`	`fontedit`	`iconedit`	`mailtool`
`ls`(sysV)	`ls`(ucb)	`mail`	`mount`
`nawk`	`od`(sysV)	`od`(ucb)	`pg`
`pr`(sysV)	`pr`(ucb)	`ps`	`sed`
`sendmail`	`sh`	`shelltool`	`stty`(ucb)
`stty`(sysV)	`textedit`	`umount`	`vi`
`wc`	`write`		

The following table list commands that are not yet 8-bit clean. All commands not on this table or the previous table are assumed to have been 8-bit clean in previous releases.

Table 7-4 *8-bit Dirty Commands*

8-Bit Dirty Commands			
adb	addbib	as	awk
catman	cc†	cflow	cpp
ctags	cxref	dbx	dbxtool
deroff	dis	keylogin	keylogout
lex	lint	login	logname
m4	man	newgrp	nroff
passwd	refer	rlogin	rmail
rusers	rwho	sdb	spell
strings	su	troff	users
w	who	whoami	whois
yacc			

†Supports 8-bit characters in strings and comments.

Fonts for Extended ASCII

The following fonts have been extended from ASCII to ISO 8859-1:

serif.r.[10,11,12,14,16,18]
screen.r.[11,12,14,16]
screen.b.[12,14,16]
cour.r.[10,12,14,16]
cour.b.[12,14,16].

European characters can now be entered with the Type-4 keyboard, and can be printed on PostScript printers by using the Transcript 2.1.1 application.

Kernel Changes (all 8-bit clean)

Support for non-ASCII code includes the following:

- Data path through system calls
- tty driver and pseudo-tty driver
- Support for the Type-4 keyboard

Command Changes (all 8-bit clean)

Any single-byte, 8-bit encoding scheme can now be used through almost all SunOS utilities. This more than meets the X/Open Portability Guide requirements.

It is important to understand what the term "8-bit clean" means. It means that the standard commands shipped with the OS (with the exceptions noted above) are now capable of reading, writing, and processing characters that use all bits of the single byte to represent information. Prior to 4.1 this was not possible.

8-bit clean does not imply that the commands will understand any other aspect of a "locale" or language requirement. For example non-English collation sequences and non-English messaging are not yet supported in any SunOS command.

Support for non-standard 8-bit code sets

SunOS release 4.1 assumes that the 8-bit codeset of choice is ISO 8859/1 (ISO Latin 1 code set). It is possible for other 8-bit code sets to be supported under SunOS release 4.1, but in order to do so you must customize the workstation to some degree. There are no plans to add implicit support for any other 8-bit code set other than ISO 8859/1. In particular, for non-standard 8-bit code sets you will have to:

- Create a new family of fonts for your code set,
- create a collection of new locale tables,
- name and set up the locale for your environment, and
- perform code set mapping as and when necessary.

Support for Non-standard Peripherals

There is no implicit support in SunOS release 4.1 for peripherals that expect to use code sets that are not based on ISO 8859/1. However, it is not too difficult a task to perform code set mappings at the required point of data I/O. If at all possible, your application should attempt to keep to the 8859/1 standard when dealing with 8-bit characters.

Library Changes (all 8-bit clean)

New library routines exist that allow SunOS release 4.1 to be XPG-2 and IEEE Std. 1003.1-1988 (POSIX.1) (Internationalization features) compliant.

Type-4 Keyboard Support

Only the Type-4 Keyboard allows users to enter European characters. See the *SunView 1.80* section of this Chapter, `loadkeys`(1) in the *SunOS Reference Manual* (Part Number 800-3827-10), and the *System Services Overview* (Part Number 800-3846-10) for more information.

Sunview 1.80 supports all of the European keyboards inside all of the Sunview tools. In order for you to be able to use `cmdtool` or `shelltool` with any of the non-ASCII characters generated by these keyboards you must enable an "8-bit" data-path. By default under SunOS release 4.1 the data-path is set to 7-bit mode for backwards compatibility. To handle characters generated by the non-U.S keyboards or by the compose key of any keyboard, the following two lines must be placed in either your personal `.profile` file (Bourne shell)

```
LC_CTYPE=iso_8859_1;export LC_CTYPE
stty pass8
```

or, in your personal `.cshrc` file (for the csh):

```
setenv LC_CTYPE iso_8859_1
stty pass8
```

7.9. SunView 1.80

The major features of SunView 1.80 are:

- an **online help mechanism***, allowing application developers to provide Spot Help for their users,
- **programmable alarms*** for dramatically notifying users,
- keyboard support
  - Type-4 keyboard
  - upgraded description of the `.textswrc` file*
- enhanced color capabilities
  - **colored panel items***
  - support for **24-bit true color***
- changes to the defaults database
- several user changes
- various bug fixes*.

Online SunView Help

The new release of SunView offers two related mechanisms for providing online help to users

- Spot Help, a cursor-position sensitive facility to display one 32 × 80 character panel of online help,
- More Help, to provide additional information when the one panel of Spot help is not enough.

To get help, the user places the pointer over the object (panel, button, etc) of inquiry and then strikes the Help key. Whatever information is available is then displayed.

* See the *SunView 1.80 Update*. This new publication, part number 800-4738-10, is an appendix to the *SunView Programmer's Guide* offering a more a detailed description of these features. It contains the update information for both the *SunView Systems Programmer's Guide* as well.

Help Keys

On a Type-3 keyboard, the Help Key is Meta-/, obtained by pressing the Meta key and the / key at the same time. There are two Meta keys, which are immediately to the left and right of the long space bar on the bottom center of the keyboard.

On a Type-4 keyboard, the Meta-/ combination works, and there is an explicit Help key also.

- The Meta keys are in the same place as on the Type-3 keyboard, beside the space bar, marked with a diamond: ♦.
- Help is the double-width key located at the bottom of the left hand block of function keys (the ones labeled Stop, Again, etc.)

Limitations of Spot Help

There are two limitations to the use of Spot Help on SunView 1.80:

- At this time, **only the mechanism** for Spot Help is provided; *no actual Help Text is provided.* Engineering priorities do not allow the development of Help Text for SunView itself, but the mechanism is being made available to developers who want to provide cursor-position sensitive help in their applications.
- Also note that Spot Help supports a single window of text, 32 lines by about 80 characters (longer lines are not supported at this time). To obtain longer messages, you must use the More Help feature. This is a user-implemented feature called by the **More** button on the help window.

Programmable Alarms

SunView 1.80 provides programmable alarms, which "beep" and "flash" at the user in a way that is settable from either a C program or from shell commands.

CAUTION **SunView must be installed *and be running* for the alarms to occur, even though you can manipulate the environment variable without SunView.**

A beep is the sounding of the bell on the user's keyboard. A flash is a color reversal in a window; the window frame is repainted with the colors reversed, and then painted again normally.

- The number of beeps and the number of flashes can be independently set.
- There is one setting, however, for the duration of both beeps and flashes, which is also the interval between successive beeps/flashes.

Note that the `defaultsedit(1)` values for `SunView/AudibleBell` and `SunView/VisibleBell` will determine whether beeps and flashes, respectively, occur at all. If disabled by the indicated default, that aspect of the alarm will not occur, no matter what the alarm setting is.

Command Interface to Alarms

SunOS 4.1 provides shell commands to set and get the characteristics of the alarm, and to ring it. These commands rely on an environment variable:

```
% set WINDOW_ALARM=:beeps=b:flashes=f:dur=t:
#   where
#   b    = number of beeps
#   f    = number of flashes
#   t    = duration of each beep/flash in milliseconds
```

The setting of this variable can be performed either directly, or through the command:

- **set_alarm** [**-b** *b* **-f** *f* **-d** *t*]

where the option arguments correspond to the fields in the environment variable.

There is a counterpart command that returns the setting, in the form shown above:

- **get_alarm**

And there is a command to actually ring the alarm:

- **ring_alarm**

This command gets the attributes from `WINDOW_ALARM` and rings the alarm with these attributes. The alarm's behavior is controlled by the SunView `defaultsedit(1)` entries `SunView/Audible_Bell` and `SunView/Visible_Bell`, so the sound and flash can be disabled by the user, regardless of `WINDOW_ALARM`.

Program Interface to Alarms

The C program interface to the alarms uses a SunView attribute:

- `WIN_ALARM`

a data structure:

- `alarmval`

and several function calls:

- window_get(window, WIN_ALARM);
- window_set(window,WIN_ALARM,&alarm,0);

Keyboard Support

Type-4 Keyboard

In SunOS 4.1, SunView 1.80 supports the Type-4 keyboard. Previously, a Type-4 keyboard could be connected to a Sun workstation but only operate it as a Type-3 keyboard. Now it can be used to full capacity.

Internationalization

There is one physical/electrical Type-4 keyboard with a number of international variants. These variants are achieved by:

1. replacing selected plastic key caps at the factory, so that the appropriate characters are visible on the keyboard,
2. selecting (or modifying) the appropriate character translation table.

The translation tables are described in `keytables(5)` and are selected with the `loadkeys(1)` command.

Numeric Keypad

On the Type-4 keyboard, there are more right hand function keys than on the Type-3 keyboard. The additional keys can be used as a numeric keypad that accountants and bookkeepers are familiar with.

The key in the upper right corner, [NumLock], toggles the right hand keys between an ASCII mode and a function key mode. In the ASCII mode, the keystrokes are interpreted as a numeric keypad, with the values displayed on the top of the plastic keycovers ([=] , [/] , [*] , and so on). In the function key mode, the keys are interpreted in a similar manner to the right-hand keys on a Type-3 keyboard. That is, they can be assigned functions via the `.textswrc` file or have the SunView functions mapped on them by selecting the `/Input/Left-Handed` option in the `Defaults Editor`.

Turning on [NumLock] turns off the arrow keys and any functions assigned to the right keypad keys.

The mode is displayed to the user by a light emitting diode (LED) above the keypad, labeled "NumLock". This light is lit to indicate that the right keypad is in ASCII mode; unlit, it indicates function key mode.

New Function Keys

Three new function keys are available (and can be assigned functions): keys [F10] , [F11] , and [F12] at the top of the keyboard. Functions are assigned to these keys through the `.textswrc` file.

Key Clicks

When each key is struck, the Type-4 keyboard makes a short clicking sound in addition to the sound of impact when the key strikes down. The `click(1)` command is used to turn this feature on and off:

```
#turn on
%click -y
#turn off
% click -n
```

`.textswrc` **file**

In the *SunView 1.80 Update* appendix to the *SunView Programmer's Guide*, there is a discussion of using the `.textswrc` file to assign functions to the function keys. This topic was not previously considered in depth.

Keyboard Device Driver Compatibility

The keyboard STREAMS module, normally active on every Sun workstation keyboard port, has undergone some changes to support the new non-U.S. keyboard layouts. These new keyboard variations, in effect, generate non-ASCII keystation

codes. The compatibility issues associated with the new STREAMS modules changes are discussed below.

Binary compatibility

The SunOS release 4.1 keyboard STREAMS module (`kbd.o`) is fully binary compatible with all previous versions. All ioctl support is binary compatible.

Source Compatibility

The following special considerations apply when recompiling source code that directly issues keyboard driver ioctls:

- The `KIOCLAYOUT` ioctl is a new ioctl that will return the keyboard layout `id` for any Type-4 keyboard.
- The `KIOCSLED` and `KIOCGLED` ioctls are new. They will set or get the LED lights for any Type-4 keyboard.
- The `KIOCSETKEY` and `KIOCGETKEY` ioctls have been replaced with `KIOCSKEY` and `KIOCGKEY`, respectively. The old ioctls are still supported in binary form.

The internal keyboard translation table has been extended from an array of type `char` to an array of type `short`. The bitmasks for of all "special" entries in the keyboard translation table have changed to be 16 bits in width because of this change. Their unique names have been retained, however (e.g. `BUCKYBITS` is still called `BUCKYBITS`). This reflects itself in the `kio_entry` element of the `kiockeymap` structure, which in turn is pointed to by the argument to the `KIOCSKEY` and `KIOCGKEY` ioctls.

Consequently, the source code for applications that used the `KIOCSETKEY` (`KIOCGETKEY`) ioctl operations must now use the `KIOCSKEY` (`KIOCGKEY`) ioctls. Special attention should be given to the handling of these "special" bits, especially if the application is running the keyboard in `TR_UNTRANS_EVENT` mode (see kb(4)). Under most standard keyboard modes (i.e. `TR_ASCII` or `TR_EVENT`), the application will be supplied with the correct unambiguous information.

Keyboard Compatibility Mode

Normally the keyboard STREAMS module is in "compatibility mode" when it starts up. In this mode, when the keyboard is in the `TR_EVENT` translation mode, ISO 8859/1 characters from the "upper half" of the character set (that is, characters with the 8th bit set) are presented as events with codes in the `ISO_FIRST` range (as defined in <sundev/vuid_event.h>). The event code is `ISO_FIRST` plus the character value. This is for backwards compatibility with older versions of the keyboard driver. If compatibility mode is turned off, ISO 8859/1 characters are presented as events with codes equal to the character code.

7.10. Software Functionality Added in SunOS Release 4.0.3

`fdformat`(1): Utility for Formatting Diskettes

The new `fdformat`(1) utility supports floppy disk formatting on the Sun-3/80 and SPARCstation 1. See the `fdformat`(1) manual page for a full description of `fdformat` features.

7.11. Software Functionality Added in SunOS Release 4.0.3 - 4c

The `sundiag` Program

The `sundiag`(8) program is a SunView-based online system exerciser for testing peripheral devices. SunDiag can be used to test any of Sun's graphics hardware and peripherals (for example CG6, CG9, CG8, GP2, FPU2, SunDials, SunButtons) on configurations running SunOS release 4.0.3 or later.

`sundiag` executes system tests that formerly were performed by `sysdiag`, and runs all `sundiag` tests written for new Sun products. See `sundiag`(8) in the *SunOS Reference Manual* (Part Number 800-3827-10), and the *SunDiag User's Guide* manual (Part Number 800-3818), for details. NOTE: SunDiag is loaded on your system only if you have installed the optional `User_Diag` category of the SunOS.

New and Changed `/usr` Directories

See the *System and Network Administration* manual (Part Number 800-3805-10) for changes to the `/usr` directory structure.

7.12. Programs No Longer Supported

SunCGI™ and SunCore™ End of Life

The two bundled graphics products, SunCGI and SunCore, were phased out of SunOS in release 4.0.3. The `Graphics` category of `suninstall` contains software for programmers who want develop graphics-based applications on a SunView environment. The functionality of SunCGI can be found in SunGKS, while the functionality of SunCore can be found in SunPHIGS. It is recommended that all new development requiring a graphics package be based on either SunGKS or SunPHIGS and existing applications be ported to those products.

Neither SunCGI nor SunCore will be ported to any new systems or frame buffers. The Answer Center no longer supports SunCGI or SunCore, except for questions about porting to SunGKS and SunPHIGS. Bug reports on SunCGI and SunCore will no longer be accepted. The Answer Center will continue to support both

SunGKS and SunPHIGS. The last release which supported SunCGI and SunCore was SunOS release 4.0.3 - 4c; customers may now purchase SunCore or SunCGI source code from the Consulting Services group.

`sysdiag` — The `sysdiag` utility is replaced with the new `sundiag` utility, a window-based diagnostic exerciser. See the discussion of `sundiag` in the section above for more information.

8

New Hardware

8.1. Hardware Introduced In SunOS release 4.0.3-4c

The SPARCstation 1 Desktop Workstation

The SPARCstation 1 is a small, powerful, flexible, inexpensive workstation, built around the Scalable Processor ARChitecture (SPARC) CPU. It is compatible with the Sun-4 family of workstations and servers.

Features of the SPARCstation 1 include:

- 20 MHz SPARC integer unit, delivering 12.5 MIPS
- 1.5 MFLOP single-chip floating point co-processor
- 8-16 MB of RAM, expandable to 64 MB using 4Mbit SIMM modules
- 64 KB cache memory
- 3 expansion slots
- GX graphics accelerator
- up to 2 internal 100 MB SCSI drives
- 1.44 MB IBM-compatible floppy disk drive
- 8-bit digital audio capability

See *Appendix D: SPARCstation 1-specific Information* of this manual for more information SPARCstation 1 users should know when running SunOS release 4.1, and details on the SPARCstation 1's capabilities and features.

8.2. Hardware Introduced In SunOS release 4.0.3

Sun-3/80, Sun-3/470, Sun-3/480: MC68030-based Desktop Workstations

The Sun-3/80 Desktop Workstation

The Sun-3/80 is a low-cost desktop system featuring optional integrated mass storage and an optional floppy disk. Based on a 20 MHz Motorola 68030 processor with a base configuration of 4 megabytes of dynamic random access memory (DRAM), the Sun-3/80 is binary-compatible with most applications based on other Sun-3 products. This compact system has a wide variety of configurations.

The Sun-3/80 has many attractive features for users preferring the familiar 680*x*0 architecture:

- Compact packaging
- Motorola 68882 floating point unit (standard)
- Optional integrated floppy disk
- Optional integrated 3-1/2″ SCSI hard disks
- Many frame buffer and graphics accelerator options
- External mass storage expansion capability

Floppy for the Sun-3/80

SunOS release 4.1 supports the 3-1/2″ flexible diskette (floppy disk) drive used in the Sun-3/80 Workstation. This floppy is an industry standard, non-SCSI device with a 1.44 megabyte capacity.

The floppy drive is used in the same ways as you use other disks. For example, the floppy disk can contain a UNIX file system and can be mounted like any other disk partition. In addition, floppy disks can be used to boot the system and to copy application software from diskette to your system. SunOS release knows the floppy as `/dev/fd0`.

Sun-3/470 Deskside Workstation and Sun-3/480 Server

The Sun-3/470 and Sun-3/480 are based on the 33 MHz Motorola 68030 processor. Both systems provide many important features and offer a variety of expansion options:

- 33 MHz Motorola 68882 floating point unit (standard)
- Optional high-performance floating point accelerators
- Many frame buffer and graphics accelerator options
- 60 and 150-megabyte 1/4″ tape drives

- 1/2" 6250/1600 bpi tape drive
- Up to 1.3 gigabytes of SCSI disk storage
- Any combination of 4 SMD-4 controllers and 8 SMD disk drives on the Sun-3/470
- Any combination of 4 SMD-4 controllers and 16 SMD disk drives on the Sun-3/480

Differences Between `sun3` (MC68020-based) and `sun3x` (MC68030-based) Workstations

The new MC68030-based `sun3x` computers (Sun-3/80, Sun-3/470, and Sun-3/480) are members of the Sun-3 family and run Sun-3 user-level applications unchanged. However, while the `sun3` uses the MC68020 and the Sun MMU, the `sun3x` uses the MC68030 with an on-chip MMU. This difference necessitates a unique kernel for each of the two system architectures.

For detailed information about the MC68020, see the *MC68020 User's Manual, MC68020UM/AD Rev 1* and the *Sun-3 Architecture Manual.* To understand the MC68030, read the *MC68030 User's Manual, MC68030UM/AD Rev 1.*

Because the MC68020 and MC68030 run the same application binaries, they are said to have a `sun3` *application architecture.* Because they require different kernel-dependent code, however, their *kernel architecture* is different: the MC68020 has a `sun3` kernel architecture, while the MC68030 has a `sun3x` kernel architecture.

The application architecture is displayed by the `arch` command, without options, while the kernel architecture is displayed by `arch -k`. See the `arch(1)` manual page for details.

Following is a list of differences between the `sun3` and `sun3x` architectures that can require you to recompile and relink, or rewrite a program.

User Programs

- Recompile and relink programs that read or write kernel data structures. Recompile programs that depend on kernel data structure offsets or sizes.
- Programs using the MC68020 `callm` and `rtm` instructions do not work on the MC68030 because these instructions do not exist on the MC68030. This should not cause problems, as the standard compilers never generate these instructions.
- Re-port programs that assume that the user stack starts at a particular location. On a `sun3x`, the user stack starts at 0xe0000000. On a `sun3`, the user stack starts at 0x0f000000. In particular, programs assuming the stack pointer is "positive" will not work.

Drivers

- Recompile drivers accessing kernel data structures (such as the user area and the process table). The offsets of a field in these structures can be different between the `sun3` and the `sun3x`.
- Rewrite drivers that "know" about page table format, such as some graphics drivers, to use the 68030 page table format.

- Rewrite drivers with hard-coded kernel addresses to use the correct addresses on the `sun3x` architecture.
- Change drivers that access the system enable register, such as some graphics drivers. This is a `short` on the `sun3x` architecture rather than a `char`.

The following two items apply to the Sun-3/470 and Sun-3/480 only:

- Insert delays for some drivers (for example, the DES driver) to wait for hardware to be ready. The Sun-3/470 and Sun-3/480 machines need these delays because they are faster than other Sun-3 machines.
- For improved performance, rewrite disk and Ethernet drivers (and possibly others) to take advantage of the I/O cache. To do this, mark the buffer with the **B_IOCACHE** flag in the strategy routine if the buffer is aligned properly (16 bytes), and turn this flag off in the interrupt routine when the I/O completes.

Other Differences

The kernel, `kadb`, the boot blocks, the `tftpboot` program for `sun3x` clients, and `libkvm` are different between the MC68020 and the MC68030. Also, `arch`(1) has been enhanced to recognize the difference between `sun3` and `sun3x` kernel architectures.

Compiling Kernel-Dependent Code

The following techniques are recommended when the same kernel-dependent source code is to be compiled for both the sun3 and sun3x targets.

- Use the enhancements to `arch(1)` and `make(1)` in your makefile to define either "sun3" or "sun3x." These `arch(1)` and `make(1)` enhancements are new with SunOS release 4.1, and the following example does not work with the `arch(1)` and `make(1)` commands from previous releases.

```
ARCH:sh = arch -k
CPPOPTS= -D$(ARCH)
CFLAGS= $(CPPOPTS)
```

- As appropriate, use `#ifdef sun3` and `#ifdef sun3x` in your code.

```
#if defined(sun3) || defined(sun3x)
    code identical for the Sun-3 and the Sun-3x
#endif sun3 || sun3x

#ifdef sun3
    code for the Sun-3 only
#endif

#ifdef sun3x
    code for the Sun-3x only
#endif
```

Note that you must explicitly ask for the definition of C pre-processor architecture symbols like `sun3` and `sun3x`, invoking the compiler with `cc -Dsun3` or `cc -Dsun3x`. These symbols are not defined by default.

Also, while `-sun3` is a legal `cc` command-line option for both the MC68020-based Sun-3 and the MC68030-based Sun-3, `-sun3x` is not a legal `cc` command-line option.

SPARCsystem 300 Deskside Workstations and Servers

SPARCsystem 300 Overview

The SPARCsystem 300 is a family of high-performance computer systems based on the 25 MHz Reduced Instruction Set Computer (RISC) processor, SPARC (Scalable Processor ARChitecture). The SPARCsystem 300 packaging options and expansion capabilities meet a diverse set of computing needs.

SPARCsystem 330

The SPARCsystem 330 compact deskside package offers integrated mass storage and backup capabilities. Features of the SPARCsystem 330 include the following.

- High-performance floating point unit (FPU2; standard)
- Many frame buffer and graphics accelerator options
- Three 9U × 400mm VME slots
- Two 6U × 160mm VME slots
- 150-megabyte 1/4″ tape drive
- Up to 1.3 gigabytes of SCSI disk storage

SCSI ID Selection for SPARCsystem 300 Series Boot PROMs

Following are the SCSI configurations supported by the Revision 3.0.1 Boot PROM for the first SCSI host adapter (on the CPU board) and second SCSI host adapter (a VMEbus board). SunOS release 4.1 supports a mixture of SCSI disk and tape drives for SunOS SCSI ID 2 and 3 only when the system has the SCSI host adapter on the CPU board (the first host adapter).

The Boot PROM driver first probes the SCSI devices to determine if there is a disk or tape drive, and then operates correctly from that point.

For example, if you enter one of the following boot commands, you are using the SCSI ID shown in italics.

```
>b st(0,10,0)   (SCSI ID = 2)
or
>b sd(0,10,0)   (SCSI ID = 2)
or
>b sd(0,18,0)   (SCSI ID = 3)
or
>b st(0,18,0)   (SCSI ID = 3)
```

The Boot PROMs will be able to support 4 tape drives at a given time. If 8mm drives (2 gigabytes each) are used, 8 gigabytes of storage space is available. The table below shows supported devices.

SCSI	*First Host Adapter (on-board) Multiplexed disks and tapes*				*Second Host Adapter (VME) No Multiplex devices*	
IDs	*Logical*	*Physical*	*Logical*	*Physical*	*Logical*	*Physical*
0	sd0	sd(0,0,0)	**	**	**	
1	sd2	sd(0,8,0)	**		**	**
2	sd4	sd(0,10,0)	st3	st(0,10,0)	**	**
3	sd6	sd(0,18,0)	st2	st(0,18,0)	**	**
4	st0	st(0,20,0)	**		st2	st(1,20,0)
5	st1	st(0,28,0)	**		st3	st(1,28,0)
6	Floppy (When Available)				**	
7	Host adapter ID				Host adapter ID	

The conventions used in this table are as follows:

Logical: Logical device (used by SunOS in your kernel configuration file)
Physical: The device name used while booting from the Boot PROM
** : Not available

Note 1: At any given time, no more than one device may have the same SCSI ID (2 or 3.) The device may be a disk drive or a tape drive but not both.

Note 2: The probing order for the SPARCsystem 300 series is `sd6` and `sd0`, and so on. (Unchanged from the probing order for SunOS Version 3.0)

Note 3: Possible SPARCsystem 300 series Configurations:

If there is only one host adapter (on-board SCSI):

- 4 disk drives and 2 tape drives.
- 2 disk drives and 4 tape drives.

If the second host adapter is used:

- 4 disk drives and 4 tape drives

Note 4: The SPARCsystem 300 series Boot PROM also supports both 8mm and Front Load Tape 1/2 inch tape drives.

8.3. FPU2 Floating-Point Unit

The FPU2 is a TI 8847-based floating-point unit for SPARCstation 1 and SPARCstation 330/370 workstations. This unit uses an LSI Logic controller chip, TI 8847 arithmetic unit and a 144-pin connector which can be plugged into the

FPU socket on the CPU board. The FPU2 supports the same functionality as the Weitek FPU chip and vice versa. From the user's point of view the two are identical, except for certain operations which the FPU2 unit performs faster (for example, division and square root problems).

Systems with an FPU2 installed run SunOS 4.0/Fortran 1.1 programs unaltered. The inline expansion template file `/usr/lib/sqrt.il`, included in the c tape, may be used to improve performance of SPARCstation 1 on problems that perform multiple square root operations. This inline expansion template replaces calls to `sqrt` subroutines with hardware `sqrt` instructions (although Weitek FPU chip and FPU2 still support `sqrt`). Executables created with these templates may run slower on older Sun-4s without hardware `sqrt` instruction (for example, Sun-4/110 and Sun-4/260 with Weitek 1164/1165).

The following example command lines show how to compile a Fortran or C program with the `sqrt` inline expansion template file:

```
@aboy% f77 -O4 source.f /usr/lib/sqrt.il /usr/lib/libm.il
   or
@aboy% cc -O4 source.c /usr/lib/sqrt.il /usr/lib/libm.il -lm
```

For more information on inline expansion templates, see the `inline`(1) manual page and the *Floating-Point Programmer's Guide* (Part No. 800-1781-01) accompanying SunOS 4.0.

A new utility program searches for the FPU2. This utility, `fpuversion4`(8), determines whether the high-performance floating point components are installed on the system CPU. Detecting the presence of the FPU2, the diagnostic prints a confirming message:

```
Sun-4 floating-point controller version 2 found.
```

9

Graphics Hardware

9.1. CG6 Graphics Accelerator Board

The CG6 board accelerates the performance of many 2D and 3D graphics applications. This P4 color frame buffer uses two ASICs to accelerate production of eight-bit color images.

Supported Graphics Application Software

The following four types of graphics libraries are available for Sun workstations equipped with CG6 boards.

- PixWin (bundled)
- Pixrect (bundled)
- SunGKS (unbundled)
- SunPHIGS (unbundled)

9.2. CG8 24-bit Frame Buffer

The new 24-bit frame buffer, the CG8, uses 24-bit-deep pixels to produce true-color images. The CG8 color lookup table allows for color adjustments, such as gamma correction, on the standard 900x1152 resolution display.

The monochrome overlay plane and enable plane allow displaying either the color or the monochrome plane on a pixel-by-pixel basis.

SunOS release 4.1 provides a software driver and code for enhancements to Pixrects and SunView 1 that take advantage of 24-bit color. The code changes are documented in the *Pixrect Reference Manual* and in the *SunView Programmer's Guide* manual. See also the *Addenda and Errata* at the end of this document for release notes on 24-bit-color support.

Note: Eight-bit indexed color applications must be modified to run with the 24-bit color frame buffer. Unmodified applications will run, but will display images incorrectly—most likely with an all-red screen.

9.3. CG9 24-Bit VME Color Frame Buffer

GP2/CG9

The CG9 frame buffer, when used with the GP2 graphics accelerator board and the proper GPSI microcode, provides accelerated true color graphics support.

Planegroups

Like the CG4 frame buffer, the CG9 frame buffer has three plane groups.

The color plane group provides 24-bits deep, true color displays. The CG9 uses the XBGR format for pixel values. Each pixel is 32-bits deep, consisting of 8 bits for the red, green, and blue components of the pixel color, with the top 8 bits unused. The colors can be adjusted with the frame buffer's lookup table. Unlike an 8-bit color board, the number of different colors that can be displayed is not limited to 256.

The monochrome plane group is used to provide fast text and window system monochrome graphics on the display, without the overhead of writing full 24-bit pixels. It improves text scrolling and window system performance.

In the CG9, the overlay/monochrome plane group has been modified from the original CG4 implementation to provide more functionality.

The overlay and enable planegroups now work together to provide four possible visibility states for a pixel. This scheme allows three possible colors for the monochrome plane pixels; the CG4 scheme only allowed two. See Table 9-1 for details:

Table 9-1 *Enable/Overlay Planes for CG4 and CG8/CG9*

Overlay Plane	*Enable Plane*	*CG4 Scheme*	*CG8/CG9 Scheme*
0	0	8-bit color	24-bit color
0	1	color 0	color 1
1	0	8-bit color	color 2
1	1	color 1	color 3

Lookup Table

The CG9 has a way of adjusting pixel color values globally. The color components of the true color plane can be adjusted by modifying the plane's lookup table, a 256 by 3 bit table which control primary color component intensities. This table is different from a colormap, in that it adjusts the intensity response of each color component separately.

The color lookup table is used to adjust the intensity response of a pixel's three color components; red, green, and blue. This feature is often used for gamma correction: the intensity response of the color components can be adjusted to correct for inaccurate color reproduction in the system's display hardware.

Double Buffering

The CG9 provides double buffering of the true color plane by splitting a 24-bit pixel into two 12-bit pixels. The application reads or writes to each of the double buffers as if they are 24-bits deep. The CG9 hardware thresholds the color by storing only the 4 high-order bits of the red, green, and blue color components.

Because of this thresholding, an application will not necessarily read the same value back from the double buffer that it wrote to it. Double buffering is discussed in more detail in the *4.0.3 - GFX Rev. 1 Pixrect Reference Manual.*

Documentation

For more details on manipulating plane groups, the use of the true color lookup table, or double buffering, refer to the *4.0.3 - GFX Rev. 1 Pixrect Reference Manual.*

For details on the SunView Window system, and the use of overlay planes, see the *SunView1 Programmer's Guide.*

9.4. The SunDials™ Image Manipulation Device

SunDials is an image-manipulation input device for Sun-3 and Sun-4 workstations, using an RS-232 serial interface. The desk-top dialbox is compact, measuring 8.63″×5″×1.15″.

The dials can be programmed to change colors and manipulate images. SunDials is designed to perform CAD image manipulation such as scaling, translating, rotating, and zooming.

SunView directs SunDials input to the process owning the window (where the cursor is located). SunDials extends the interactivity of the display controller beyond the current capabilities of a mouse or digitizing tablet.

The SunDials device driver is integrated into the GENERIC kernel. No optional software or kernel reconfiguration is required, but kernel reconfiguration is recommended for optimal performance.

See the `dialtest`(6) manual page for details.

9.5. SunButtons™ Graphics Manipulation Device

SunButtons is a special input device for graphics applications. Software support for SunButtons includes SunView modifications to accept SunButtons as a graphical pick input device.

SunButtons hardware consists of a box containing thirty-two buttons, each of which can be used as a different function key. A serial cable assembly plugs into one of the workstation serial ports. A power supply provides current to the button box. SunButtons is supported on all Sun-4 workstations.

Each time a button is pressed, one byte of data is sent to indicate the button number. The serial communication protocol is RS-232C or RS-423.

SunButtons Documentation

The SunButtons hardware package includes a *SunButtons Hardware Installation and Programmer's Guide* (Part Number 800-3724-10) as well as a *Read Me First* document.

9.6. Sun-3/E Color Video Board

The 3E480 color video board for the Sun-3/E workstation is a P2 bus slave which provides a megabyte color frame buffer, an overlay plane, an enable plane, and a color map. It communicates with the CPU over the P2 bus, and outputs control signals to the video monitor. It appears to the CPU as a number of physical addresses; physically, it is connected over the P2 bus.

The color board uses a Brooktree Bt458 DAC chip to generate video control signals. This DAC provides a 256-by-24 color lookup table, and RS-343A compatible color output signals. It is described in the Brooktree Bt458/451 *Data Sheet.*

10

New Hardware: Peripherals

This section covers the new peripherals available with SunOS release 4.1.

10.1. The SunCD™ Driver

Introduction

The Desktop SunCD™ Pack is a Compact Disc/Read Only Memory (CD-ROM) drive. The SunCD reads and utilizes the large amounts of data that can economically be stored on compact discs.

The following is an overview of the SunCD's features; for details, see *Appendix B: Using the SunCD Driver* of this manual, and the *Desktop Storage Pack Installation Guide* (Part Number 800-4476-10) shipped with the SunCD hardware.

SunCD-supported Hardware

In SunOS release 4.1, SunCD is supported on the following platforms

- SPARCstation 1
- SPARCserver 1
- SPARCstation 330

CD-ROM in the Sun Environment

CD-ROM is by far the most economical way of distributing and publishing data currently available. In the Sun environment, there are two additional benefits:

High Performance
Sun workstations take full advantage of the SunCD's 64 KB cache and a 1.2 MB/second transfer rate.

Data Sharing
The SunCD can be accessed from any NFS-connected workstation, enabling users to economically share data as well.

SunCD Software

The SunCD software includes:

- An ISO-9660 and High Sierra Format filesystem,
- A device driver, and

- Utilities for
  a) Mounting the CD-ROM filesystem,
  b) Ejecting the CD-ROM caddy via software command, and
  c) Playing audio CD disc.

  See *Appendix B: Using the SunCD Driver* for details.

High Sierra Group File System Support

The SunCD file system conforms fully to the ISO-9660 standard format for CD-ROM filesystem, commonly referred to as the HSFS (*"High Sierra"*) filesystem. Application developers and data users need not learn or use any new commands or programmatic access to utilize CD-ROM files. After mounting the CD-ROM, files are fully accessible using standard SunOS commands and system calls. The High Sierra filesystem is implemented through the VFS (Virtual File System); see appendix B of this manual for details.

Disc Specifications

A CD-ROM disc has the same physical size and properties as the common audio "CD." The disc is single sided containing up to 644 MB of data, 74 minutes of audio, or any combination of the two. The Desktop SunCD Pack is designed to read CD-ROM discs, CD-Audio discs, and combined-mode discs.

Transparent Shared Network Access to CD-ROM Applications

In contrast to standalone personal computer use, SunCD permits CD-ROM to be a resource for network computing. Because it is integrated seamlessly with NFS under the VFS architecture, the SunCD is available to any user in the NFS network, including those on workstations not supported by local SunCD hardware connection. Any number of network users may even access the same disc simultaneously. However, this is not a recommended practice since it creates contention for data on very different portions of the CD-ROM disc, and may cause degradation in performance.

Use with DOS Windows™

See *Appendix B: Using the SunCD Driver* for details.

10.2. Front-Load Tape Drive

The Sun front-load 1/2-inch tape drive is a 1/2-inch reel-to-reel tape drive with autoloading capability. This horizontally oriented tape drive, when appropriately mounted, fits into a Sun 56-inch data center cabinet.

The front-load tape drive offers the following features:

- Compact, ergonomic design with front autoload
- High performance
- 125 ips tape speed
- Start/Stop performance using 512 KB buffering
- SCSI bus arbitration with disconnect/reconnect capability for multi-target or multi-initiator systems
- Higher capacity when using 1 mil thick, 3600 feet-per-reel tape
- Custom operating features, selected by the operator from the control panel
- Tape reel compatibility with 800 bpi, 1600 bpi, and 6250 bpi formats

Density may be configured as needed. The drive supports the following formats:

- 800 bpi (bits per inch) Non Return to Zero Inverted (NRZI)

- 1600 bpi Phase Encoded (PE)
- 6250 bpi Group-Coded Recording (GCR) format.

10.3. QIC-150 Tape Drive

The QIC-150 tape drive has a special 1/4″ cartridge that holds 150 MB of data versus the current 60 MB.

The 150 MB tape drive can write to the 3M-DC6150 tape cartridge (formerly the 3M-DC600XTD tape cartridge), available as Sun part number 370-1203-01. The new device is `st24` (`st0` or `st1` on SPARCstation 1).

Recommended `dump` parameters

The most efficient use of the QIC-150's new capabilities can be obtained by setting Tracks to 18 and size to 700, as below.

```
narc% dump ${LEVEL}unctsf  18 700  /dev/rst0
```

Using the instructions above will allow you to dump 139.056 MB out of 150 MB per tape.

Tapes written on the 60 MB tape drive, such as the 3M-DC300XL/P tape cartridge (Sun part number 370-0543) and DEI Series II Silver (Sun part number 370-0543) tape cartridge, can be *read* on the 150 MB tape drive.

NOTE: **Failure to use the proper tape for the specified tape drives, or use of any other tape cartridges, is not supported by Sun Microsystems.**

To find the type of tape drive installed on your system, load a tape into the drive and enter the following command, replacing ***NUM*** with the number of your tape device (for example, `/dev/rst8`):

```
mt -f /dev/rstNUM status
```

The following message displays when you have a 150 MB 1/4″ tape drive:

```
Archive QIC-150 tape drive:
   sense key(0x0)= no sense   residual= 0   retries= 0
   file no= 0   block no= 0
```

10.4. High-Performance SMD Disk Drive and Controller

688-MB SMD Disk Drive

This is a high-capacity, 8″ disk drive based on Winchester technology. Called the Storage Pedestal Upgrade Drive (SPUD), the drive has an unformatted capacity of 700 MB and formats to 688 MB. This is 2.5 times greater than the existing 8″ disk drive in the Sun Mass Storage Pedestal, although it occupies the same physical space.

The SPUD also improves on the access time, cost per megabyte of storage, and reliability of the current 8″ drive. Performance improvement results in part from SMD-4, the new SMD controller.

VME/SMD Disk Controller

This high-performance VME/SMD disk controller has a 128-kilobyte read-ahead cache and a pipelined bus DMA architecture. Each controller supports up to four of one type of the following disk drives:

- 8″ 688 MB disk drive
- 10″ 575 MB disk drive
- 9″ 892 MB disk drive

The disk controller also supports the following functions:

- Overlapped seeks
- Read and write optimizations

A

Special Notes for Unbundled Products

A.1. About this Appendix

This Appendix describes compatibility information needed for Sun products that are not sold with SunOS release 4.1.

A.2. Unbundled Products Requiring Special Instructions

SunLink

Installing SunLink Products

When installing some 6.0 and 6.1 SunLink products, you must run the `/usr/etc/sunlink.install` script *after* you run `extract_unbundled`, but *before* you run any product-specific installation scripts. If you do not run `sunlink.install` first, the product-specific installation scripts fail.

The new installation steps are as follows:

1. Run `/usr/etc/extract_unbundled` to extract the product off the tape.
2. Run `/usr/etc/sunlink.install`, selecting the product being installed.
3. Run any product-specific installation or configuration scripts according to the product documentation.

Table A-1 lists the affected SunLink products.

Table A-1 *Run* `sunlink.install` *for These Products*

6.1 BSC3270
6.0 BSCRJE
6.0 Channel Adapter
6.0 DDN
6.0_SCP
6.0 DNI*
6.0 HSI
6.0 INR*
6.0 MCP*
6.1 SNA3270
6.0 Peer-to-Peer

*There may be additional steps for these products. See the next sections for details.

Installing the Internetwork Router

If you are installing the SunLink Internetwork Router on a system that is running SunOS release 4.1, you must perform the following step:

After you run the `extract_unbundled` and `sunlink.install` scripts, but before you run the `install.inr` script, enter the following command as `root`:

```
blob# rm /usr/sunlink/inr/sys/arch/OBJ/in_pcb.o
```

where *arch* is a directory name that designates your machine architecture, for example, `sun4` for a Sun-4. You can then proceed to run `install.inr` as documented in the *SunLink Internetwork Router System Administration Guide.*

Installing the MCP on Sun-3/400 Series

If you are installing the SunLink MCP (Multiprotocol Communication Processor) software on a Sun-3/470 or Sun-3/480 workstation that is running SunOS release 4.1, you must perform the following steps:

After you run the `extract_unbundled` and `sunlink.install` scripts, but before you run the `install.mcp` script, enter the following commands as `root`:

```
blob# cd /usr/sunlink/mcp/sys
blob# cp GENERIC.mcp.add temp
blob# chmod +w GENERIC.mcp.add temp
blob# sed 's-16-32-' temp > GENERIC.mcp.add
```

You can then run `install.mcp` as documented in the *SunLink MCP Installation and Configuration Guide*.

Special Note for Diskless Clients

Diskless clients that attempt to use their CPU serial ports may hang during boot. For example, some SunLink products may be configured to use the diskless client's local CPU port.

In order to install SunLink products that contain kernel drivers on diskless clients running SunOS release 4.1, you must edit the configuration file and move the line that describes the Ethernet device (device `ie0` or `le0`) *before* the lines that describe the `zs` ports.

Change:

```
device          zs0 at obio ? csr 0x20000 flags 3 priority 3
device          zs1 at obio ? csr 0x00000 flags 0x103 priority 3
device          ie0 at obio ? csr 0xc0000 priority 3
```

To:

```
device          ie0 at obio ? csr 0xc0000 priority 3
device          zs0 at obio ? csr 0x20000 flags 3 priority 3
device          zs1 at obio ? csr 0x00000 flags 0x103 priority 3
```

Exporting SunLink Software for Multiple SunOS Versions

When you install SunLink software, the installation script installs the software only in the hierarchy for the SunOS version that the machine receiving the installation is running. That is, if you install on a machine running SunOS release 4.1, the SunLink software goes into the hierarchy for 4.1. If that machine is a server that has diskless clients that are running different SunOS versions from the server's and that require access to SunLink software, you must do the following:

- Create a `sunlink` mount point at the end of the version-specific hierarchy that the client mounts on `/usr`.
- Edit the client's `fstab` file so that the client mounts the hierarchy where the SunLink product was installed on the newly-made mount point.

For example, assume you have a Sun-4 server running SunOS version 4.1 that serves diskless Sun-4s running SunOS 4.0.3 and SunOS 4.1. Your 4.1 clients mount one hierarchy as `/usr`:

```
/export/exec/sun4.sunos.4.1
```

while your 4.0.3 clients mount a different hierarchy as `/usr`:

```
/export/exec/sun4.sunos.4.0.3
```

After you install SunLink software on the server, it is available as:

```
/usr/sunlink/<product_name>
/export/exec/sun4/sunlink/<product_name>
/export/exec/sun4.sunos.4.1/sunlink/<product_name>
```

To allow your 4.0.3 clients to access the SunLink software, use the following commands to create the mount point `sunlink`:

```
server# cd /export/exec/sun4.sunos.4.0.3
server# mkdir sunlink
```

Note that the hierarchy `/export/exec/sun4.sunos.4.0.3` is already mounted as `/usr` on the 4.0.3 clients. In the `fstab` for these clients, add an entry so that `/export/exec/sun4.sunos.4.1/sunlink` (the hierarchy that contains SunLink software) is mounted on the newly created mount point `/usr/sunlink` after the `/usr` entry, for example:

```
<server>:/export/exec/sun4.sunos.4.0.3       /usr         nfs ro 0 0
<server>:/export/exec/sun4.sunos.4.1/sunlink /usr/sunlink nfs ro 0 0
```

`setsid` Problems When Running SunLink DNI 6.0

If you install SunOS 4.1 on a machine running SunLink DNI 6.0, you must provide a "wrapper" around the DNI virtual terminal daemon (`dnilogind`) so that it conforms to the POSIX- standard requirements for acquiring a controlling terminal.

To allow a Sun node to remain accessible via SunLink DNI when a `set host` command is issued on a VAX/VMS† system, as `root`, enter the following commands after DNI installation is complete:

† VAX and VMS are trademarks of Digital Equipment Corporation.

```
blob# cd /usr/sunlink/dni
blob# mv dnilogind .dnilogind
blob# cat > dnilogind
#!/bin/sh
/usr/etc/setsid -b /usr/sunlink/dni/.dnilogind "$@"
^D
blob# chmod a+x dnilogind
```

It is not necessary to reboot your machine.

For background on the reasons for this requirement, see the SunOS 4.1 man page on `setsid (8V)`. (This man page is not present in previous SunOS versions.)

`setsid` Problems When Running SunLink X.25 6.0

If you install SunOS 4.1 on a machine running SunLink X.25 6.0, you must provide a "wrapper" around the X.29/X.3 server (`x29`) so that it conforms to the POSIX- standard requirements for acquiring a controlling terminal.

As `root`, enter the following commands after X.25 installation is complete:

```
slack# cd /usr/sunlink/x25
slack# mv x29 .x29
slack# cat > x29
slack#!/bin/sh
/usr/etc/setsid -b /usr/sunlink/x25/.x29 "$@"
^D
slack# chmod a+x x29
```

It is not necessary to reboot your machine.

For background on the reasons for this requirement, see the SunOS 4.1 man page on `setsid (8)`. (This man page is not present in previous SunOS versions.)

FORTRAN, C, Pascal, Modula-2: Missing Profiling or Debugging Libraries Generate Error Message

If you are trying to debug or profile with C, FORTRAN, Pascal, or Modula-2, you may get an error message indicating that a particular function is missing. This may be missing profiling or debug libraries.

For example, if the profiling libraries are not loaded, you will get a message as shown below.

```
demo% f77 -p test.f
test.f:
 MAIN bork:
ld: -lc_p: No such file or directory
demo%
```

Ask your system administrator to help you install the missing libraries.

FORTRAN 1.2

The installation of FORTRAN 1.2 may fail on SunOS release 4.1.

To fix this problem, modify the `/usr/tmp/1.2_fortran` file and restart the installation as indicated below.

1. Stay logged in as superuser.
2. Change directory to `/usr/tmp`

```
demo% cd /usr/tmp
```

3. In any editor, revise the `/usr/tmp/1.2_fortran` file as follows:

   Change: `SOS_COMPAT="4.0"`
   to: `SOS_COMPAT="4.1 4.0"`

   and resave the file.

4. Issue the following command:

```
demo% /usr/tmp/unbundled/1.2_fortran -rrmt_host -ddev
```

Where *rmt_host* is the name of the remote host if the tape is mounted remotely, and *dev* is the device specification (`st0`, `mt1`, etc.).

5. Restart the installation.

```
demo% install_unbundled -f
```

TranScript™ 2.1 Installation Failure

The TranScript installation script looks for a release level of 4.0.*X*. This can be fixed by editing the release level in `/etc/motd` from 4.1 to 4.0, then installing TranScript. After the installation, restore the correct release level in `/etc/motd`.

SunDraw 1.0

Installing SunDraw from the Floppy Media Set fails due to a missing `eject` command. Opening a second window and issuing the `eject` command will successfully complete the installation.

SunTrac (1032520)

The SunTrac tutorial program contains reference to a `Play`, that has its starting date on 01/01/90. Since the real date is now greater than that date, a feature of the product is activated which requires the user to modify a considerable number of dates in the Tutorial. This can hinder testing considerably and create documentation errors. The `Clear Date Error` menu item will clear the date errors each time they are encountered.

Sun58TE™ 1.0 Installation Failure

Sun58TE fails to installation on SunOS release 4.1 because of a compatibility check. The workaround is to edit the `/usr/tmp/unbundled/1.0_Sun58TE` file, changing line 63 from

```
SOS_COMPAT="4.0"
```

to

```
SOS_COMPAT="4.1"
```

and lines 75 and 1021 from

```
ARCH_OS_LIST=" sun3.40-2 sun4.40-3 sun386.40-4 opt_sharable-5"
```

to

```
ARCH_OS_LIST=" sun3.41-2 sun4.41-3 sun386.40-4 opt_sharable-5"
```

Running OpenWindows™ On SunOS Release 4.1

The instructions below are necessary to run OpenWindows on SunOS release 4.1; security has been enhanced and these steps are required to adjust for the new functionality. Specifically, when window security is enabled, the user must be the owner of the framebuffer and window device files in order to start a window system (such as SunView™ and X11/NeWS™†).

The SunView window system is bundled with the SunOS 4.1 release, and has been modified appropriately. OpenWindows 1.0 and OpenWindows 1.0.1 were released prior to SunOS 4.1, however, and one of the two procedures described below must be followed in order to run it on release 4.1.

The first set of instructions below describes the recommended procedure for starting OpenWindows 1.0 and 1.0.1. The second set of instructions describes the procedure for disabling window security. Use this method if you have a multi-headed system with different users on each head. Security enhancements are not supported in this configuration.

Modified OpenWindows Start-up Procedure

1) Rename the OpenWindows server:

```
sash% mv $OPENWINHOME/bin/xnews $OPENWINHOME/bin/xnews.orig
```

2) Place the following script in the `$OPENWINHOME/bin/xnews` file:

†The X window system is a product of the Massachusetts Institute of Technology

```
#! /bin/csh
# xnews -- startup script for xnews

# Change the ownership of the win devices to the current user.
#     Start with win0, change all 256 wins, change 240 wins in the
#     background (changing 240 in the background reduces startup cost).
/usr/bin/sunview1/sv_acquire 0 256 240

# Start up the xnews server
xnews.orig

# Change the ownership of the win devices back to root
#     Change ownership in background to minimize termination cost.
/usr/bin/sunview1/sv_release &
```

3) Change permissions of the script:

```
sash% chmod 755 $OPENWINHOME/bin/xnews
```

Disabling Window Security by Changing Ownership of the Frame Buffer and Window Device Files

The following steps describe how to change ownership of the framebuffer and window device files, disabling window security. Once these steps are followed, OpenWindows and other window systems compatible with SunOS release 4.1 can be used without further modification.

1. Disable logins (e.g., bring system to single user).
2. Comment out all lines in `/etc/svdtab`.
3. Change ownership of all `win` device files to root.

```
pane% chown root.wheel /dev/win*
```

4. Change permissions of all `win` device files.

```
pane% chmod 666 /dev/win*
```

5. Comment out all lines in `/etc/fbtab`.
6. Change ownership of all device files listed in `/etc/fbtab`.

```
pane% cd /dev
pane% chown root.wheel mouse kbd fb cg* bw* gp*
pane%
```

7. Change permissions of all device files listed in `/etc/fbtab`

```
pane% cd /dev
pane% chmod 666 mouse kbd fb cg* bw* gp*
pane%
```

OpenWindows™ 1.0 `image` Demo Hangs System (1033209)

When the `image` demonstration program for OpenWindows™ 1.0.1 is put into `Bounce` mode, the screen either hangs permanently or the mouse echo is delayed by more than 1 minute.

When swap space is full, OpenWindows either hangs the screen permanently, or aborts with a segmentation violation and core dump. There is no warning that this is about to happen, and no indication that the cause of the problem is no swap space. This problem occurs on both in SunOS release 4.1 and SunOS release 4.0.3.

OpenWindows also has a problem with the `image` demo. When the `image` demonstration program for OpenWindows™ 1.0.1 is put into `Bounce` mode, the screen either hangs permanently or the mouse echo is delayed by more than 1 minute.

A.3. Unbundled Products that Require `extract_patch`(8) for Installation

Extracting Patches from CDs

The following steps are necessary to extract the patches supplied with the CD-ROM version of SunOS release 4.1.

1. Running single user (or quiescent system) as root (superuser).

   NOTE: You must be running a kernel with the HSFS filesytem configured into it. (all GENERIC kernels have it)

2. Insert the CD-ROM into drive.
3. Mount the CD-ROM:

```
bladerunner% mount -rt hsfs /dev/sr0 /usr/etc/install/tar
```

4. Check the patch directory:

```
bladerunner% ls /usr/etc/install/tar/patches/sunos_4_1
```

5. For each patch of name *name*, you wish to extract, do the following:

```
bladerunner% cd /usr/tmp
bladerunner% mkdir name
bladerunner% cd name
bladerunner% tar xf /usr/etc/install/tar/patches/sunos_4_1/patch_hsfs_name
bladerunner% install_name
bladerunner% cd ..
bladerunner% rm -rf name
```

6. After extracting all the patches, `unmount` the CD-ROM

```
bladeruner% umount /usr/etc/install/tar
```

The patches available on the CD-ROM are

name	*hsfs_name*
TAAC	taac
ipc	ipc
C++_2.0	cplusplus_2_0

TAAC-1 Release 2.3

The TAAC-1 driver (`taac.o`) supplied with TAAC-1 release 2.3 will not work under SunOS release 4.1. The `taac.o` sent with the final version of SunOS release 4.1 *does* work with that release.

To use TAAC-1 on release 4.1, install the TAAC-1 software included with that release first, then install the rest of the release, then rebuild the driver. (No prior versions of the TAAC-1 software are supported under SunOS release 4.1.) If SunOS release 4.1 software has been installed first, you must take the following steps:

1. Before installing the TAAC-1 software, copy the version of `taac.o` that comes with SunOS release 4.1 to a temporary location:

```
hammer% cd /sys/ARCH/OBJ
hammer% cp -p taac.o taac.o.sav
```

2. Install the TAAC-1 software. The `taac-links` script, which is part of the installation process, will replace the existing version of `/sys/`*ARCH*`/OBJ/taac.o` with the version that comes with the TAAC-1 software.

3. Before rebuilding the system kernel, rename the saved version of `taac.o`:

```
hammer% cp -p taac.o.sav taac.o
```

4. Rebuild the system kernel as described in the *TAAC-1 Software Installation Guide* supplied with the TAAC-1 software.

SunIPC™ 1.2 Installation

SunIPC requires the installation of a software patch to work with SunOS release 4.1, because of a kernel interface change. The patch directory includes the following:

- A README file with instructions for installing.
- New kernel drivers for `sun3`, `sun3x`, and `sun4` kernel architectures.
- A patch installation script (`install_ipc`).
- A new tape extraction script (`1.2_IPC`).
- A new SunIPC configuration script (`ipc_configure`).

Patch Installation with `extract_patch(8)`

To install the patch directory, use the following steps:

1. Install SunOS release 4.1.
2. Extract the patch directory by typing `extract_patch` on a command line. `extract_patch` will prompt you for the answers to four questions. The first question is a prompt for the name of the patch requested. The name of this patch is "`ipc`". The `extract_unbundled`(8) command can then be used to install the script. `extract_unbundled` will prompt you for answers to whether the installation is local or remote, the name of the tape drive, and the pathname to the desired location for the patch directory. If no location is specified, the SunIPC patch directory will be put in `/usr/tmp/Patch_ipc` by default.
3. `extract_patch` will then ask if you want to run the `install_ipc` installation script. Answer "`yes`" and the patch files will be installed.
4. You can now install SunIPC software by loading the tape containing SunIPC software and typing `1.2_IPC` on a command line in the patch directory. `1.2_IPC` is the new tape extraction script.

You should now have a working SunIPC version 1.2. If you do not, contact your local Sun AnswerCenter.

Special Note for `sun3x` Users

NOTE: This section only applies to users of workstations with `sun3x` kernel architectures (Sun-3/80, 3/460, 3/470, 3/480).

The SunIPC 1.2 installation script does not take sub-architectures into account. The distributed configuration script `files/ipc_configure` will not work on workstations with `sun3x` kernel architectures. Instead, you must run the new `ipc_configure` script included in the patch directory.

The easiest way to do this is to install the SunIPC 1.2 software normally, but answer "`no`" to the following question:

```
The SunIPC 1.2 software is now installed into your system.
Would you like to configure the IPC software? no
```

NOTE: If you are mounting the SunIPC software instead of reading it from tape, then you are already at this point.

Then run the new `ipc_configure` script included in the patch directory.

You should now have a working SunIPC 1.2. If you do not, contact your local Sun AnswerCenter.

Sun C++ 2.0

SunOS 4.1 introduces new library functions and system calls. For Sun C++ 2.0 to run under SunOS 4.1 it is necessary to install a patch. This section describes how to install the patch.

Included in the patch are: a README file, the patch installation script, a directory with header files for SunOS 4.1 and updated versions of `dbx` for both Sun-3 and Sun-4 workstations.

Where is the Sun C++ Patch Installed?

Install the patch in the directory in which Sun C++ 2.0 was previously installed. The default is `/usr/CC` for standalone and homogeneous servers, and either `/export/exec/sun3/CC` or `/export/exec/sun4/CC` for heterogeneous servers. You also have the option of specifying a non-default directory as the installation directory. Just make sure it exists and that you have already installed Sun C++ 2.0 in it; otherwise, the installation script will not allow you to install the patch.

The patch script will remove the original `incl` header file directory and the `dbx` executable and install the new files in their place; therefore, the original `incl` directory and `dbx` executable *will be lost.*

The script is designed to install the patch for a single architecture (Sun-3 or Sun-4). If you are installing the patch on a heterogeneous server with clients of different architectures, you *must* run the script for each architecture.

DON'T FORGET!

Install SunOS 4.1 and Sun C++ 2.0 before you install the Sun C++ 2.0 patch.

If you do not understand some of the terms used here, see the installation instructions in the Sun C++ 2.0 RTF.

Preparing for Patch Installation

If you are installing on a standalone workstation and using the default installation directory, you may use default installation directions. If you are going to install the software onto a server, follow the non-default installation directions. If you need additional information, see the man page for `extract_patch`(8).

Default Patch Installation

1. To extract the patch directory become the root user of the workstation and type **/usr/etc/extract_patch -DEFAULT C++_2.0** on a command line. The script will extract the patch files from the tape, display the README file and ask whether you want to execute the patch installation program. Enter **yes**.
2. If the default destination `/usr/CC` exists and contains directories `incl` and `sunX` (where `sunX` is either `sun3` or `sun4`), this is what you will see:

```
Ready to install C++ 2.0 patch in /usr/CC

Do you want to continue: [y|n]? y


Doing the patch.....

Done!
```

Non-Default Patch Installation

Become the root user of the workstation and type `/usr/etc/extract_patch C++_2.0` on a command line. The script will extract the patch files from the tape, display the README file and ask you whether you want to execute the patch installation program. Enter `yes`.

1. The following message will be displayed on the screen:

```
Do you want to see a description
of this patch script [y|n]? y

Patch of Sun C++ FCS release 2.0 for SunOS 4.1


Patching should take approximately 5 minutes.
```

2. If you are installing the patch on a machine that is running on SunOS 4.0, you will see the following message:

```
WARNING: This patch script is for Sun OS 4.1
 You are running it on Sun OS 4.0
Do you want to continue anyway [y|n]?
```

Enter **y** if you are installing the patch into a server running on SunOS 4.0 that will support machines running on SunOS 4.1.

3. The script will then ask you a number of questions about what type of system you have (standalone or server), what type of server (homogeneous or heterogeneous), and what type of client will the product run on (Sun-3 or Sun-4).
4. After you answer the questions, the script will display the default Sun C++ 2.0 directory for your configuration:

```
Currently the default C++ 2.0 directory is /usr/CC
Do you want to change the default directory [y|n]?
```

Enter **n** if the default directory is correct. Enter **y** if you want to specify a different directory. You will then be asked to type in the name of your installation directory.

5. See item 2 under the "Default Patch Installation", the script will continue in the same manner.

Sun C++ 2.0 should now work under SunOS 4.1; if it does not, contact your local Sun AnswerCenter.

A.4. Unbundled Products that are not Supported on SunOS release 4.1

SunWrite™ 1.1

SunWrite 1.1 fails in SunOS release 4.1; release 4.1 will not allow it to open documents. This problem will be fixed in the next revision of SunWrite.

FDDI 1.0

Use of FDDI is not supported with release 4.1.

SPE 1.1

Use of SPE is not supported on release 4.1. A patch is available from your local Sun sales representative which will allow SPE to run on release 4.1.

Channel 7.0

Use of Channel 7.0 is not supported on release 4.1. A patch is available from you local Sun sales representative which will allow Channel 7.0 to run on release 4.1.

Use of NSE 1.2 on Release 4.1 is Not Supported

Installation of NSE on a system running 4.1 will cause some system files to be corrupted and may cause a system failure. It's use is not supported on release 4.1; the problem will be corrected in the next revision of NSE.

A.5. Type-4 Keyboard/ Internationalization Compatibility with Unbundled Products

Internationalization will be phased into Sun's unbundled software products over time, starting with the release of SunOS 4.1. Some unbundled software products will take immediate and full use of the native-language keyboards, while other products will add national language support in newer, enhanced releases of the products at later dates. The following lists show the status of major software products at the time of beta release of SunOS 4.1.

†Not supported in SunOS release 4.1.

Initial testing indicates that the following products provide full national language support when used with native language keyboards. Note that in most compilers, extended characters may not be used in variable names.

FORTRAN 1.2	Pascal 2.0	Modula-2 2.1
C 1.0	C++ 2.0	X Compilers 3.0
Deskset 1.0	SunGKS 3.0	SunPHIGS 1.1
IR 6.0	HSI 6.0	DDN 6.0
MCP 6.0	FDDI 1.0†	NSE 1.2†
SunNet Manager 1.0	SunNet License 1.0	Transcript 2.1.1

The following products will correctly recognize any of the national keyboards, but will not handle 8-bit characters. Behavior when an accented character is typed is unpredictable -- some will produce graphics characters and some will ignore the keystroke.

SunWrite†/Paint/Draw 1.1	SunTrac 1.3	BSC3270 6.1
BSCRJE 6.0	CG3270 6.0	SNA Peer 6.0
Local 3270 6.1	Channel 7.0†	SNA3270 6.1
X.25 6.0	OSI 6.0	MHS 6.0
SunCobol 1.0	SCLisp 3.0	NetISAM 1.0.DE

The following products will not recognize the existence of a national keyboard and will interpret all keystrokes as if the keyboard were a standard U.S. version. Many symbols and is some case alphabetic characters will not be read correctly.

OpenWindows 1.0*	SunIPC 1.2*	Guide*
DOS Windows 1.0	TE100 6.0	Sun58TE 1.0

*Upgrade to full national support expected by early fall.
†Not supported in SunOS release 4.1.

B

Using the SunCD Driver

B.1. About this Appendix

This appendix describes the functions and manipulation of the SunCD Desktop Storage Pack. For details on installation of SunCD, see the *Desktop Storage Pack Installation Guide* (Part Number 800-4476-10) shipped with the SunCD Driver.

B.2. CD-ROM in the Sun Environment

CD-ROM is by far the most economical way of distributing and publishing data currently available. In the Sun environment, there are two additional benefits:

High Performance

Sun workstations take full advantage of the SunCD's 64 KB cache and a 1.2 MB/second transfer rate.

Data Sharing

The SunCD can be accessed from any NFS-connected workstation, enabling users to economically share data as well.

The filesystem is implemented as a special file system type using the Virtual File System (VFS) architecture.

B.3. The SunCD Driver

The CD-ROM device driver conforms to the industry-wide SCSI 2 specification, supports block and character device modes, and supports a programmable audio interface. The hardware components, including their installation and use, are described in the *Desktop Storage Pack Installation Guide* (Part Number 800-4109) which accompanies the SunCD Desktop Pack.

B.4. SunCD Software

SunOS Release 4.1 has been enhanced to support the SunCD driver, including:

- An ISO-9660 and High Sierra Format filesystem,
- A device driver, and
- Utilities for
  a) Mounting the CD-ROM filesystem, and
  b) Ejecting the CD-ROM caddy via software command, and
  c) Playing audio CD disc.

B.5. Disc Specifications

A CD-ROM disc has the same physical size and properties as the common audio "CD." The disc is single sided containing up to 644 MB of data, 74 minutes of audio, or any combination of the two. The Desktop SunCD Pack is designed to read CD-ROM discs, CD-Audio discs, and combined-mode discs.

A disc contains from 1 to 99 tracks. Each track can contain exclusively audio or data information. A disc containing both audio and data would have at least two tracks, one for audio and one for data. A track has a minimum length of four seconds or 300 sectors. A data track can contain either mode 1 or mode 2 data. In mode 1 data track, each sector contains 2048 bytes of user data, with 288 bytes of ECC/EDC for error correction. In mode 2 data track, each sector contains all 2336 bytes of data without the ECC/EDC error correction. (In SunOS 4.1, the SunCD software supports the data reading of mode 1 data only) There also exists a lead-in and a lead-out area on the disc. This allows for overshoot during seek operations. These areas are outside the user defined area. Each track can be sub-divided by indexes. Every track has at least one index, and may have as many as 99. The user information starts at index one within a track. The index numbers are contiguous and ascending.

Each disc has a table of contents (TOC). The TOC contains information on the number of tracks on a disc, and the starting location of the user information within the track. The TOC also indicates which tracks are audio and which tracks are data. A CD-ROM device is very similar to a SCSI hard-disk, as far as a device driver is concerned. The differences are:

- CD-ROM is read-only.
- CD-ROM disc has no disk label.
- CD-ROM's default sector size is 2048 for mode-1 data.
- CD-ROM is a removable media.
- CD-ROM has special functionality such as the ability to read table of contents, audio tracks, and the Q sub-channel.

B.6. High Sierra Group File System Support

The SunCD filesystem conforms fully to the ISO-9660 standard format for CD-ROM filesystem, commonly referred to as the HSFS, or *"High Sierra"* filesystem. Application developers and data users need not learn or use any new commands or programmatic access to utilize CD-ROM files. After mounting the CD-ROM, files are fully accessible using standard SunOS commands and system calls. The SunCD implementation is a level 2 data interchange (meaning files consist of only one section) and a level one implementation (meaning Supplemental Volume Descriptors are not available) as defined in standard ISO 9660-1988. Support of demand paged execution also allows the kernel to execute programs directly from the CD-ROM disc.

CD-ROM access through the ISO 9660 filesystem (HSFS) is similar to magnetic disk drive access through Sun's UNIX filesystem (UFS) or network connection access through NFS. Overarching HSFS, UFS, and NFS is Sun's Virtual File System (VFS) interface which assures transparent access to all SunOS and standard system calls.

B.7. Block and Character Data Access

The CD-ROM device can be accessed through the mounted CD-ROM filesystem (see B.7) or an `ioctl()` system call on the CD-ROM raw (character) device. The SunCD drive is default to a logical block length of 512 bytes. Therefore, data are accessed in blocks of 512 bytes, either through the filesystem or `ioctl()` system call, despite the fact that the physical sector size of the disc is 2048 bytes.

The `ioctl()` system call allows program control of the CD-ROM drive. Besides regular data access, it provides CD-ROM specific operations such as audio play or read sub-channel. The "Programming Interface" section below describes the `ioctl()` interface in detail.

Through this call, many audio and CD-ROM specific operations can be performed. The *Programming Interface* section below describes the `ioctl()` interface in detail. The SunCD driver reads 512 bytes per block, the same as a hard disk.

B.8. Transparent Shared Network Access to CD-ROM Applications

In contrast to standalone personal computers, SunCD permits CD-ROM to be a resource for network computing. Because it is integrated seamlessly with NFS under the VFS architecture, the CD-ROM is available to any user in the NFS network, including those on workstations not supported by local SunCD hardware connection. Any number of network users may even access the same disc simultaneously. This is not a recommended practice since it creates contention for data on very different portions of the CD-ROM disc, and may cause degradation in performance.

B.9. Use of DOS Windows

Many basic PC-based application CD-ROM discs will run on SunCD without porting to UNIX under Sun DOS Windows™, a separately available unbundled software emulation product.

A mounted CD-ROM disk looks like any other filesystem to DOS Windows. You can just change directory to the mounted CD-ROM and access it.

Users may encounter three limitations when using a CD-ROM from DOS Windows 1.0:

1. Some applications only support VGA or EGA graphics adapters. DOS Windows 1.0 only emulates Hercules and CGA display adapters; if an application *requires* a VGA or EGA card, then it will not run in DOS Windows 1.0 (independent of CD-ROM issues).
2. DOS Windows use the normal DOS Windows file redirection method to access the CD-ROM drive. Applications requiring the use of any "Microsoft CD-ROM extensions" will not be able to access them. This is most commonly found in applications designed to play music from the CD.
3. Finally, applications that explicitly look for the presense of a MS-DOS CD-ROM driver will not find one. In the absence of such a driver, these applications will not even attempt to run, even though the CD-ROM drive is accessible from within DOS Windows.

B.10. Using the Desktop SunCD Pack

The device name for CD-ROM is `/dev/sr0` for block device and `/dev/rsr0` for character device. See the `sr`(4) and `cdromio`(4) man pages in the *SunOS Reference Manual* (Part Number 800-3827-10) for details about the SunCD device driver and `ioctl` operations with the character device. Once the Desktop SunCD Pack is connected, any CD-ROM with High Sierra or ISS0 9660 format data disc can be mounted as a filesystem.

B.11. Mounting and Unmounting File Systems

For directories mounted to a High Sierra or ISO 9660 format CD-ROM, the protection bits are same as the block device `/dev/sr0` file. This is also true for any files and subdirectories of the mounted directory. Therefore, the protection bit for the block device `/dev/sr0` is nominally set to 555 to preclude write permissions, while allowing read and execute for files and directories on the CD-ROM.

The following illustrates mounting the Sun SunCD Demo Disc 1.0 into a filesystem.

```
coffee% su
coffee#
coffee# mkdir /cdrom
coffee# mount -rt hsfs /dev/sr0 /cdrom
coffee# df /cdrom
Filesystem      kbytes  used    avail   capacity    Mounted on
/dev/sr0        251224  251226      0     100%       /cdrom
coffee# ls -F /cdrom
_readme*            demos_sun_software/ images_sun_systems/
audio_sparc/        documentation/      manual_pages/
catalyst_software/  fun_software/
demos_graphic/      images_misc/
```

(*The* **_readme** file contains a short description of the each directory's contents.)

NOTE: Only NFS and UFS filesystems will be automatically mounted upon rebooting. HSFS, like RFS, will not automatically be mounted; the user must `mount /cdrom` every time the system is rebooted.

Once the disc is mounted, the manual **eject** button on the drive is disabled. To change discs, unmount the filesystem and press **eject** on the drive. `/bin/eject cd` will also eject the CD from the player. (See the `eject`(1) manual page in the *SunOS Reference Manual* (Part Number 800-3827-10) for details on how to use the `eject` command).

```
coffee# umount /cdrom
coffee# eject cd
```

Playing Audio with `cdplayer`(6)

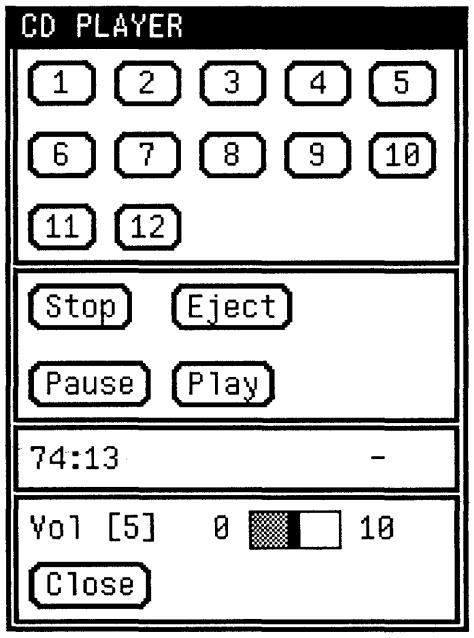
The *Desktop Storage Pack Installation Guide* illustrates how to hook up the audio output. Audio tracks can be played using the `ioctl()` commands or through a Sun supplied audio CD demonstration program: `/usr/demo/CDROM/cdplayer`. This program, built on these same audio `ioctl()` commands can play any audio track (on a CD-DA or mixed mode CD-ROM). To start the program, type the following:

NOTE: To run multiple CD applications simultaneously, use the `-n` option to `cdplayer` as shown below.

```
coffee# cd /usr/demo/CDROM
coffee# make cdplayer
coffee# cdplayer -n &
     (a cdplayer icon should appear)
```

Click the icon and a window will appear as shown:

Figure B-1 *A sample CD window.*



The top panel shows that there are 12 tracks on the CD. The third panel shows the entire CD is 74 minutes and 13 seconds long. Pressing **Play** button at this point will start audio play at track 1. If you wish to start playing from track 3, simply click the 3 button, then **Play**. The third panel will now display the music address and the track being played. Other available features are **Stop**, **Pause**, **Eject**, and VOL (volume control).

See `cdplayer`(6) in the *SunOS Reference Manual* (Part number 800-3827-10) for more information.

B.12. Utilizing NFS for CD-ROM

Once a CD-ROM disc in SunCD has been mounted by the local workstation, the local workstation must "export" the CD-ROM filesystem by making a one-line entry in the `/etc/exports` file as follows (as superuser):

NOTE: If the `/etc/exports` files does not already exist, you will need to reboot the system.

```
coffee#
coffee# mount -rt hsfs /dev/sr0 /cdrom
coffee# cat /etc/exports
/cdrom -ro
coffee# exportfs /cdrom
coffee# exportfs
/cdrom -ro
```

B.13. Programming Interface

You can access the SunCD through the filesystem (by issuing a system call to the mounted filesystem) or through the ioctl system call on the raw device of the CD-ROM. Normally, if you just need to access a file or execute a program that is stored on a High Sierra or ISO 9660 disc, you can just do a filesystem call. However, if you want to access raw data, CD-ROM specific information (Read the Table of Contents of the disc, etc.) or issue audio commands (play track 3, pause, etc.), you will need to use the `ioctl` call. Refer to the `sr`(4) and `cdromio`(4) manual pages in the *SunOS Reference Manual* (Part Number 800-3827-10) for more information. The CD-ROM device driver supports a set of `ioctl()` commands for audio operations and CD-ROM specific operations. It also supports part of the `dkio` operations. The following is a list of ioctl commands supported by the CD-ROM device driver:

- `DKIOCINFO, DKIOCGGEOM, DKIOCGPART, DKIOCGAPART, DKIOCGDIAG`: `dkio` operations.
- `CDROMPAUSE`: pause the audio play operation.
- `CDROMRESUME`: resume the paused audio play operation.
- `CDROMPLAYMSF`: audio play operation, starting and ending at the specified CD-ROM address in MSF format.
- `CDROMPLAYTRKIND`: audio play operation, starting and ending at the specified CD-ROM audio address at `track/index` format.
- `CDROMREADTOCHDR`: read Table Of Contents (TOC) header information. This includes the starting track number and the ending track number of the disc.
- `CDROMREADTOCENTRY`: read an entry in the TOC. The track number is supplied and this command returns the starting address and the type of data of the track.

- CDROMSTOP: spin down the disc.
- CDROMSTART: spin up the disc and seek to the last address requested.
- CDROMEJECT: spin down the disc and eject the caddy with the disc.
- CDROMVOLCTRL: control the audio output level.
- CDROMSUBCHNL: read the Q sub-channel data of the current block. The Q sub-channel data includes track number, index number, absolute CD-ROM address, track relative CD-ROM address, control data and audio status.

B.14. The Generic User SCSI Command

The CDROMUSCSICMD command allows the device driver to support all SCSI commands for any CD-ROM drives: SCSI 2 supported commands as well as vendor specific SCSI commands. In certain cases, this command would require the knowledge of the individual SCSI command block and the SCSI command set of the particular device. The interface for CDROMUSCSICMD is to supply a SCSI command block (of any group), user buffer address, user-SCSI command flags and address for the returning status block. The device driver will fire up the requested SCSI command to the device. The user has to provide all the parameters within the SCSI command block.

CDROMUSCSICMD is only available for use in SunOS release 4.0.3 - 4c, the SunOS for the SPARCstation 1.

B.15. Error Messages

The following are error messages which may be returned when attempting to mount or access a disc. See the *Desktop Storage Pack Installation Guide* for help with hardware problems.

1. This message appears if there is no disc in the driver, the disc is not connected, or the power is off. Check the drive and SCSI address:

```
decaf# mount -rt hsfs /dev/sr0 /cdrom
mount_hsfs: /dev/sr0 on /cdrom: No such device or address
mount: giving up on:
    /cdrom
#
```

2. This message may appear if the disc in the drive is either defective, or an audio compact disc.

```
decaf# mount -rt hsfs /dev/sr0 /cdrom
mount_hsfs: /dev/sr0 on /cdrom: I/O error
mount: giving up on:
    /cdrom
#
```

3. This message appears if either the CD-ROM device driver or the HSFS file system was not configured into the kernel during the manual software installation. Check with your system administrator:

```
decaf# mount -rt hsfs /dev/sr0 /cdrom
mount_hsfs: /dev/sr0 on /cdrom: No such device
mount: giving up on:
    /cdrom
```

4. This message appears if you attempt to mount a CD-ROM that is not formatted in the ISO 9660 or HSFS format.

```
decaf# mount -rt hsfs /dev/sr0 /cdrom
hsfs: Unknown CD-ROM structure format
mount_hsfs: /dev/sr0 on /cdrom: Invalid argument
mount: giving up on /cdrom
#
```

5. This message may appear in your console window when doing an `ls` on a CD-ROM file system:

```
hsfs: file type (0x4) not supported
hsfs: file type (0x8) not supported
```

File type (`0x4`) is an associated file (similar to a resource fork used by Macintosh™ files). file type (`0x8`) is a record format file. These files will not be accessible under the CD-ROM file system, and they will not be listed using the `ls` command.

6. This message appears in your console window when you start the cdplayer program if the compact disc if the compact disc is not inserted or if the drive is not connected.

```
cdplayer: failed to open device /dev/rsr0: No such device or address.
```

B.16. Removing the Disc

Once the compact disc is mounted, the manual eject button on the front panel of the storage pack is disabled. To remove a compact disc use the following steps.

Unmount the file system

Unmount the files system as follows:

```
decaf# /etc/umount /cdrom
decaf#
```

You may see this error message when attempting to execute the `umount` command.

```
decaf# /etc/umount /cdrom
/cdrom : Device busy
```

If it does appear, make sure no other user or program is accessing the file system. If so, change directories out of the `/cdrom` directory and halt any program that is accessing `/cdrom`.

For more information about the `umount` command, see the Sun *System and Network Manager's Guide*, or the `umount`(8) man page, either online or in the *SunOS Reference Manual.*

Press the Eject Button

Once you are certain no programs or users are using the `/cdrom` directory, press the eject button on the front panel of the SunCD Desktop Storage Pack, or enter the `eject` command. See the `eject`(1) manual page for more information about the `eject` command. If the `eject` is successful, the following message is returned:

```
sr0: Caddy ejected
```

If you attempt to eject the CD-ROM from the drive while another user is accessing it, the `eject` will fail, returning the following message:

```
decaf# /bin/eject cd
eject: Eject of "cd" failed: Device busy
```

If there is no compact disc in the drive or if the system is not connected to a SunCD Desktop Storage Pack, the `eject` will fail, and the following message will appear:

```
decaf# /bin/eject cd
eject: Open fail on cd -> /dev/rsr0: No such device or address
```

C

New adb Macros for Debugging Crash Dumps

The following is a list of the macros in /usr/lib/adb to assist in debugging kernel crash dumps. Unless otherwise noted, these macros will work on all architectures.

Name	Usage or function
astoproc	*addr*$<astoproc

astoproc — Given the address of a struct as, displays the proc structure that owns it.

`calltrace`

```
calltrace    addr$<calltrace (typical usage is <sp$<calltrace)
calltrace.nxt    used by calltrace
```

This macro is useful mainly on SPARC-based workstations. It does a stack back-trace on SPARC workstations, which looks something like this:

	l0	l1	l2	l3
	l4	l5	l6	l7
	i0	i1	i2	i3
	i4	i5	i6	i7
_u+0x2928:	f8110364	f8113cf8	e00002b2	80000000
	f80e4f40	fffff000	f80395d4	ffffe928
	f80d0349	72772c	0	1e84800
	fffff	0	ffffe990	f809b1f0
_u+0x2928:	0xf8110364	0xf8113cf8	0xe00002b2	0x80000000
	_panic_regs	_u+0x3000	_panic+0x7c	_u+0x2928
_sizestr+0x89	0x72772c	0	0x1e84800	
	0xfffff	0	_u+0x2990	_trap+0x190
_trap+0x190:	call_panic			
_u+0x2990:	4000c5	f8060e90	10000	40
	7	81	f8110364	0
	7	ffffea5c	400	0
	ffffeaa4	0	ffffea00	f80052f4
_u+0x2990:	0x4000c5	_blkatoff+0xc0	0x10000	PGSHIFT_DEBUG+0x34
	NW	WINMASK+2	0xf8110364	0
	NW	_u+0x2a5c	WINMASK+0x381	0
	_u+0x2aa4	0	_u+0x2a00	st_have_window+0x5c
st_have_window+0x5c:	call	_trap		

The first set of 4 lines is a header to remind you which registers are which. (The outs are the next frame's ins.)

Each stack frame has three groups to it. The first group is the register contents in hex. The second group is the register contents interpreted symbolically (`/p`). The third group is an effective `<i7/i` to show you where we were called from. In the above example, the first frame is for `panic`, which was called by `trap` (at `_trap+0x190`). The second frame is for `trap`, which was called by `locore` at `st_have_window+0x5c`, etc. Note that for frames due to traps you cannot believe `i7` but must rely on `l1` and `l2` to tell you where you were.

The traceback ends when

(a) `i6` of a frame contains `0` (end of stack marker), or

(b) `adb` cannot resolve either `i6` (to get the next frame) or `i7` (to display where we came from).

For case (b), you can manually restart the traceback by using the last displayed `i6` as the starting value. Note that the traceback can and will continue back into user-space, as SunOS release 4.1 `libkvm` understands how to translate user addresses. See also `stacktrace` in this section.

`direct`

`direct`	*addr,count*`$<direct`
`direct.nxt`	used by `direct`

Displays `count` consecutive `struct directs` beginning at location *addr*. Compare with `$<dir`, which displays `struct directs`. While `$<dir` deals with filesystem-independent directory information, `$<direct` is used for displaying ufs directories, as resident in kernel memory buffers, or when `adb`'ing a disk.

`dumphdr`

`dumphdr`	`dumphdr$<dumphdr`

Used for displaying the contents of a `struct dumphdr` (new-format crash dump header).

`forward`

`forward`	*addr*`$<forward`
`forward.nxt`	used by `forward`

The *addr* for `$<forward` is a "known" good stack pointer. It attempts to trace forward, looking for a frame that contains this value in `i6`, and then recurses forward. Useful when a stack is broken, and `$c` or `$<calltrace` ends prematurely. It stops when it reaches the frame pointed to by `<sp`, or when `adb` complains. This macro will also work with user core dumps (`adb a.out core`).

`fpu`

`fpu`	*addr*`$<fpu`

Displays a `struct fpu` located at *addr*. Most useful for SPARC-based workstations.

kforward

```
kforward        addr$<kforward
kforward.nxt    used by forward
```

Like `forward`, but starts from the known top-of-kernel stack address. The *addr* for `$<kforward` is the address of the `proc` structure for the process stack to be traced. For example, `*masterprocp$<kforward`. Only searches in the u-area (kernel stack). Use `forward` to continue the search into the interrupt stack if that is where `<sp` is. It stops when it reaches the frame pointed to by `<sp`, or when `adb` complains.

memseg

```
memseg          addr,count$<memseg
```

Displays `count struct memsegs`, beginning with the one located at *addr* and following the `next` pointer.

msgbuf

```
msgbuf          $<msgbuf
```

Displays the `msgbuf` header, followed by the "unread" portion of the msgbuf. For kernel panics, this is the most interesting part anyway. For the complete message buffer, use `msgbuf+10/s`.

pme

```
pme             addr,count$<pme     [not new, but improved]
pme.nxt         used by pme
```

Now takes an optional count of how many consecutive `struct pments` to display. Not supported on Sun-3/80.

pmenext

```
pmenext         pme_next$<pmenext
```

Displays the `struct pment` pointed to by the `pme_next` field of a `struct pment`. Not supported on Sun-3/80.

pmetov

```
pmetov          addr,count$<pmetov
```

Given a `struct pment` address, displays the virtual address associated with that `struct pment`. (It does this for `count` consecutive `struct pments`). Not supported on Sun-3/80.

`pmgrp`

`pmgrp` *addr,count*`$<pmgrp`

Displays `count` consecutive `struct pmgrps`. Not supported on Sun-3/80.

`regs`

`regs` *addr*`$<regs`

Displays a `struct regs`; for example, the second argument to `trap()` is a `struct regs`. Will work only on SPARC-based workstations.

`smap`

`smap` *addr,count*`$<smap`

Now takes an optional count of how many consecutive `struct smaps` to display.

`smap.find`

`smap.find` *vaddr*`$<smap.find`

Find the `struct smap` associated with the kernel virtual address *vaddr*, which should be located within the `segkmap` segment.

`snode`

`snode` *addr,count*`$<snode`

Displays `count` consecutive `struct snodes` beginning at *addr*.

`stack`

`stack` *addr,count*`$<stack`

Displays `count` stack frames, in `calltrace` format, beginning at *addr* and working backwards. May stop prematurely if an `i6` or `i7` cannot be resolved by `adb`. Most useful for SPARC-based workstations.

`stacktrace`

`stacktrace` *addr*`$<stacktrace`
`stacktrace.nxt` used by `stacktrace`

Will only work on SPARC-based workstations. Like calltrace, but eliminates the `i7` "where were we called from" information. Useful when `$<calltrace` keeps dying from bad `i7`s, and you want to get to the top of the stack quickly. This macro will also work with user core dumps (`adb a.out core`).

sysmap

`sysmap` *vaddr,count*`$<sysmap`

Display `count` consecutive `struct ptes` from the Sysmap, beginning with the one that maps the kernel virtual address *vaddr*.

u_fpu

`u_fpu` *addr*`$<u_fpu`

Will work only on SPARC-based workstations. Displays the struct `fpu` contained within the specified user structure. For example, `*uunix$<u_fpu`.

ucalltrace

`ucalltrace` *addr*`$<ucalltrace` (typical usage is `<sp$<ucalltrac`
`ucalltrace.nxt` used by `ucalltrace`

A version of `calltrace` that works on user core dumps (`adb a.out core`). Mainly useful on SPARC-based workstations.

ustack

`ustack` *addr,count*`$<ustack`

A version of stack that works on user core dumps (`adb a.out core`).

vattr

`vattr` *addr*`$<vattr`

Displays the `struct vattr` located at *addr*.

wbuf

`wbuf` *addr*`$<wbuf`
`wbuf.lbuf` used by `wbuf`

Will work only on SPARC-based workstations. Displays any saved user register windows in the u-area specified by *addr*. For example, `*uunix$<wbuf`. Can also be used when debugging user core dumps, if you use the following hack to map the u-area. Determine the size of the core dump with `ls -l`, then initiate the following:

```
adb - core
/m 0 -1 0
0t{size-of-core}-4000$<wbuf
```

D

SPARCstation 1-specific Information

This appendix explains features that are specific to the SPARCstation 1 desktop workstation. This appendix is divided into the following headings:

- SPARCstation 1 Audio Programming
- SPARCstation 1 Graphics Support
- Using a Second Disk
- Using an External Storage Module
- Parity Recovery
- Rebuilding the SPARCstation 1 kernel
- New PROM User Interface
- SCSI Unit Numbering and the SPARCstation 1 PROM
- EEPROM Differences

D.1. SPARCstation 1 Audio Programming

As predicted in Section 12 of the *SunOS release 4.0.3-Sun4c Release Notes*, the device driver for the SPARCstation 1 audio device has changed substantially for release 4.1. The `audio`(4) man page in the *SunOS Reference Manual.* (Part Number 800-3827-10) fully describes the new interface. Most audio programs compiled under release 4.0.3-Sun4c will operate normally in 4.1; however, *they will not compile in the 4.1 environment.* Binary compatibility with 4.0.3-Sun4c programs will not be maintained in future releases of the operating system. The audio programming interface provided with SunOS release 4.1 *will* be supported in future releases of the operating system.

The Demo category of software for SPARCstation 1 includes two programs to use the audio device. These programs, `soundtool` and `gaintool`, are described in the `soundtool`(6) and `gaintool`(6) manual pages in *SunOS Reference Manual.* They run under the SunView window system. `sound` allows a user to record, playback and do simple editing of audio files. `gain` is a prototype audio control panel that allows you to display and alter the state of the audio device. The program source, its `Makefile` and a sound sample are in `/usr/demo/SOUND`.

Also in `/usr/demo/SOUND` are copies of the following manual pages:

- `gaintool`(6),
- `play`(6),
- `record`(6),
- `raw2audio`(6), and
- `soundtool`(6).

The following files, in `/usr/demo/SOUND/man3`, are supplied as a preliminary interface to useful capabilities. They are unsupported and their syntax or semantics may be redefined in a future release of the SunOS:

- `audio_misc`(3)
- `audio_ulaw2linear`(3),
- `audio_convert`(3),
- `audio_device`(3)
- `audio_filehdr`(3),
- `audio_hdr`(3), and
- `audio_intro`(3)

Compiling and Running the Audio Demonstration Programs

1. Ensure that you are running the SunView window system. If not, you just type `sunview` to start it. See the `sunview`(1) manual page in *SunOS Reference Manual* (Part Number 800-3827-10) for more information about running SunView.
2. Run the `sound` program from a shelltool or cmdtool, according to the usual UNIX pathname rules (as described in the `sh`(1) or `csh(1)` manual pages).

To avoid tying up a shell while running it, you may run it in the background using "`&`". (Remember, processes running in the background will continue until terminated):

```
@aboy% /usr/demo/SOUND
@aboy% make
@aboy% gaintool &
@aboy% soundtool &
```

The `gaintool` demonstration program provides a control panel that may be used to control the state of the audio device. In particular, the following audio parameters may be modified:

`Record Volume`	Controls the recording level.
`Play Volume`	Controls the output level.
`monitor`	Controls the feedback from the microphone input to the output. This allows, for instance, the monitoring of input signals, such as a directly connected tape recorder.
`Output`	Toggle; built-in speaker/headphone jack.
`Pause/Resume`	Toggle; this button only works when audio playback is in progress.

The audio control panel illustrates the ability to control various aspects of the audio configuration outside of particular audio applications. For instance, since play volume may be controlled from the panel, it is not necessary for all audio applications to provide an output volume control themselves. The audio driver sends a `SIGPOLL` signal to processes that request notification of audio state changes. In this way, the sound demo may be notified when the gain demo alters the volume, and vice versa.

The `soundtool` Demonstration Program

1. Go to the directory where the source is, and run `make`:

```
@aboy% cd /usr/demo/SOUND
@aboy% make
```

2. Ensure that you are running the SunView window system. If not, type `sunview` to start it. See the `sunview`(1) manual page in the *SunOS Reference Manual* (Part Number 800-3827-10) for more information about running SunView.
3. Run the `soundtool` program from a shelltool or cmdtool, according to the usual UNIX pathname rules (as described in the `sh`(1) or `csh(1)` manual pages).

To avoid tying up a shell while running it, you may run it in the background using "`&`":

```
@aboy% /usr/demo/SOUND/soundtool &

     or

@aboy% cd /usr/demo/SOUND
@aboy% soundtool &
```

Shortly after you type the command, a new window should appear on the screen.

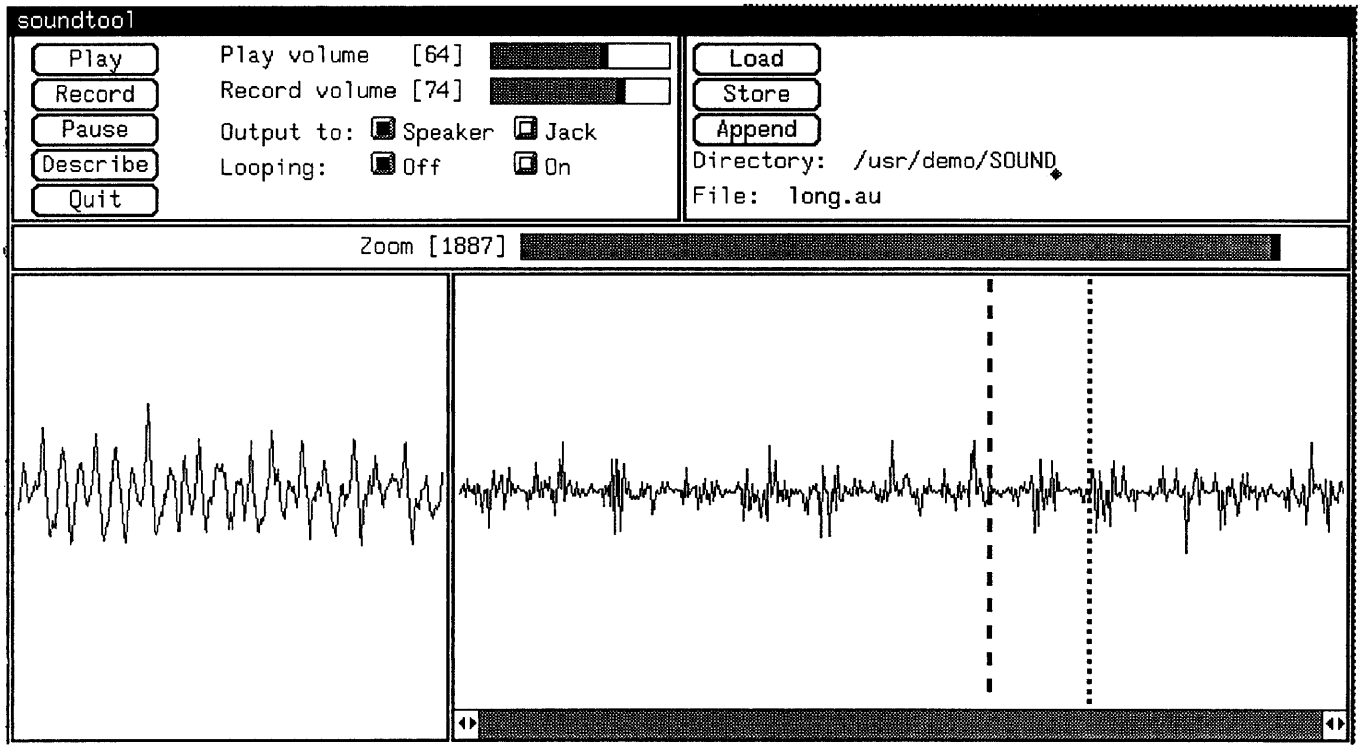


Figure D-1 *The* `soundtool` *Manipulation Window*

The `soundtool` Demonstration Program Sub-Windows

There are several subwindows, known as panels, in the upper left of the main `soundtool` control panel. This is the main control panel for this tool; it manipulates the `soundtool` program with the following controls:

Main Panel Controls

`Play/Stop`
The **Play** Button. Select to play sound. When selected it turns into the **Stop** button.

`Record/Stop`
The **Record** button. Select to record sound; when selected, it turns into the **Stop** button.

`Pause/Resume`
The **Pause** button. Select to pause playing or recording. When selected, it turns into the **Resume** button. When **Resume** is selected, `Pause/Resume` is reversed and turns into **Pause**.

`Describe`
Describe the sound data in the program's buffer. Pressing the left mouse button will open this window. This window can be closed either by moving the cursor into it and pressing the **Open** key (on the keyboard), or by moving the cursor into the border and popping up the SunView menu with the right button and selecting `Done`.

`Quit`
The [Quit] button.

`Play volume` slider
Used to vary the output volume. Move the cursor to the desired position and click with the left button; you may also hold down the left button and move the slider while the button is held.

`Record volume` slider
Used to vary the record level. This uses the same mechanism as the Play volume slider.

`Output to` buttons
Click (with left button) in box to select where the sound will go, to the built-in speaker or to the external jack.

`Looping Mode`
Selects play once or continuous looping of sound playback. This uses the same mechanism as the Output to buttons.

File Control Panel Controls

In the upper right is the `sound` file control panel. This is how you tell the `sound` program which files you want to use.

`Load`
The [Load] button. Select with left mouse button to load file specified by the File name field into the `sound` program.

`Store`
The [Store] button. Select with left mouse button to store data in the `sound` program buffer into a file specified by the File name field.

`Append`
The [Append] button. Same as [Store], except that it appends the sound data to the file.

`Directory` name field
Specifies the current working directory for the `sound` program. It specifies the directory used when selecting from a list of sound files (see below). The user can also change to a different directory.

`File` name field
This field can be used in two ways. The first is to enter the name of the file. To do so, click (with the left button) within the field to select it and then enter or delete text with the keyboard. The second way is to move the cursor to the field, hold down the right button, and move the cursor to the desired entry (notice that it highlights) and release the button. The selected name appears in the File name field. Note that the `sound` program only looks for files ending in `.au` (the convention for audio files). It will use files of other names, but you have to manually enter the name.

Across the middle is a small panel with a slider `Zoom`, controls the compression factor used to display the data in the waveform display panel. This lets you vary how much of the sound data is displayed. If you select a low compression, you see a small part of the data, though at a high resolution. A high compression lets you see more of the data, though lowering the resolution.

In the lower left is the `scope` panel. This displays the uncompressed waveform in near real-time.

In the lower right is the waveform display panel. This is the area used for display and editing of a sound sample. There are two cursors in here. The dashed cursor is the start cursor, it is moved with the left mouse button. The dotted cursor is the end cursor, it is moved with the right mouse button. The cursors are effective for playback and storing sound data to a file. Note that the cursors move to the spot the mouse cursor is when the mouse button is released.

Hooking Up External Speakers

You may want to have better quality (or louder) sound than the internal speaker can provide; if so, you may connect an external (powered) speaker. Or you might want to use a headphone for private audio. There is a 1/8" phone jack on the rear of the SPARCstation 1 that is designed to drive headphones which have an impedance of 30 ohms (or higher) or powered loudspeakers such as those made by Sony or Yamaha. Although the jack has two conductors in addition to the shield, the same signal appears on both of them. In other words, the output is not stereo.

You may also wire up the output signal using the 8 pin MINI-DIN that is also used for audio input (see below). This is the same signal that goes to the output jack.

Playback of `Sound` Files

This assumes the `sound` demo program is running.

1. Select the file to be played. There is a `sample.au` file in `/usr/demo/SOUND/sounds`.
2. Select **Load**. You will see the waveform appear in the waveform display panel.
3. Select an Output device (either Speaker or Jack)
4. If desired, select Looping On.
5. Select **Play**. You should now hear the sound and see the waveform displayed in the `scope` panel. If you like, you may adjust the play volume using the `Play volume` slider.

Recording Sound

First you have to hook up a microphone. There is a ready made cable you can get from Sun that has a 1/8" phone plug for input and a 1/8" phonograph jack for output - the part number is 530-1594-01. Otherwise you can make a cable yourself, using an 8 pin MINI-DIN connector. The input signals are:

Mini-DIN 8-pin end	Phono Jack
Pin 3	Microphone Tip (center connection)
Pin 6	Microphone Ring (also connect to shield)
Pin 7	Speaker Ring (sleeve)
Pin 8	Speaker Tip (center connections - not stereo)

NOTE: Connect only the pins shown; the rest are reserved.

The input is designed for microphone levels and not for line levels. Practically speaking, this means that devices such as tape decks and CD players must have their signal level attenuated when using them as sources for SPARCstation 1 audio files. Attenuating adapters are available at your local electrical supply or hobby store. The Radio Shack Inline Attenuating Adapter (Cat. No. 274300A) has been used to digitize some source material correctly in SPARCstation 1 development.

The process of recording generates data at the rate of 8KHz (8000) samples per second, so plan accordingly for file sizes and such. There is a software imposed limit of 120 seconds in the `soundtool` demo program. Also note that if your system is running heavily loaded, there may be gaps and pauses due to paging or other activity.

To Begin Recording

Select [Record]. As you record, note that the sound is displayed in the `scope` panel. If it is just barely oscillating, the record volume should be increased (with the `Record volume` slider). If it is pegging out, you need to decrease the record volume. When you are done recording select [Stop] (which is where the [Record] button was). You should then play back the sound to see if the record volume was correctly set, and that your recording is how you want it. Now you may edit or save the sound.

Saving a Sound File

Merely fill in a file name in the `File:` field (as described above) and select [Store] or [Append]. [Store] will ask for confirmation if asked to overwrite an existing file.

Editing Sound

Editing is done with the cursors in the waveform display panel and by appending to an existing file. To cut off the ends of a sample is relatively easy, just move the cursors until the play back is correct, then store the file. You may play back a limited area by moving the cursors. To remove a portion of the sound requires selecting and storing the first part of the sound to be kept, then re-positioning the cursors to select the second part of sound to be kept, then `appending` the second part to the first.

D.2. SPARCstation 1 Graphics Support

SBus Frame Buffers

Monochrome Frame Buffers

Two types of SBus monochrome frame buffers are supported; both implement the `bw2` software interface.

- The analog monochrome board supports 1152 x 900 pixel analog video monochrome monitors. It can also display monochrome graphics on color or grayscale monitors.
- The ECL monochrome board supports 1152 x 900 pixel and 1600 x 1280 pixel ECL video monochrome monitors. The resolution is automatically selected if the correct cable and monitor are used. A jumper is provided on the board to force selection of the high resolution 1600 x 1280 pixel mode.

Color Frame Buffer

The SBus color board is an 1152 x 900 pixel 8 bit memory frame buffer. It implements the `cg3` software interface and emulates that of the `cg4`.

GX Graphics Accelerator Board

The GX graphics accelerator board accelerates the performance of many 2D and 3D graphics applications. It has an 1152 x 900 pixel 8 bit frame buffer which can also be accessed as memory. It implements the `cg6` software interface and emulates the `cg3` and `cg4` software interfaces.

Color Frame Buffer Compatibility

The SBus color and GX frame buffers emulate the `cg4` frame buffer. The following compatibility considerations apply:

- Dynamically linked SunView and pixrect applications run normally and are accelerated when running on the GX.
- Statically linked SunView applications run using the `cg4` emulation mode. They are not accelerated by the GX.
- Statically linked pixrect applications that explicitly manipulate the `cg4` overlay and overlay enable planes may require relinking.
- Applications that access the frame buffer directly with `mmap` may require code changes and recompilation.

Documentation

The following manuals are general references on graphics and GX-related topics:

- The *SunView Programmer's Guide*, Part Number 800-1783-10, and the *SunView System Programmer's Guide*, Part Number 800-1784-10
- The *Pixrect Reference Manual*, Part Number 800-1785-10
- The *SunPHIGS Reference Manual*, Part Number 800-2475-01, and the *SunPHIGS Programming Guide*, Part Number 800-2476-01
- The *SunGKS Reference Manual*, Part Number 800-3560-01

- The *SunGKS Software Installation Guide*, Part Number 800-3561-01
- *"Read This First"* for the *SunGKS Installation Guide*, Part Number 800-3652-01

For information on tuning applications for best GX performance, refer to the *GX Technical Note*.

Known GX Software Problems

The pixrect problems listed also affect the corresponding pixwin functions.

- The line and polygon drawing algorithms of the GX frame buffer differ slightly from other frame buffer and memory pixrects. This will affect applications that expect that the identical pixels are drawn on both the GX screen pixrect and a memory pixrect. For example, the screen image may change when a retained pixwin is refreshed.
- Old statically linked SunView programs running in `cg4` compatibility mode may access the frame buffer during a GX drawing operation, causing incorrect results.

D.3. Using A Second Disk

Moving `/home` to Your Second Disk

If you have a second internal disk which is not being used you may use all or a portion of it for added local storage. (If you are running the pre-installed software, or you did not select it if you installed the software with `suninstall` you probably have all or part your second disk free.)

If you do not intend to use a portion of `/home` for a swap area (see below), enter:

```
@aboy% su
@aboy# newfs /dev/rsd1c
```

then add the following to your `/etc/fstab` file:

```
/dev/sd1c /home 4.2 rw 1 3
```

If you have any *local* (not NFS mounted) files or directories currently in `/home`, as superuser, enter:

```
@aboy# mount /dev/sd1c /mnt
@aboy% cd /home; tar cfh - . | (cd /mnt; tar xpBf -)
```

which will transfer all files living in `/home` and under to the filesystem on the second disk. When this completes without error (that is, no error messages), enter:

```
@aboy# umount /dev/sd1c
```

After this is done, enter (still as superuser):

```
@aboy# mount /home
```

Each time you reboot, the second disk will be mounted and can be used.

If you have a second internal disk that you wish to devote only a part of for extra storage (for example, if you were going to add an extra swap partition), follow the instructions above, but use `/dev/sd1g` instead of `/dev/sd1c`. This gives you approximately 72 MB of extra space as opposed to 90 MB of extra space when you use the entire second disk.

Similar steps may be followed if you have an external disk you would like to use as well, where you would replace the `1` with the appropriate SunOS unit number (for example, `3` for a External Storage Module disk).

Adding Extra Swap Space

Using `sd1b` for Swap Space

If the needs of an application causes swap space to be insufficient (programs will begin to exit with the error message `Not Enough Memory`), check to see how much swap is being used by using the command `pstat -s` (see the `pstat`(8) manual page for details). Any unused disk partition may be added to the list of partitions to be used as swap partitions.

For example, if you have not used the entire second internal disk as a `/home` filesystem, adding the following line to your `/etc/fstab` file will allow that extra space to be added to the swap area.

```
/dev/sd1b swap swap rw 0 0
```

Then, enter the following commands to add `/dev/sd1b` to the swap area; each time you reboot afterwards, `/dev/sd1b` will be used as additional swap area.

```
@aboy% su
@aboy# swapon -a
@aboy# exit
@aboy%
```

Using a Regular File for Extra Swap Space

A regular file may also be created and used as a swap file. If you have the available space in a disk filesystem, you may create a large file with the `mkfile`(8) command (note: do not use the `-n` option) and swap to it in a similar fashion. For example, say you have already used a large portion of your second internal disk as the filesystem for `/home`, but have at least 24 megabytes of space free in it (use the `df`(1) command to check this). As superuser, enter:

```
@aboy# mkfile 16m /home/swap
```

This will create a 16 MB file called `/home/swap`. Then add the following line to `/etc/fstab`:

```
/home/swap swap swap rw 0 0
```

and then, as superuser, enter:

```
@aboy# swapon -a
```

and this file will be added to the list of files and partitions to swap on. Do not remove this file while the system is using it for swapping. Instead, remove the entry in `/etc/fstab` and reboot, and *then* remove the file.

A similar procedure may be followed for swapping on an NFS file given a cooperative server with adequate disk space, in which case the entry in `/etc/fstab` will look like:

```
server:/some/server/pathname swap swap rw 0 0
```

Diskless machines use this method for swap files. Of course, you or a System Administrator must actually create a file on the server for this to work properly.

D.4. Using an External Storage Module

As distributed, the SunOS release 4.1 will support the following SCSI devices:

Device	SunOS Unit #	SCSI target,logical device	Comments
sd	0	3,0	First internal disk
sd	1	1,0	Second internal disk (if present)
sd	3	0,0	External Storage Module disk #1
st	0	4,0	External Storage Module tape
sd	2	2,0	External Storage Module disk #2
sd	1	1,0	External Expansion Module NOTE: Conflicts with 2nd internal disk. Do *not* connect with system if second internal disk present.
sd	2	2,0	Desktop Disk Pack keyed to SCSI target 2
sd	3	0,0	Desktop Disk Pack keyed to SCSI target 0
sd	1	1,0	Desktop Disk Pack keyed to SCSI target 1 NOTE: Conflicts with 2nd internal disk. Do *not* connect with system if second internal disk is present.
sd	0	3,0	Desktop Disk Pack keyed to SCSI target 3 NOTE: Conflicts with 1st internal disk. Do *not* connect with system if first internal disk is present.
st	0	4,0	Desktop Backup Pack keyed to SCSI target 4. NOTE: Conflicts with any tape in an External Storage Module. Do not connect both simultaneously.
st	1	5,0	Desktop Backup Pack keyed to SCSI target 5.
sr	0	6,0	SunCD CD-ROM Drive

Both SunOS and the SPARCstation 1 PROM monitor interpret these devices the same way.

These default mappings may be changed by reconfiguring a kernel to have it understand a different mapping (see the *Rebuilding the Kernel* section of this

chapter), or by using the `eeprom`(8S) command to change the PROM's interpretation of these mappings. It is strongly recommended that any changes to the kernel mapping be reflected by a like change to the PROM mapping.

NOTE *The SPARCstation 1 does not support the early Sun 2 Mass Storage Subsystems (MSS or "Shoebox") with the Sysgen tape drive. The Sysgen tape drive is easily recognized: the tape loads edge-wise rather than end-wise. Connecting this Shoebox to a SPARCstation 1 will have unpredictable results.*

D.5. Parity Recovery

The SPARCstation 1 is robust in its treatment of memory subsystem parity errors. In addition to identifying the SIMM (Single Inline Memory Module) causing the error, the kernel attempts to recover from the error. The kernel classifies the page containing the error into one of three categories, and takes the action indicated:

Error A page that is backed by a file system (or swap) page, and has not been modified since the last time it was read from or written to the file.
Kernel Action In this case, the parity error is treated as a page fault; the current page is invalidated so that a fresh page will be read in from the backing store when execution resumes.

Error A page that is backed by a file system (or swap) page, and *has* been modified since the last time it was read from or written to the file.
Kernel Action Here the parity error is treated as a bus error; a SIGBUS is sent to all processes that have that page mapped. The running process will also have `code` set to BUS_HWERR and `addr` set to the faulting address; see the `sigvec`(2) manual page for signal handling details. (Due to implementation restrictions, other processes that also have the page mapped will not have `code` or `addr` set properly. If the kernel had the page mapped for its own purposes, then a system panic will occur.) Unless a process has made arrangements to catch this signal, it will core dump. In addition, the page is invalidated so that future references to the file will get the old or unmodified page off of backing store.

Error All other pages.
Kernel Action In this case, there is no choice but for the system to panic.

In all cases, the kernel tries to recover the page by writing to, and reading back from, a variety of bit patterns from the failing location. If they all read without error, the page is placed back onto the free list. If an error occurs (either another parity error, or a pattern mismatch), the page is taken out of service until the next system reboot.

Parity Error Messages

Although rare, parity errors may indicate a hardware problem that should be addressed. The system is relatively verbose when one happens; most messages are written to the system console. If a console window is running, the messages appear there if the system recovers. Otherwise they can be extracted from the kernel's crash dump after the panic. See "Handling System Crash Dumps" in the *Sun System and Network Manager's Guide* (in the SPARCstation 1 Owner's Set) for more information.

Certain messages are written to the controlling terminal of the affected process(es).

Here is an annotated description of the messages caused by a memory error (including a parity error):

Synchronous Parity Errors

First, a series of messages describe the error in question; these appear on the system console.

```
Parity error reported at 0x%x, actual address is 0x%x
```

- This message indicates both the hardware-reported virtual address and the location that the kernel has determined is the problem. Due to cache-fill effects, the address the hardware reports is not always the address of the failing location.

```
Parity Error, ctx = 0x%x, virt addr = 0x%x
```

- The hardware context and (adjusted) virtual address of the parity error.

```
pme = %x, phys addr = %x
```

- The contents of the Page Map Entry (pme) for the virtual address in question, and the (computed) physical address of the failing location.

```
Parity Error Register %b
```

- The contents of the parity error register at the time of the error. The register print out would look like this:

```
ff<ERROR,MULTI,TEST,CHECK,ERR00,ERR08,ERR16,ERR24>
```

if all the bits were on.

ERROR	an error has occurred; should be on
MULTI	more than one error has occurred; may be on
TEST	the system is in test mode; should not be on
CHECK	parity checking is enabled; should be on
ERR00	the parity error is associated with bits 0-7
ERR08	the parity error is associated with bits 8-15
ERR16	the parity error is associated with bits 16-23
ERR24	the parity error is associated with bits 24-31
	One or more of the `ERRxx` bits should be on

```
bad chip in %s
```

- For each `ERRxx` bit that is on, the board location of the failing chip is printed. This typically looks like `U0nnn`, where `nnn` is the slot number of the failing SIMM. The reference number of each SIMM slot is silk-screened on the logic board.

In some circumstances, a SIMM number cannot be calculated for the failing address. In that case, one of the following messages may appear:

```
Not a RAM location
```

- A parity error was reported for other than main memory.

```
No U-number can be calculated for this memory address
```

The error is associated with a bank of memory that was not planned for when the kernel was written.

If Recovery is Possible...

Next, one of the following messages may appear, as the kernel determines if recovery is possible:

```
parity recovery: more than one error
```

- The MULTI bit was on in the Parity Error Register.

```
parity recovery: no page structure
```

- The kernel has no information as to the use of this page.

```
parity recovery: no vnode
```

- The kernel can not associate a file with the page.

```
parity recovery: i/o in progress
```

- The page in question was being used for `i/o`.

If a Recovery Candidate

If none of the above messages appears, the page is a candidate for recovery, and one or more of the next set of messages may appear.

If the page was modified, then the following message will appear for each process that has the page mapped:

```
pid %d killed: parity error
```

- This message also appears on the process's controlling terminal.

If the modified page was mapped by the kernel, this message appears:

```
parity recovery: kernel address space
```

- System recovery will not be possible.

If the page was not modified, but the error occurred on the second word of the argument of a load double instruction, and the register that was loaded with the first word of the double word was also used as one of the source address registers, then one of the following messages appears:

```
pid %d killed: parity error on ldd
```

- The load double was issued by a user process. This message also appears on the process's controlling terminal.

```
parity recovery: unrecoverable ldd
```

- The load double was issued by the kernel. System recovery will not be possible.

System Tries to scrub the Failing Location

Next, the system tries to "scrub" the failing location. If successful, the following message will appear:

```
parity error at %x is permanent; page back in service
```

Otherwise, the following messages will appear (one of these messages will appear for each pattern that fails):

```
parity error at %x with pattern %x"
```

```
parity error at %x is permanent; page marked out of service
```

Summary Message Appears

Finally, a "summary" message is printed. If recovery was successful, the following message appears:

```
System operation can continue, will test location anyway
```

Otherwise, the following messages appear:

```
System operation cannot continue, will test location anyway
panic: parity error
(followed by the usual system panic messages)
```

Synchronous, Non-parity Memory Errors

These only happen if a hardware failure occurs. These messages all appear on the system console.

```
Non-parity Synchronous memory error, ctx = 0x%x, virt addr = 0x%x
pme = %x, phys addr = %x
Sync Error Register %b
```

If all the bits were to be on, the register would look like this:

```
80fb<WRITE,INVALID,PROTERR,TIMEOUT,SBBERR,MEMERR,SIZERR,WATCHDOG>
```

WRITE	the error occurred during a write cycle
INVALID	the page map entry was marked invalid
PROTERR	the page map entry denied permission
TIMEOUT	a bus timeout occurred
SBBERR	an Sbus error occurred
MEMERR	a synchronous memory error occurred (should be on)
SIZERR	a size error occurred
WATCHDOG	a watchdog reset has occurred (should not be on)
`panic:`	`sync memory error`

Parity Errors During DVMA Activity

These are always fatal.

```
Dvma Parity Error, ctx = 0x%x, virt addr = 0x%x
pme = %x, phys addr = %x

Parity Error Register %b bad chip in %s
```

or

```
Not a RAM location
```

or

```
No U-number can be calculated for this memory address.
```

or

```
System operation cannot continue, will test location anyway

parity error at %x is permanent; page back in service.
```

or

```
parity error at %x with pattern %x
parity error at %x is permanent; page marked out of service
```

or

```
panic: dvma parity error
```

SCSI Subsystem Messages

You may also see messages from the SCSI subsystem if it is active during a parity error, for example:

```
    sd0: Transport failed; reason 'reset'; retrying.
```

Other Asynchronous Errors

```
Asynchronous memory error, ctx = 0x%x, virt addr = 0x%x
pme = %x, phys addr = %x
Async Error Register %b
```

If all the bits were to be on, the register would look like this:

```
b0<WBACKERR,TIMEOUT,DVMAERR>
```

WBACKERR	error on a buffered write
TIMEOUT	timeout on a buffered write
DVMAERR	error on a DVMA cycle

```
Parity Error Register %b
 bad chip in %s
```

or

```
Not a RAM location
```

or

```
No U-number can be calculated for this memory address.
panic: async memory error
```

What You Should Do:

If "transient" parity errors occur frequently, or if "permanent" parity errors occur, note the SIMM numbers associated with these errors and have the failing memory replaced when convenient. An occasional transient parity error can be caused by cosmic rays and does not indicate a problem with the hardware.

D.6. Rebuilding the SPARCstation 1 Kernel

The process of rebuilding the kernel of a SPARCstation 1 is much simpler than for other Sun systems running SunOS release 4.1. The PROM monitor for SPARCstation 1 in conjunction with the kernel eliminates nearly all of what is in kernel configuration files for other Sun systems pertaining to specifying devices.

Standard SunOS kernel configuration files contain many lines describing bus connections, controller addresses and slave units on controllers. The SPARCstation 1 kernel has simple declarations, which describe devices. For example, in the SunOS release 4.1 GENERIC configuration file, the following lines are all that is necessary to support the listed devices:

```
device-driver   sbus      # 'driver' for sbus interface
device-driver   bwtwo     # monochrome frame buffer
device-driver   cgthree   # 8 bit color frame buffer
device-driver   cgsix     # 8 bit accelerated color frame buffer
device-driver   dma       # 'driver' for dma engine on sbus interface
device-driver   esp       # Emulex SCSI interface
device-driver   fd        # Floppy disk
device-driver   audio     # sound chip
device-driver   le        # Lance ethernet
device-driver   zs        # UARTs
```

That is all that is necessary to specify the inclusion of device drivers and kernel support for these devices; the PROM and the kernel auto-configuration code do the rest.

There are two pieces of additional information that this mechanism does not provide.

`flags` Word to Ignore CARRIER DETECT

Users familiar with other SunOS kernel configuration files may ask where the `flags` word goes that specifies ignoring CARRIER DETECT for the `zs` (UART) driver. This has been replaced by data in the EEPROM, which the kernel auto-configuration code asks the PROM to fetch. These fields (`ttya-ignore-cd` and `ttyb-ignore-cd`) may be set either using either the `eeprom`(8S) command, or commands from the PROM monitor (see `monitor`(8S)).

The other piece of information needed is the specification of what possible SCSI disks and tapes may be connected to this system, and where to find them. This must be specified rather than determined by auto-configuration because Sun supports some non-CCS (Command Command Set) devices which do not respond to the SCSI INQUIRY command (which would normally determine what kind of a device it is).

The syntax of describing where to look for these disks and tapes comes in two parts. The first is a declaration that there is a SCSI bus connected to the SPARCstation 1:

```
scsibus0 at esp      # declare first scsi bus
```

Declaring SCSI Disks and Tapes

The second part declares what disks and tapes might be connected to this SCSI bus:

```
disk sd0 at scsibus0 target 3 lun 0 # first hard SCSI disk
disk sd1 at scsibus0 target 1 lun 0 # second hard SCSI disk
disk sd2 at scsibus0 target 2 lun 0 # third hard SCSI disk
disk sd3 at scsibus0 target 0 lun 0 # fourth hard SCSI disk
tape st0 at scsibus0 target 4 lun 0 # first SCSI tape
tape st1 at scsibus0 target 5 lun 0 # second SCSI tape
tape sr0 at scsibus0 target 6 lun 0 # CD-ROM
```

To be more specific, the first line above says that there may be a disk, which we will call `sd0`, on `scsibus` number `0`, at SCSI target address `3`, logical unit `0`. These declarations merely state that there *may* be this device at that location; look for it when booting, and (if not found) look again if a program attempts to open it while the system is running.

SPARCstation 1 kernels are rebuilt for the same reasons kernels are rebuilt on other Sun workstations: to save main memory (and improve performance) by specifying precisely what is needed for the applications used, and adding the required drivers and modules. There is a set of standard configuration files that come with SunOS release 4.1 for the SPARCstation 1 variant requirements. They are the GENERIC, SDST60, NFS60 and DL60 configuration files (which can be found in `/usr/share/sys/sun4c/conf`). The GENERIC configuration file covers (as its name implies) all the bases. The SDST60 configuration is, for all intents and purposes, a smaller version more tuned to specific SPARCstation 1 needs. The NFS60 file is the same as the SDST60 file, but specifies a root and swap on NFS. This is for environments which want to use any disks connected to a SPARCstation 1 solely for user data and relegate all system functions to a server

on the network. The DL60 configuration file is optimized for SPARCstation 1 machines that do not have any disks.

D.7. PROM User Interface

The **Open PROM** interface provided with the SPARCstation 1 is a significant departure from the Monitor supplied with previous Sun products. A "compatibility mode" is included to ease the transition to this new user interface.

The SPARCstation 1 normally boots automatically after powering-up, making this new interface invisible to the casual user. However, any time the `L1-A` (or `break` on a serial-port console) sequence is pressed, the PROM Monitor is activated. From the new `ok` prompt, the interpreter allows the user to perform a wide variety of commands to test the system, configure non-volatile parameters, or debug hardware and/or software. Built-in `help` is also provided for many commands.

Compatibility Mode

When the `PROM` interface is activated, the first message displayed is:

```
Type b (boot), c (continue), or n (new command mode)
>
```

This is the `Compatibility Mode`, which emulates a subset of the old-style Sun monitor. The only commands currently supported are the three most common: `boot`, `continue`, and `new`.

`b` may be followed by optional boot specifiers in exactly the same style as previous Sun products.

Boot Specifier	Device
sd(0,0,0)	First SCSI Disk
sd(0,1,0)	Second SCSI Disk
sd(0,2,0)	Third SCSI Disk
sd(0,3,0)	Fourth SCSI Disk
st(0,0,0)	First SCSI Tape
st(0,1,0)	Second SCSI Tape
fd(0,0,0)	Floppy Diskette

`c` continues any program which was interrupted with `L1-A` or `BREAK`.

`n` is a new command which brings up the `ok` prompt, enabling all other available `PROM` commands.

From the `ok` prompt, the command `old-mode` brings you back to the > prompt. This is provided for convenience, but should not be necessary, as the `boot`, `go`, and `help` commands are available from the `ok` prompt as well.

Consult the *PROM Toolkit User's Guide* for more information about this new interface.

D.8. SCSI Unit Numbering and the SPARCstation 1 PROM

The SPARCstation 1 PROM restores (for SCSI) the Sun paradigm of the boot address triple (see `boot`(8S)) for disks:

```
>b xx(C,U,P)
```

where *xx* is a two character device mnemonic (for example, `xy`, `xd`, or `sd`), *C* is the controller number, *U* is the unit on that controller, and *P* is the partition on that disk.

The Bad News

On previous Sun platforms, the specific SCSI address of a disk was "overloaded" into the unit field of this boot address triple. That is, a SCSI disk has an address of SCSI target id and logical unit ("lun") at that target, so the unit field for the boot command line was calculated as

```
8 * target + lun
```

This is somewhat confusing, since the SunOS, "unit number" of a disk is something quite different.

The Good News

For SPARCstation 1 (both SunOS and the PROM), unit *means* unit. The kernel configuration file for SunOS establishes the relationship between unit number and specific SCSI address (for example, `sd2 at scsibus0 target 2 lun 0`) for SunOS. For the PROM, there is a property called `sd-targets` which does likewise. `sd-targets` is a string of digits, where the PROM takes the unit number (the *U* portion of the boot line above) and uses that to index into the `sd-targets` string to retrieve the SCSI target id that that disk unit number maps to. Thus, for SPARCstation 1, the default setting for sd-targets is:

```
sd-targets  "31204567"
```

or, disk unit 0 is at SCSI target 3, disk unit 1 is at SCSI target 1, etc.

D.9. EEPROM Differences

<mon/eeprom.h>

The software interface to the EEPROM in machines using the Open Boot PROM (such as the SPARCstation 1) is different from previous Sun Workstations. The `eeprom`(8) command has been modified to account for these differences. If you wish to examine and change the EEPROM settings from SunOS release 4.1, you may use this command as always. See the `eeprom`(8) man page. Programs that specifically read or modify the EEPROM must be rewritten to conform to the new interface.

Table D-1 *SPARCstation 1 PROM Parameters and Values*

Parameter Name	Value	Default Value
sunmon-compat?	false	true
selftest-#megs	1	1
oem-logo		
oem-logo?	false	false
oem-banner		
oem-banner?	false	false
ttyb-mode	9600,8,n,1,-	9600,8,n,1,-
ttya-mode	9600,8,n,1,-	9600,8,n,1,-
ttyb-rts-dtr-off	false	false
ttyb-ignore-cd	true	true
ttya-rts-dtr-off	false	false
ttya-ignore-cd	true	true
sbus-probe-list	0123	0123
fcode-debug?	false	false
screen-#columns	80	80
screen-#rows	34	34
boot-from-diag	le()vmunix	le()vmunix
boot-from	kadb	vmunix
auto-boot?	false	true
watchdog-reboot?	false	false
input-device	keyboard	keyboard
output-device	screen	screen
keyboard-click?	false	false
sd-targets	31204567	31204567
st-targets	45670123	45670123
scsi-initiator-id	7	7
hardware-revision		
last-hardware-update		
testarea	0	0
mfg-switch?	false	false
diag-switch?	false	false

These parameters are under the same usage constraints as others on `eeprom`(8): You must be superuser to change parameters, and characters special to the shell (for example, parentheses) must be quoted with single or double quotes.

`ttya-ignore-cd` and `ttyb-ignore-cd`

The `flags` parameter of the `device` line in the `/etc/ttytab` configuration file, is usually used to indicate that the corresponding port should be treated as hard-wired with carrier detect always present. This option is set by the presence or absence of the keyword "`local`" in `/etc/ttytab` for the serial ports described in `/etc/ttytab`.

Alternatively, the `ttya-ignore-cd` and `ttyb-ignore-cd` parameters can be used to set to ignore or not ignore carrier detect. A value of `true` indicates that the carrier-detect signal should be ignored, and is the default. A value of `false` indicates that carrier-detect should not be ignored; an open of the port will be blocked until the connected modem indicates that carrier is present.

These options can be changed either from the PROM interface:

```
> n
ok setenv ttyb-ignore-cd false
ok
```

or by using the `eeprom`(8S) command:

```
@aboy% eeprom ttyb-ignore-cd=false
```

Remember, a `tty`[*ab*]`-ignore-cd` option is overridden by the presence or absence of the keyword "`local`" in `/etc/ttytab`. See the `ttysoftcar`(8), `eeprom`(8), and `ttytab`(5) manual pages for more information.

E

Table of Contents for SunOS Release 4.1 Tapes and Diskettes

E.1. TABLE OF CONTENTS for SunOS release 4.1 Tapes

The following are illustrations of the correct Tables Of Contents for SunOS release 4.1 tapes, diskettes, and compact disc. They appear in the following order:

- **Compact Disc** (Contains files for all architectures)
- **sun3** (1/4-inch tape)
- **sun3** (1/2-inch tape)
- **sun3x** (1/4-inch tape)
- **sun3x** (1/2-inch tape)
- **sun4** (1/4-inch tape)
- **sun4** (1/2-inch tape)
- **sun4c** (Diskettes)
- **sun4c** (1/4-inch tape)
- **sun4c** (1/2-inch tape)

CD-ROM Directory Structure

The compact disc version of SunOS release 4.1 contains all of the files needed to install that release on all Sun workstations. A recursive `ls` of the disc will return the following:

```
mousetrap% ls -R

_copyright      avail_arches       export          patches

export:
exec    share

export/exec:
kvm                     sun3_sunos_4_1
proto_root_sunos_4_1    sun4_sunos_4_1

export/exec/kvm:
sun3_sunos_4_1  sun3x_sunos_4_1 sun4_sunos_4_1  sun4c_sunos_4_1

export/exec/kvm/sun3_sunos_4_1:
kvm             miniroot_sun3   sys             xdrtoc

export/exec/kvm/sun3x_sunos_4_1:
kvm             miniroot_sun3x  sys             xdrtoc

export/exec/kvm/sun4_sunos_4_1:
kvm             miniroot_sun4   sys             xdrtoc

export/exec/kvm/sun4c_sunos_4_1:
kvm             miniroot_sun4c  sys             xdrtoc

export/exec/sun3_sunos_4_1:
debugging               security                tli
demo                    shlib_custom            user_diag
games                   sunview_demo            usr
graphics                sunview_programmers     uucp
install                 sunview_users           versatec
networking              system_v
rfs                     text

export/exec/sun4_sunos_4_1:
debugging               security                tli
demo                    shlib_custom            user_diag
games                   sunview_demo            usr
graphics                sunview_programmers     uucp
install                 sunview_users           versatec
networking              system_v
rfs                     text

export/share:
sunos_4_1
```

```
export/share/sunos_4_1:
manual

patches:
sunos_4_1

patches/sunos_4_1:
_readme                  patch_ipc
patch_cplusplus_2_0      patch_taac
```

sun3 (1/4-inch tape)

Table of Contents for the `sun3` kernel architecture (1/4-inch tape).

```
SunOS 4.1 700-1877-14 Rev. A of Mon Feb 12 16:36:13 PST 1990 from Sun Release
Engineering.
ARCH sun3
VOLUME -1
 Vol File              Name       Size   Type
   1    0              boot      32768   image
   1    1            XDRTOC       4096   toc
   1    2              copy      33280   image
   1    3         mini-root    6144000   image
   1    4             munix     705024   image
   1    5      munixfs.tape    1638400   image
   1    6              root     229376   tar
   1    7               usr   23052288   tar
   1    8               Kvm    3276800   tar
   1    9           Install     761856   tar
   1   10        Networking     966656   tar
   1   11               Sys    3973120   tar
   1   12          System_V    3588096   tar
   1   13               TLI      40960   tar
   1   14         Copyright       1024   image
   2    0              boot      32768   image
   2    1            XDRTOC       4096   toc
   2    2               RFS     860160   tar
   2    3         Debugging    2080768   tar
   2    4     SunView_Users    2490368   tar
   2    5 SunView_Programmers 1572864   tar
   2    6      SunView_Demo     524288   tar
   2    7      Shlib_Custom    1277952   tar
   2    8              Text     647168   tar
   2    9         User_Diag    2514944   tar
   2   10          Graphics    2752512   tar
   2   11              uucp     557056   tar
   2   12            Manual    7471104   tar
   2   13              Demo    5472256   tar
   2   14             Games    2940928   tar
   2   15          Versatec    6086656   tar
   2   16          Security     262144   tar
   2   17         Patch_IPC     139264   tar
   2   18     Patch_C++_2.0    2420736   tar
   2   19        Patch_TAAC      20480   tar
   2   20         Copyright       1024   image
```

sun3 (1/2-inch tape)

Table of Contents for the `sun3` kernel architecture (1/2-inch tape).

```
SunOS 4.1 700-1878-14 Rev. A of Mon Feb 12 16:36:14 PST 1990 from Sun Release
Engineering.
ARCH sun3
VOLUME -1
 Vol File             Name       Size   Type
   1    0             boot      32768   image
   1    1           XDRTOC       4096   toc
   1    2             copy      33280   image
   1    3        mini-root    6144000   image
   1    4            munix     705024   image
   1    5     munixfs.tape    1572864   image
   1    6             root     229376   tar
   1    7              usr   23052288   tar
   1    8              Kvm    3276800   tar
   1    9          Install     761856   tar
   1   10       Networking     966656   tar
   1   11        Copyright       1024   image
   2    0             boot      32768   image
   2    1           XDRTOC       4096   toc
   2    2              Sys    3973120   tar
   2    3         System_V    3588096   tar
   2    4              TLI      40960   tar
   2    5              RFS     860160   tar
   2    6        Debugging    2080768   tar
   2    7    SunView_Users    2490368   tar
   2    8 SunView_Programmers 1572864   tar
   2    9     SunView_Demo     524288   tar
   2   10     Shlib_Custom    1277952   tar
   2   11             Text     647168   tar
   2   12        User_Diag    2514944   tar
   2   13         Graphics    2752512   tar
   2   14             uucp     557056   tar
   2   15           Manual    7471104   tar
   2   16             Demo    5472256   tar
   2   17        Copyright       1024   image
   3    0             boot      32768   image
   3    1           XDRTOC       4096   toc
   3    2            Games    2940928   tar
   3    3          Versatec    6086656   tar
   3    4         Security     262144   tar
   3    5        Patch_IPC     139264   tar
   3    6    Patch_C++_2.0    2420736   tar
   3    7       Patch_TAAC      20480   tar
   3    8        Copyright       1024   image
```

sun3x (1/4-inch tape)

Table of Contents for the `sun3x` kernel architecture (1/4-inch tape).

```
SunOS 4.1 700-1881-14 Rev. A of Mon Feb 12 15:46:51 PST 1990 from Sun Release
Engineering
ARCH sun3x
VOLUME -1
 Vol File             Name       Size   Type
   1    0             boot      32768   image
   1    1           XDRTOC       4096   toc
   1    2             copy      33280   image
   1    3        mini-root    6144000   image
   1    4            munix     713216   image
   1    5     munixfs.tape    1638400   image
   1    6             root     229376   tar
   1    7              usr   23068672   tar
   1    8              Kvm    3334144   tar
   1    9          Install     761856   tar
   1   10       Networking     966656   tar
   1   11              Sys    3973120   tar
   1   12        Copyright       1024   image
   2    0             boot      32768   image
   2    1           XDRTOC       4096   toc
   2    2         System_V    3588096   tar
   2    3              TLI      40960   tar
   2    4              RFS     860160   tar
   2    5        Debugging    2080768   tar
   2    6    SunView_Users    2490368   tar
   2    7 SunView_Programmers 1572864   tar
   2    8     SunView_Demo     524288   tar
   2    9     Shlib_Custom    1277952   tar
   2   10             Text     647168   tar
   2   11        User_Diag    2514944   tar
   2   12         Graphics    2752512   tar
   2   13             uucp     557056   tar
   2   14           Manual    7471104   tar
   2   15             Demo    5472256   tar
   2   16            Games    2940928   tar
   2   17          Versatec    6086656   tar
   2   18         Security     262144   tar
   2   19        Patch_IPC     139264   tar
   2   20    Patch_C++_2.0    2420736   tar
   2   21       Patch_TAAC      20480   tar
   2   22        Copyright       1024   image
```

sun3x (1/2-inch tape)

Table of Contents for the `sun3x` kernel architecture (1/2-inch tape).

```
SunOS 4.1 700-1882-14 Rev. A of Mon Feb 12 16:07:02 PST 1990 from Sun Release
Engineering
ARCH sun3x
VOLUME -1
 Vol File             Name        Size   Type
   1    0             boot       32768   image
   1    1           XDRTOC        4096   toc
   1    2             copy       33280   image
   1    3        mini-root     6144000   image
   1    4            munix      713216   image
   1    5     munixfs.tape     1572864   image
   1    6             root      229376   tar
   1    7              usr    23068672   tar
   1    8              Kvm     3334144   tar
   1    9          Install      761856   tar
   1   10        Copyright        1024   image
   2    0             boot       32768   image
   2    1           XDRTOC        4096   toc
   2    2       Networking      966656   tar
   2    3              Sys     3973120   tar
   2    4         System_V     3588096   tar
   2    5              TLI       40960   tar
   2    6              RFS      860160   tar
   2    7        Debugging     2080768   tar
   2    8    SunView_Users     2490368   tar
   2    9 SunView_Programmers 1572864   tar
   2   10     SunView_Demo      524288   tar
   2   11     Shlib_Custom     1277952   tar
   2   12             Text      647168   tar
   2   13        User_Diag     2514944   tar
   2   14         Graphics     2752512   tar
   2   15             uucp      557056   tar
   2   16        Copyright        1024   image
   3    0             boot       32768   image
   3    1           XDRTOC        4096   toc
   3    2           Manual     7471104   tar
   3    3             Demo     5472256   tar
   3    4            Games     2940928   tar
   3    5         Versatec     6086656   tar
   3    6         Security      262144   tar
   3    7        Patch_IPC      139264   tar
   3    8    Patch_C++_2.0     2420736   tar
   3    9       Patch_TAAC       20480   tar
   3   10        Copyright        1024   image
```

sun4 (1/4-inch tape)

Table of Contents for the `sun4` kernel architecture (1/4-inch tape).

```
SunOS 4.1 700-1879-14 Rev. A of Mon Feb 12 15:55:59 PST 1990 from Sun Release
Engineering.
ARCH sun4
VOLUME -1
 Vol File              Name       Size   Type
   1    0              boot      49152   image
   1    1            XDRTOC       4096   toc
   1    2              copy      41472   image
   1    3         mini-root    6144000   image
   1    4             munix     950784   image
   1    5      munixfs.tape    2150400   image
   1    6              root     229376   tar
   1    7               usr   22585344   tar
   1    8               Kvm    4382720   tar
   1    9           Install     860160   tar
   1   10        Networking    1064960   tar
   1   11               Sys    4734976   tar
   1   12               TLI      49152   tar
   1   13               RFS     950272   tar
   1   14         Copyright       1024   image
   2    0              boot      49152   image
   2    1            XDRTOC       4096   toc
   2    2          System_V    4079616   tar
   2    3         Debugging    2924544   tar
   2    4     SunView_Users    2719744   tar
   2    5 SunView_Programmers 1875968   tar
   2    6      SunView_Demo     524288   tar
   2    7      Shlib_Custom    1400832   tar
   2    8              Text     729088   tar
   2    9         User_Diag    2768896   tar
   2   10          Graphics    1826816   tar
   2   11              uucp     622592   tar
   2   12            Manual    7471104   tar
   2   13              Demo    5505024   tar
   2   14             Games    3211264   tar
   2   15          Versatec    6094848   tar
   2   16          Security     319488   tar
   2   17         Patch_IPC     139264   tar
   2   18    Patch_C++_2.0    2420736   tar
   2   19        Patch_TAAC      20480   tar
   2   20         Copyright       1024   image
```

sun4 (1/2-inch tape)

Table of Contents for the `sun4` kernel architecture (1/2-inch tape).

```
SunOS 4.1 700-1880-14 Rev. A of Mon Feb 12 16:07:49 PST 1990 from Sun Release
Engineering.
ARCH sun4
VOLUME -1
 Vol File              Name        Size   Type
   1    0              boot       49152   image
   1    1           XDRTOC         4096   toc
   1    2              copy       41472   image
   1    3         mini-root     6144000   image
   1    4             munix      950784   image
   1    5      munixfs.tape     2097152   image
   1    6              root      229376   tar
   1    7               usr    22585344   tar
   1    8               Kvm     4382720   tar
   1    9           Install      860160   tar
   1   10        Networking     1064960   tar
   1   11               Sys     4734976   tar
   1   12          System_V     4079616   tar
   1   13               TLI       49152   tar
   1   14               RFS      950272   tar
   1   15         Debugging     2924544   tar
   1   16     SunView_Users     2719744   tar
   1   17 SunView_Programmers  1875968   tar
   1   18      SunView_Demo      524288   tar
   1   19      Shlib_Custom     1400832   tar
   1   20              Text      729088   tar
   1   21         User_Diag     2768896   tar
   1   22          Graphics     1826816   tar
   1   23              uucp      622592   tar
   1   24            Manual     7471104   tar
   1   25              Demo     5505024   tar
   1   26             Games     3211264   tar
   1   27          Versatec     6094848   tar
   1   28          Security      319488   tar
   1   29         Patch_IPC      139264   tar
   1   30     Patch_C++_2.0     2420736   tar
   1   31        Patch_TAAC       20480   tar
   1   32         Copyright        1024   image
```

sun4c (Diskettes)

Table of Contents for the `sun4c` kernel architecture (Diskettes).

```
SunOS 4.1 702-1188-10 Rev. A of Mon Feb 12 16:40:28 PST 1990 from Sun Release
Engineering
ARCH sun4c
VOLUME -1
 Vol File              Name       Size   Type
   0   -1                XDRTOC       4096   toc
   1    0                  root       71213   tarZ
   1 71680                  usr    10143889   tarZ
   8 23040                  Kvm     2497176   tarZ
   9 1064448             Install      444823   tarZ
  10 53248            Networking      464675   tarZ
  10 518144                 Sys     2104861   tarZ
  11 1167360            System_V     1889859   tarZ
  13 145408                 TLI       22338   tarZ
  13 167936                 RFS      398963   tarZ
  13 567296           Debugging     1565676   tarZ
  14 676864        SunView_Users     1156037   tarZ
  15 376832 SunView_Programmers      860529   tarZ
  15 1237504        SunView_Demo      196687   tarZ
  15 1434624        Shlib_Custom      783639   tarZ
  16 762368                Text      347977   tarZ
  16 1110528           User_Diag     1404373   tarZ
  17 1058816            Graphics      836941   tarZ
  18 439808                uucp      286755   tarZ
  18 727040              Manual     2625557   tarZ
  20 440832                Demo     2579903   tarZ
  22 108544               Games     1786703   tarZ
  23 439296             Versatec     2398301   tarZ
  24 1381888            Security      247637   tarZ
  -1   -1             Copyright        1024   image
```

sun4c (1/4-inch tapes)

Table of Contents for the `sun4c` kernel architecture (1/4-inch tape).

```
SunOS 4.1 700-2521-10 Rev. A of Mon Feb 12 16:32:40 PST 1990 from Sun Release
Engineering
ARCH sun4c
VOLUME -1
 Vol File             Name        Size   Type
   1    0             boot       28672   image
   1    1           XDRTOC        4096   toc
   1    2             copy       16896   image
   1    3        mini-root     6144000   image
   1    4            munix      868864   image
   1    5     munixfs.tape     2150400   image
   1    6             root      229376   tar
   1    7              usr    22585344   tar
   1    8              Kvm     4022272   tar
   1    9          Install      860160   tar
   1   10       Networking     1064960   tar
   1   11              Sys     4669440   tar
   1   12        Copyright        1024   image
   2    0             boot       28672   image
   2    1           XDRTOC        4096   toc
   2    2         System_V     4079616   tar
   2    3              TLI       49152   tar
   2    4              RFS      950272   tar
   2    5        Debugging     2924544   tar
   2    6    SunView_Users     2719744   tar
   2    7 SunView_Programmers 1875968   tar
   2    8     SunView_Demo      524288   tar
   2    9     Shlib_Custom     1400832   tar
   2   10             Text      729088   tar
   2   11        User_Diag     2768896   tar
   2   12         Graphics     1826816   tar
   2   13             uucp      622592   tar
   2   14           Manual     7471104   tar
   2   15             Demo     5505024   tar
   2   16            Games     3211264   tar
   2   17         Versatec     6094848   tar
   2   18         Security      319488   tar
   2   19        Patch_IPC      139264   tar
   2   20    Patch_C++_2.0     2420736   tar
   2   21       Patch_TAAC       20480   tar
   2   22        Copyright        1024   image
```

sun4c (1/2-inch tapes)

Table of Contents for the `sun4c` kernel architecture (1/2-inch tapes).

```
SunOS 4.1 700-2522-10 Rev. A of Mon Feb 12 16:40:26 PST 1990 from Sun Release
Engineering
ARCH sun4c
VOLUME -1
 Vol File              Name       Size   Type
   1    0              boot      28672   image
   1    1           XDRTOC        4096   toc
   1    2              copy      16896   image
   1    3         mini-root    6144000   image
   1    4             munix     868864   image
   1    5      munixfs.tape    2097152   image
   1    6              root     229376   tar
   1    7               usr   22585344   tar
   1    8               Kvm    4022272   tar
   1    9           Install     860160   tar
   1   10        Networking    1064960   tar
   1   11               Sys    4669440   tar
   1   12          System_V    4079616   tar
   1   13               TLI      49152   tar
   1   14               RFS     950272   tar
   1   15         Debugging    2924544   tar
   1   16     SunView_Users    2719744   tar
   1   17 SunView_Programmers 1875968   tar
   1   18      SunView_Demo     524288   tar
   1   19      Shlib_Custom    1400832   tar
   1   20              Text     729088   tar
   1   21         User_Diag    2768896   tar
   1   22          Graphics    1826816   tar
   1   23              uucp     622592   tar
   1   24            Manual    7471104   tar
   1   25              Demo    5505024   tar
   1   26             Games    3211264   tar
   1   27          Versatec    6094848   tar
   1   28          Security     319488   tar
   1   29         Patch_IPC     139264   tar
   1   30     Patch_C++_2.0    2420736   tar
   1   31        Patch_TAAC      20480   tar
   1   32         Copyright       1024   image
```

Index

O

P

Q

S

T

U

W

Y