sun®
microsystems

Release 3.4 Manual
for the Sun Workstation

Part Number: 800-1614-10
Revision: A of 14 May 1987

Contents

Tables

Figures

1

Introduction

1

Introduction

This manual explains how to install the new Release 3.4 software and describes the many features and bug fixes included in the release. The text is organized as follows:

- Chapter 1 gives an overview of the document and the conventions it uses.
- Chapter 2 contains the installation instructions for upgrading to Release 3.4.
- Chapter 3 explains how to reconfigure the kernel for Release 3.4.
- Chapter 4 describes enhancements and bug fixes made in Release 3.3, including an optional disconnect/reconnect facility, a subnet capability, enhancements to the Pascal compiler, and assorted bug fixes.
- Chapter 5 describes enhancements included in Release 3.4 for the following areas: graphics, SunView, kernel, networking, and the `termcap` file.
- Chapter 6 describes bug fixes made since Release 3.3.
- Chapter 7 contains additions and changes made to Release 3.2 manuals on the following subjects: languages, graphics, SunView , and system administration.
- Appendix A explains how to install the SunPro optional software.
- Appendix B lists optional software for Release 3.4.
- Appendix C contains a table of all additions and changes made to the both the online and printed manual pages. Manual pages that have been updated make up the remainder of this appendix.

1.1. Supporting Documentation

If you need more information about topics in this manual, refer to the following:

- *Installing UNIX on the Sun Workstation* for Release 3.2 (800-1521)
- *Commands Reference Manual for the Sun Workstation* (800-1295)
- *Sun Diagnostics Manual* (800-1361)
- *System Administration Manual for the Sun Workstation* (800-1323)

1.2. Documentation Conventions

The following conventions are used in the procedures and examples throughout this document:

- prompts from the system are printed in `listing font like this.`
- Information that you type in response to the system is shown in **`boldface listing font like this`**. Type everything shown in boldface exactly as it appears.
- Where parts of a command are shown in *italic text like this*, they refer to a variable that you have to substitute from a selection; it is up to you to make the proper substitution.
- Dialogues between you and the system are enclosed in gray boxes (except in Chapter 2) like the following:

```
host% ls personnel.rec
amina         azhar       bb            cameron
ernest        farrasha    gregorio      jim
linda         michael     stefania      susan
tony          turia
```

- Sections of program code are enclosed in clear boxes like the following:

```
int test[100];

main()
{
   register int a, b, c, d, e, f;

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

2

3.4 Release Upgrade Installation

2

3.4 Release Upgrade Installation

This chapter guides you through the Release 3.4 upgrade. The upgrade procedures in this chapter are designed to upgrade systems currently running Release 3.2 and Release 3.3. Release 3.4 is completely compatible with these previous releases; any program developed to run under them runs under 3.4. However, you have to recompile to take advantage of the new features.

The procedures in this chapter support the following configurations:

- standalone machines with a local tape drive and disk(s)
- standalone machines with disk(s) but without a local tape drive
- homogeneous servers with a local tape drive and disk(s)
- homogeneous servers with disk(s) but without a local tape drive
- heterogeneous servers with a local tape drive and disk(s)
- heterogeneous servers with disk(s) but without a local tape drive

Please read through this chapter a few times to familiarize yourself with the procedures before you start the upgrade. Pay particular attention to the information in Section 2.3, "Requirements for Each Configuration," because it describes the requirements for each configuration before you can upgrade it to Release 3.4.

2.1. What Is on the Distribution Tape?

The release software is on either a single 1/2-inch tape or two 1/4-inch tapes for a specific architecture. The lists of files on each type of tape follow.

Table 2-1 *Contents of the 1/2-Inch Distribution Tape*

File 1:	Copyright file
File 2:	Table of contents
File 3:	Upgrade utilities (tar format)
File 4:	root files (tar format)
File 5:	pub files (tar format)
File 6:	Client files (tar format)
File 7:	usr files (tar format)
File 8:	System files (tar format)
File 9:	Networking files (tar format)
File 10:	Debugging files (tar format)
File 11:	SunView user files (tar format)
File 12:	SunView programmer files (tar format)
File 13:	SunView source files (tar format)
File 14:	Text processing tools (tar format)
File 15:	Fortran files (tar format)
File 16:	Pascal files (tar format)
File 17:	Graphics files (tar format)
File 18:	Profiled libraries (tar format)
File 19:	System V files (tar format)
File 20:	Demonstration files (tar format)
File 21:	Manual pages (tar format)
File 22:	Game files (tar format)
File 23:	SunPro files (tar format)
File 24:	Sunview profiled libraries (tar format)
File 25:	Copyright file (tar format)

Table 2-2 *Contents of the 1/4-Inch Distribution Tapes*

Tape 1

File 1:	Copyright file
File 2:	Table of contents
File 3:	Upgrade utilities (tar format)
File 4:	root files (tar format)
File 5:	pub files (tar format)
File 6:	Client files (tar format)
File 7:	usr files (tar format)
File 8:	System files (tar format)
File 9:	Networking files (tar format)
File 10:	Debugging files (tar format)
File 11:	SunView user files (tar format)
File 12:	SunView programmer files (tar format)
File 13:	SunView source files (tar format)
File 14:	Text processing tools (tar format)
File 15:	Fortran files (tar format)
File 16:	Pascal files (tar format)
File 17:	Graphics files (tar format)
File 18:	Profiled libraries (tar format)
File 19:	Copyright file

Tape 2

File 1:	Copyright file
File 2:	Table of contents
File 3:	System V files (tar format)
File 4:	Demonstration files (tar format)
File 5:	Manual pages (tar format)
File 6:	Game files (tar format)
File 7:	SunPro files (tar format)
File 8:	Sunview profiled libraries (tar format)
File 9:	Copyright file (tar format)

Terminology

Below are some of the specific terms used in this chapter. It assumes that you have some experience in installing UNIX on a Sun workstation. If you do not understand some of the following instructions or terms, refer to *Installing UNIX on the Sun Workstation* and the *System Administration Manual for the Sun Workstation* for details.

Tape host The system with the tape drive.

Homogeneous server A server that supports clients of its own architecture.

Heterogeneous server A server that supports both Sun-2 (MC68010) and Sun-3 (MC68020) clients.

2.2. Requirements for Each Configuration

This section lists the requirements of each configuration that the upgrade procedures support. Read the information in the section pertaining to your particular system configuration very carefully. Your system MUST meet all the requirements listed before you can properly upgrade it.

Upgrading with a remote tape drive is slower than upgrading with a local tape drive. Unless you have no other choice, upgrading with a local tape drive is highly recommended. Client partitions that are commented out in `/etc/nd.local` on a server will not be upgraded. Refer to Section **2.6** of this chapter for instructions on how to upgrade these partitions.

Standalone System with Local Tape Drive

- This configuration is a standalone system running 3.2 or 3.3 with a local tape drive.
- The release tape(s) should be appropriate for the standalone system's architecture.

Standalone System with Remote Tape Drive

- This configuration is a standalone system running 3.2 or 3.3.
- The release tape(s) should be appropriate for the standalone system's architecture and the tape host's tape drive type.
- The tape host must be reachable from the standalone system through the Ethernet. The tape host and the standalone system must be on the same network.
- `/etc/hosts` on a standalone system must contain the Internet address of the tape host if `yp` is not running on the standalone system. If `yp` is running, the `/etc/hosts` file on the master `yp` server must contain the Internet address of the tape host.
- `/etc/hosts` on the tape host must contain the Internet address of the standalone system if `yp` is not running. If `yp` is running, the `/etc/hosts` file on the master `yp` server must contain the Internet address of the standalone system.
- `/.rhosts` on the tape host must contain the name of the standalone system.

Homogeneous Server with Local Tape Drive

- This configuration is a homogeneous server running 3.2 or 3.3 with a local tape drive.
- The release tape(s) should be appropriate for the system's architecture and tape drive type.
- Make sure that `/etc/hosts` and `/etc/ethers` on the homogeneous server contain correct Internet and Ethernet addresses of all clients supported by the server if it is not running `yp`. If it is running `yp`, the `/etc/hosts` and `/etc/ethers` files on the master `yp` server must contain this information.

Homogeneous Server with Remote Tape Drive

- □ This configuration is a homogeneous server running 3.2 or 3.3.
- □ The release tape(s) should be appropriate for the system's architecture and the tape host's tape drive type.
- □ Make sure that /etc/hosts and /etc/ethers on the homogeneous server contain correct Internet and Ethernet addresses of all clients supported by the server if it is not running yp. If it is running yp, the /etc/hosts and /etc/ethers files on the master yp server must contain this information.
- □ The tape host must be reachable from the homogeneous server through the Ethernet. The tape host and the homogeneous server must be on the same network.
- □ /etc/hosts on the homogeneous server must contain the Internet address of the tape host if yp is not running on the homogeneous server. If yp is running, the /etc/hosts file on the master yp server must contain the Internet address of the standalone system.
- □ /etc/hosts on the tape host must contain the Internet address of the homogeneous server if yp is not running. If yp is running, the /etc/hosts file on the master yp server must contain the Internet address of the standalone system.
- □ /.rhosts on the tape host must contain the name of the homogeneous server.

Heterogeneous Server with Local Tape Drive

- □ This configuration is a heterogeneous server running 3.2 or 3.3 with a local tape drive.
- □ The release tapes must be appropriate for both architectures and suitable for the local tape drive. For example, you must have suitable Sun-2 and Sun-3 tapes, and tapes must both be 1/4 or 1/2 inch. You cannot mix media types.
- □ Make sure that /etc/hosts and /etc/ethers on the heterogeneous server contain correct Internet and Ethernet addresses of all clients supported by the server if it is not running yp. If it is running yp, the /etc/hosts and /etc/ethers files on the master yp server must contain this information.
- □ Make sure that each client mounts a user file system of its own architecture from the server. An MC68010 client uses the server's /usr.MC68010 file. An MC68020 client uses the server's /usr.MC68020 file. Make sure that the /etc/fstab file for each client partition contains the correct information.

Heterogeneous Server with Remote Tape Drive

- □ This configuration is a heterogeneous server running 3.2 or 3.3.
- □ The release tapes must be appropriate for both architectures and suitable for the remote tape drive. For example, you must have suitable Sun-2 and Sun-3 tapes, and tapes must both be 1/4 or 1/2 inch. You cannot mix media types.

□ Make sure that `/etc/hosts` and `/etc/ethers` on the heterogeneous server contain correct Internet and Ethernet addresses of all clients supported by the server if it is not running `yp`. If it is running `yp`, the `/etc/hosts` and `/etc/ethers` files on the master `yp` server must contain this information.

□ The tape host must be reachable from the heterogeneous server through the Ethernet. The tape host and the heterogeneous server must be on the same network.

□ `/etc/hosts` on the heterogeneous server must contain the Internet address of the tape host, if `yp` is not running on that server. If `yp` is running, the `/etc/hosts` file on the master `yp` server must contain the Internet address of the standalone system.

□ `/etc/hosts` on the tape host must contain the Internet address of the heterogeneous server if `yp` is not running. If `yp` is running, the `/etc/hosts` file on the master `yp` server must contain the Internet address of the standalone system.

□ `/.rhosts` on the tape host must contain the name of the heterogeneous server.

□ Make sure that each client mounts a user file system of its own architecture from the server. An MC68010 client uses the server's `/usr.MC68010` file. An MC68020 client uses the server's `/usr.MC68020` file. Make sure that the `/etc/fstab` file for each client partition contains the correct information.

Note that when you upgrade heterogeneous systems, the server will try to upgrade its own architecture first, then the other serving architectures. For example, if the heterogeneous server is a Sun-2 serving both Sun-2 and Sun-3 clients, the Sun-2 clients are updated before the Sun-3 clients.

2.3. Upgrade Installation

Before you begin the upgrade, you should make a number of preparations. Because Release 3.4 does not provide uninstall capabilities, it is strongly advised that you do a full backup of your system *before* upgrading to this release. Thus, you will be able to reinstall a prior version of the operating system in case you need to delete 3.4. Note that unlike Release 3.2, you DO NOT have to save any system-specific files in this backup.

If you plan to install either the new SunPro utilities or Sunview profiled libraries optional software, you should check available disk space on your system before upgrading. SunPro takes up 1 Mbyte of disk space. The Sunview profiled libraries take up 1.5 Mbytes of space and are installed in the `/usr` partition. If you want these libraries, make sure that `/usr` has enough disk space to accommodate them before beginning the upgrade.

After making these preparations, follow the next procedures to perform the upgrade.

1. Become superuser and halt your system. If the system is a server, make sure that all clients are halted before you halt the system.

```
host# /etc/halt
```

2. Bring the system up in single user mode as follows:

```
>b disk(0,0,0)vmunix -s
```

where *disk* is one of the devices listed in Table 2-1 below.

Table 2-3 *Disk Devices*

Devices	*Description*
xy	Xylogics 450/451 SMD disk controller
sd	SCSI disk controller

3. Mount the 3.4 release tape onto the tape drive, and make sure the tape drive is online and ready.

4. When the system is up and displays a # prompt, extract the upgrade utilities from the release tape.

   If you have a local tape drive, type the following:

```
# cd /etc
# mt -f /dev/nrtape0 rew
# mt -f /dev/nrtape0 fsf 2
# tar xvpf /dev/nrtape0
```

where *tape* is one of the values listed under *Dev Type* in Table 2-2 below.

Table 2-4 *Tape Devices*

Dev Type	*Tape Type*	*Description*
ar	*ar*	Archive quarter-inch tape cartridge
mt	*mt*	Nine-track magnetic 1/2" tape-Tapemaster controller
st	*st*	SCSI 1/4" tape cartridge-Tape controller
mt	*xt*	Nine-track magnetic 1/2" tape-Xylogics 472 controller

If you have a standalone system with a remote tape drive, type the following:

```
# cd /etc
# mount /usr
# rsh tapehost mt -f /dev/nrtape0 rew
# rsh tapehost mt -f /dev/nrtape0 fsf 2
# rsh tapehost -n dd if=/dev/nrtape0 bs=blocksizeb | tar xvpfB -
```

where *tapehost* is the name of the system with the tape drive and *blocksize* is dependent on the type of tape drive you are using. If you are using a 1/2-inch tape drive, the block size should be 20. If you are using a 1/4-inch tape drive, the block size should be 126.

If you have a homogeneous or heterogeneous server with a remote tape drive, type the following:

```
# cd /etc
# mount /usr.architecture
# rsh tapehost mt -f /dev/nrtape0 rew
# rsh tapehost mt -f /dev/nrtape0 fsf 2
# rsh tapehost -n dd if=/dev/nrtape0 bs=blocksizeb | tar xvpfB -
```

where *architecture* is the architecture type of the server, such as MC68010 or MC68020; *tapehost* is the name of the system with the tape drive; and *blocksize* is dependent on the type of tape drive you are using. If you are using a 1/2 inch tape drive, the block size should be 20. If you are using a 1/4 inch tape drive, the block size should be 126.

5. Start the upgrade script.

```
# 3.4UPGRADE
```

6. Specify the type of the system.

```
Enter system type ? [standalone | server]:
```

If the system is a server, you need to specify whether it is a homogeneous or heterogeneous server.

```
Enter server type ? [homo | heter]:
```

7. Specify whether the upgrade will be done with a local or remote tape drive. If you are upgrading the system with a remote tape drive, make sure that you meet all the requirements specified in the last section.

```
Enter tape drive type ? [local | remote]:
```

If you are upgrading the system with a remote tape drive, you also need to specify the name of the tape host and the Ethernet type of your system, as shown in Table 2-3.

Table 2-5 *Ethernet Types*

Type	*Description*
ec	3COM Ethernet controller
ie	Sun-2, Sun-3/100* Series, Sun-3/200 Series Ethernet controller
le	Sun-3/50 Series Ethernet controller

* The Sun 3/75 is considered a member of the Sun 3/100 Series.

```
Enter host of remote drive ?

Enter ethernet type of this system ? [ec | ie | le] :
```

8. Specify the tape type. (Refer to Table 2-2, under *Tape Type*.)

```
Enter tape type ? [ar | st | mt | xt]:
```

9. The upgrade procedure is about to begin. If you entered information incorrectly, you can answer **n** and restart by going back to Step 5. If you are ready to start the upgrade, answer **y**. The screen will prompt for your attention only when you need to change the tape.

```
    Are you ready to start the upgrade ? [y/n] :
```

The upgrade procedure begins here.

10. Specify whether or not you want to install the new SunPro optional software. (Remember that SunPro takes up one Mbyte of additional disk space.)

```
Do you want to install the new make (SunPro) for architecture ? [y/n]
```

where *architecture* is the architecture type of the standalone system or server (MC68010 or MC68020). For a heterogeneous server, just before upgrading the second architecture type, you will receive this prompt again, except *architecture* will be the second architecture type.

If you choose to install SunPro, finish the 3.4 upgrade instructions in this chapter and Chapter 3. Then refer to Appendix A, ***Installing the SunPro Optional Software***, for SunPro installation instructions.

11. Specify whether or not you want to install the new Sunview profiled libraries. (Remember, these libraries take up 1.5 Mbytes of disk space in the `/usr` partition.)

```
Do you want to install the window profiled libraries for
  architecture ? [y/n]
```

where *architecture* is the architecture type of the standalone system or server (MC68010 or MC68020). For a heterogeneous server, just before upgrading the second architecture type, you will receive this prompt again, except *architecture* will be the second architecture type.

12. If you are using 1/4 inch tapes, you will receive the following prompt during the upgrade procedure:

```
 Load release tape #2 for architecure
```

where *architecture* is either MC68010 for Sun-2s or MC68020 for Sun-3s. You will only receive this prompt if software on the second tape is needed to complete the upgrade.

Below are the prompts that you see during the upgrade procedure. This particular procedure is for a Sun-2 called *niobium*. It is a standalone system that was running 3.2 before the upgrade, uses 1/2 inch tape, and has all optional software on the disk.

```
    Beginning 3.2 to 3.4 upgrade for the MC68010 architecture.

    Changing directory to "/".

    Extracting "root" files from "/dev/nrmt0" release tape.

    Changing directory to "/".

    Extracting "pub" files from "/dev/nrmt0" release tape.

    Extracting "client" files from "/dev/nrmt0" release tape.

    Changing directory to "/usr".

    Extracting "usr" files from "/dev/nrmt0" release tape.

    Extracting "sys" files from "/dev/nrmt0" release tape.
```

```
Extracting "Networking" files from "/dev/nrmt0" release tape.

Extracting "Debugging" files from "/dev/nrmt0" release tape.

Extracting "SunView_user" files from "/dev/nrmt0" release tape.

Extracting "SunView_prog" files from "/dev/nrmt0" release tape.

Extracting "SunView_src" files from "/dev/nrmt0" release tape.

Extracting "Text-processing" files from "/dev/nrmt0" release tape.

Extracting "Fortran" files from "/dev/nrmt0" release tape.

Extracting "Pascal" files from "/dev/nrmt0" release tape.

Extracting "Graphics" files from "/dev/nrmt0" release tape.

Extracting "profiled" files from "/dev/nrmt0" release tape.

Extracting "SysV" files from "/dev/nrmt0" release tape.

Extracting "Demo" files from "/dev/nrmt0" release tape.

Extracting "Man" files from "/dev/nrmt0" release tape.

Extracting "Games" files from "/dev/nrmt0" release tape.

Extracting "SunPro" files from "/dev/nrmt0" release tape.

Extracting "Window_profiled" files from "/dev/nrmt0" release tape.

/dev/rxy0g: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

3.2 to 3.4 upgrade completed.
Reboot your system and configure a kernel for your system.
#
```

13. Reboot the system.

```
# reboot
```

14. Reconfigure a kernel for your system. Refer to Chapter 3 for details.

2.4. Example Upgrade Walkthroughs

This section contains upgrade walkthrough examples for the following system configurations:

- Standalone System with a Local Tape Drive
- Standalone System with a Remote Tape Drive
- Homogeneous Server with a Local Tape Drive
- Homogeneous Server with a Remote Tape Drive
- Heterogeneous Server with a Local Tape Drive
- Heterogeneous Server with a Remote Tape Drive

Upgrade a Standalone System with a Local Tape Drive

Assume that the standalone system is a Sun-3 with a local 1/4 inch tape drive (st), a Xylogics SMD disk (xy), and that the system is running 3.2. Below is a list of the optional software that currently exists on the disk.

Debugging tools
Suntools_users
Suntools_source
Fortran
Pascal
Profiled
Man
Suntools_programmers

Enter the following:

```
>b xy(0,0,0)vmunix -s
     .
     .
     .
# cd /etc
```

Make sure that the 3.4 release tape is mounted on the tape drive. Then enter the following:

```
# mt -f /dev/nrst0 rew
# mt -f /dev/nrst0 fsf 2
# tar xvpf /dev/nrst0
# 3.4UPGRADE
```

The system responds as follows:

```
Enter system type ? [standalone | server]: standalone

Enter tape drive type ? [local | remote]: local

Enter tape type ? [ar | st | mt | xt]: st
```

```
/dev/rxy0g: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

Are you ready to start the upgrade ? [y/n] : y

Do you want to install the new (make) SunPro for MC68020? [y/n] : n

Do you want to install the new window profiled libraries for
   MC68020 ? [y/n] : n

Beginning 3.2 to 3.4 upgrade for the MC68020 architecture.

Changing directory to "/".

Extracting "root" files from "/dev/nrst0" release tape.

Changing directory to "/".

Extracting "pub" files from "/dev/nrst0" release tape.

Extracting "client" files from "/dev/nrst0" release tape.

Changing directory to "/usr".

Extracting "usr" files from "/dev/nrst0" release tape.

Extracting "sys" files from "/dev/nrst0" release tape.

Extracting "Debugging" files from "/dev/nrst0" release tape.

Extracting "Fortran" files from "/dev/nrst0" release tape.

Extracting "Pascal" files from "/dev/nrst0" release tape.

Extracting "profiled" files from "/dev/nrst0" release tape.

Load release tape #2 for MC68020:

Extracting "Man" files from /dev/nrst0" release tape.

/dev/rxy0g: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

3.2 to 3.4 upgrade completed.
Reboot your system and configure a kernel for your system.
#
```

Upgrade a Standalone System with Remote Tape Drive

Assume that you have a Sun-3 machine and it is running 3.2 with a remote st controller and xy disk. Below is a list of the optional software that currently exists on the disk.

Suntools_users
Suntools_source

Fortran
Pascal
Profiled
Man

Enter the following:

```
>b xy(0,0,0)vmunix -s
# cd /etc
# mount /usr
```

Mount the 3.4 release tape as follows:

```
# rsh pebbles mt -f /dev/nrst0 rew
# rsh pebbles mt -f /dev/nrst0 fsf 2
# rsh pebbles -n dd if=/dev/nrst0 bs=126b | tar xvpfB -
# 3.4UPGRADE
```

The system responds

```
Enter system type ? [standalone | server]: standalone

Enter tape drive type ? [local | remote]: remote

Enter host of remote drive ? pebbles

Enter ethernet type of this system ? [ec | ie | le] : ie

Enter tape type ? [ar | st | mt | xt]: st

/dev/rxy0g: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

Are you ready to start the upgrade ? [y/n]  : y

Do you want to install the new make (SunPro) for MC68020 ? [y/n] : n

Do you want to install the new window profiled libraries for
   MC68020? [y/n] : n

Beginning 3.2 to 3.4 upgrade for the MC68020 architecture.

Changing directory to "/".

Extracting "root" files from "/dev/nrst0" release tape.

Changing directory to "/".

Extracting "pub" files from "/dev/nrst0" release tape.

Extracting "client" files from "/dev/nrst0" release tape.
```

```
Changing directory to "/usr".

Extracting "usr" files from "/dev/nrst0" release tape.

Extracting "sys" files from "/dev/nrst0" release tape.

Extracting "Fortran" files from "/dev/nrst0" release tape.

Extracting "Pascal" files from "/dev/nrst0" release tape.

Extracting "profiled" files from "/dev/nrst0" release tape.

Load release tape #2 for MC68020:

Extracting "Man" files from "/dev/nrst0" release tape.

/dev/rxy0g: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

3.2 to 3.4 upgrade completed.
Reboot your system and configure a kernel for your system.
#
```

Upgrade a Homogeneous Server with Local Tape Drive

Assume that you have a Sun-3 server called *godzilla* with an xy disk controller and mt tape controller. Then assume *godzilla* has three Sun-3 clients : *frodo*, *grendel*, and *sofia*. In this case, all 3.2 optional software exists on the disk.

Enter the following:

```
>b xy(0,0,0)vmunix -s
# cd /etc
```

Make sure that the 3.4 release tape is mounted on the tape drive, then enter the following:

```
# mt -f /dev/nrmt0 rew
# mt -f /dev/nrmt0 fsf 2
# tar xvpf /dev/nrmt0
# 3.4UPGRADE
```

The system responds

```
Enter system type ? [standalone | server]: server

Enter server type ? [homo | heter]: homo

Enter tape drive type ? [local | remote]: local

Enter tape type ? [ar | st | mt | xt]: mt
```

```
/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

Are you ready to start the upgrade ? [y/n] : y

Do you want to install the new make (SunPro) for MC68020 ? [y/n] : y

Do you want to install the new window profiled libraries for
   MC68020 ? [y/n] : y

Beginning 3.2 to 3.4 upgrade for the MC68020 architecture.

Changing directory to "/".

Extracting "root" files from "/dev/nrmt0" release tape.

Changing directory to "/pub".

Extracting "pub" files from "/dev/nrmt0" release tape.

Beginning 3.2 to 3.4 upgrade on MC68020 diskless clients.

Beginning 3.2 to 3.4 upgrade on client frodo.

Extracting "client" files from "/dev/nrmt0" release tape.
                   .
                   .
Completed 3.2 to 3.4 upgrade on client frodo.

Beginning 3.2 to 3.4 upgrade on client grendel.

Extracting "client" files from "/dev/nrmt0" release tape.
                   .
                   .
Completed 3.2 to 3.4 upgrade on client grendel.

Beginning 3.2 to 3.4 upgrade on client sofia.

Extracting "client" files from "/dev/nrmt0" release tape.
                   .
                   .
Completed 3.2 to 3.4 upgrade on client sofia.

Changing directory to "/usr".

Extracting "usr" files from "/dev/nrmt0" release tape.

Extracting "sys" files from "/dev/nrmt0" release tape.

Extracting "Networking" files from "/dev/nrmt0" release tape.

Extracting "Debugging" files from "/dev/nrmt0" release tape.

Extracting "SunView_user" files from "/dev/nrmt0" release tape.
Extracting "SunView_prog" files from "/dev/nrmt0" release tape.

Extracting "SunView_src" files from "/dev/nrmt0" release tape.

Extracting "Text-processing" files from "/dev/nrmt0" release tape.

Extracting "Fortran" files from "/dev/nrmt0" release tape.

Extracting "Pascal" files from "/dev/nrmt0" release tape.

Extracting "Graphics" files from "/dev/nrmt0" release tape.

Extracting "profiled" files from "/dev/nrmt0" release tape.

Extracting "SysV" files from "/dev/nrmt0" release tape.

Extracting "Demo" files from "/dev/nrmt0" release tape.

Extracting "Man" files from "/dev/nrmt0" release tape.

Extracting "Games" files from "/dev/nrmt0" release tape.

Extracting "SunPro" files from "/dev/nrmt0" release tape.

Extracting "Window_profiled" files from "/dev/nrmt0" release tape.

/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

3.2 to 3.4 upgrade completed.
Reboot your system and configure a kernel for your system.
```

Upgrade a Homogeneous Server with Remote Tape Drive

Assume that you have a Sun-3 server with an xy disk controller and remote mt tape controller on client *pebbles*, and that the system is running 3.2. In this case, no optional software exists on the disk.

Enter the following:

```
>b xy(0,0,0)vmunix -s
# cd /etc
# mount /usr.MC68020
```

where MC68020 represents the system's architecture. Make sure that the 3.4 release tape is mounted on the tape drive, then enter the following:

```
# rsh pebbles mt -f /dev/nrmt0 rew
# rsh pebbles mt -f /dev/nrmt0 fsf 2
# rsh pebbles -n dd if=/dev/nrmt0 bs=20b | tar xvpfB -
# 3.4UPGRADE
```

The system responds as follows:

```
Enter system type ? [standalone | server]: server

Enter server type ? [homo | heter]: homo

Enter tape drive type ? [local | remote]: remote

Enter host of remote drive ? pebbles

Enter ethernet type of this system ? [ec | ie | le] : ie

Enter tape type ? [ar | st | mt | xt]: mt

/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

Are you ready to start the upgrade ? [y/n] : y

Do you want to install the new make (SunPro) for MC68020 ? [y/n] : n

Do you want to install the new window profiled libraries for
   MC68020 ? [y/n] : n

Beginning 3.2 to 3.4 upgrade for the MC68020 architecture.

Changing directory to "/".

Extracting "root" files from "/dev/nrmt0" release tape.

Changing directory to "/pub."

Extracting "pub" files from "/dev/nrmt0" release tape.

Beginning 3.2 to 3.4 upgrade on MC68020 diskless clients.

Beginning 3.2 to 3.4 upgrade on client frodo.

Extracting "client" files from "/dev/nrmt0" release tape.
                .
                .
Completed 3.2 to 3.4 upgrade on client frodo.

Beginning 3.2 to 3.4 upgrade on client grendel.

Extracting "client" files from "/dev/nrmt0" release tape.
                .
                .
```

```
Completed 3.2 to 3.4 upgrade on client grendel.

Beginning 3.2 to 3.4 upgrade on client sofia.

Extracting "client" files from "/dev/nrmt0" release tape.
                 .
                 .
Completed 3.2 to 3.4 upgrade on client sofia.

Changing directory to "/usr".

Extracting "usr" files from "/dev/nrmt0" release tape.

Extracting "sys" files from "/dev/nrmt0" release tape.

/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)

3.2 to 3.4 upgrade completed.
Reboot your system and configure a kernel for your system.
#
```

Upgrade a Heterogeneous Server with Local Tape Drive

Assume that you have a Sun-3 server called *godzilla* with an xy disk controller and local mt tape controller. *Godzilla* has two Sun-3 clients, *frodo* and *grendel*, and one Sun-2 client, *sofia*. In this case, assume that only networking optional software from both architectures exists on the disk.

Enter the following:

```
>b xy(0,0,0)vmunix -s
# cd /etc
```

Make sure that the 3.4 release tape is mounted on the tape drive, then type the following:

```
# mt -f /dev/nrmt0 rew
# mt -f /dev/nrmt0 fsf 2
# tar xvpf /dev/nrmt0
# 3.4UPGRADE
```

The system responds as follows:

```
Enter system type ? [standalone | server]: server

Enter server type ? [homo | heter]: heter

Enter tape drive type ? [local | remote]: local

Enter tape type ? [ar | st | mt | xt]: mt
```

```
/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)
/dev/rxy0g: 1163 files, 18150 used, 40393 free (73 frags, 5040 blocks)
/dev/rxy0e: 83 files, 3574 used, 905 free (17 frags, 111 blocks)

Are you ready to start the upgrade ? [y/n] : y

Do you want to install the new make (SunPro) for MC68020 ? [y/n] : n

Do you want to install the new window profiled libraries for
   MC68020 ? [y/n] : n

Are you ready to start the upgrade ? [y/n] : y

Beginning 3.2 to 3.4 upgrade for the MC68020 architecture.

Changing directory to "/".

Extracting "root" files from "/dev/nrmt0" release tape.

Changing directory to "/pub."

Extracting "pub" files from "/dev/nrmt0" release tape.

Beginning 3.2 to 3.4 upgrade on MC68020 diskless clients.

Beginning 3.2 to 3.4 upgrade on client frodo.

Extracting "client" files from "/dev/nrmt0" release tape.
                  .
                  .
Completed 3.2 to 3.4 upgrade on client frodo.

Beginning 3.2 to 3.4 upgrade on client grendel.

Extracting "client" files from "/dev/nrmt0" release tape.
                  .
                  .
Completed 3.2 to 3.4 upgrade on client grendel.

Changing directory to "/usr".

Extracting "usr" files from "/dev/nrmt0" release tape.

Extracting "sys" files from "/dev/nrmt0" release tape.

Extracting "Networking" files from "/dev/nrmt0" release tape.

Do you want to install the new make (SunPro) for MC68010 ? [y/n] : n

Do you want to install the new window profiled libraries for
   MC68010 ? [y/n] : n

Beginning 3.2 to 3.4 upgrade for the MC68010 architecture.
```

```
Changing directory to "/pub.MC68010".

Load release tape #1 for MC68010:

Extracting "pub" files from "/dev/nrmt0" release tape.

Beginning 3.2 to 3.4 upgrade on MC68010 diskless clients.

Beginning 3.2 to 3.4 upgrade on client sofia.

Extracting "client" files from "/dev/nrmt0" release tape.
                .
                .
Completed 3.2 to 3.4 upgrade on client sofia.

Changing directory to "/usr.MC68010".

Extracting "usr" files from "/dev/nrmt0" release tape.

Extracting "sys" files from "/dev/nrmt0" release tape.

Extracting "Networking" files from "/dev/nrmt0" release tape.

/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)
/dev/rxy0g: 1163 files, 18150 used, 40393 free (73 frags, 5040 blocks)
/dev/rxy0e: 83 files, 3574 used, 905 free (17 frags, 111 blocks)

3.2 to 3.4 upgrade completed.
Reboot your system and configure a kernel for it.
#
```

Upgrade a Heterogeneous Server with Remote Tape Drive

Assume that you have a Sun-3 server with an xy disk controller, and that you will use the remote tape drive with st controller on the client *pebbles*. Also assume that only networking optional software from both architectures exists on the disk. (Networking software is on the first tape, so you will not receive a message asking you to load the second tape.)

Enter the following:

```
>b xy(0,0,0)vmunix -s
# cd /etc
# mount /usr.MC68020
```

Make sure that the tape is mounted, then enter the following:

```
# rsh pebbles mt -f /dev/nrst0 rew
# rsh pebbles mt -f /dev/nrst0 fsf 2
# rsh pebbles -n dd if=/dev/nrst0 bs=126b | tar xvpfB -
# 3.4UPGRADE
```

The system responds as follows:

```
Enter system type ? [standalone | server]: server

Enter server type ? [homo | heter]: heter

Enter tape drive type ? [local | remote]: remote

Enter host of remote drive ? pebbles

Enter ethernet type of this system ? [ec | ie | le] : ie

Enter tape type ? [ar | st | mt | xt]: st

/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)
/dev/rxy0g: 1163 files, 18150 used, 40393 free (73 frags, 5040 blocks)
/dev/rxy0e: 83 files, 3574 used, 905 free (17 frags, 111 blocks)

Are you ready to start the upgrade ? [y/n] : y

Do you want to install the new make (SunPro) for MC68020 ? [y/n] n

Do you want to install the new window profiled libraries for
   MC68020 ? [y/n] n

Are you ready to start the upgrade ? [y/n] : y

Beginning 3.2 to 3.4 upgrade for the MC68020 architecture.

Changing directory to "/".

Extracting "root" files from "/dev/nrst0" release tape.

Changing directory to "/pub."

Extracting "pub" files from "/dev/nrst0" release tape.

Beginning 3.2 to 3.4 upgrade on MC68020 diskless clients.

Beginning 3.2 to 3.4 upgrade on client frodo.

Extracting "client" files from "/dev/nrst0" release tape.
                .

                .
Completed 3.2 to 3.4 upgrade on client frodo.

Beginning 3.2 to 3.4 upgrade on client grendel.
```

```
Extracting "client" files from "/dev/nrst0" release tape.
                 .
                 .
Completed 3.2 to 3.4 upgrade on client grendel.

Changing directory to "/usr".

Extracting "usr" files from "/dev/nrst0" release tape.

Extracting "sys" files from "/dev/nrst0" release tape.

Do you want to install the new make (SunPro) for MC68010 ? [y/n] n

Do you want to install the new window profiled libraries for
   MC68010 ? [y/n] n

Beginning 3.2 to 3.4 upgrade for the MC68010 architecture.
                 .
                 .
Load release tape #1 for MC68010:

Changing directory to "/pub.MC68010".

Extracting "pub" files from "/dev/nrst0" release tape.

Beginning 3.2 to 3.4 upgrade on MC68010 diskless clients.

Beginning 3.2 to 3.4 upgrade on client sofia.

Extracting "client" files from "/dev/nrmt0" release tape.
                 .
                 .
Completed 3.2 to 3.4 upgrade on client sofia.

Changing directory to "/usr.MC68010".

Extracting "usr" files from "/dev/nrst0" release tape.

Extracting "sys" files from "/dev/nrst0" release tape.

/dev/rxy0h: 1678 files, 18810 used, 39733 free (69 frags, 4958 blocks)
/dev/rxy0g: 1163 files, 18150 used, 40393 free (73 frags, 5040 blocks)
/dev/rxy0e: 83 files, 3574 used, 905 free (17 frags, 111 blocks)

3.2 to 3.4 upgrade completed.
Reboot your system and configure a kernel for your system.
#
```

2.5. How to Upgrade a Client Partition to 3.4

This section explains how you can upgrade to Release 3.4 a partition that was not upgraded when the server was. Here is a sample `/etc/nd.local` file.

```
#
# These lines added by the Sun Setup Program
#
clear
version 1
user 0 1 /dev/xy0f 0 11960 -1
user client1 0 /dev/xy0c 50600 10120 0
user client1 1 /dev/xy0c 60720 39560 -1
#user client2 0 /dev/xy0c 100280 10120 1
#user client2 1 /dev/xy0c 110400 39560 -1
user client3 0 /dev/xy0c 149960 10120 2
user client3 1 /dev/xy0c 160080 39560 -1
son
#
# End of lines added by the Sun Setup Program
#
```

The client named "client2" was commented out during the upgrade and must be upgraded manually.

Procedures for Upgrading Client2 to 3.4

Follow these steps to perform the upgrade:

1. Make sure that `/etc/hosts` on the server contains the Internet address of client2. You need to make sure that `/etc/hosts` on the `yp` master contains the Internet address of client2 if you are running yellow pages.

2. Make sure that `/etc/ethers` on the server contains the Ethernet address of client2. You need to make sure that `/etc/ethers` on the `yp` master contains the Ethernet address of client2 if you are running yellow pages.

3. Fix `/etc/nd.local` by removing # signs from the following two lines:

```
#user client2 0 /dev/xy0c 100280 10120 1
#user client2 1 /dev/xy0c 110400 39560 -1
```

(Refer to `nd`(8) in the *Commands Reference Manual* for more information.)

4. Run `nd` as follows:

```
server# /etc/nd < /etc/nd.local
```

5. Mount the client partition.

```
server# mount /dev/ndl1 /mnt
```

6. Go to the client partition.

```
server# cd /mnt
```

7. Mount the 3.4 release tape on the tape drive.

8. Position to File 6 on the release tape.

```
server# mt -f /dev/nrtape0 rew
server# mt -f /dev/nrtape0 fsf 5
```

Refer to Table 2-2 for tape devices.

9. Extract the client files from the tape.

```
server# tar xvpf /dev/nrtape0
server# mt -f /dev/nrtape0 rew
```

10. Leave the client partition.

```
server# cd /
```

11. Unmount the client partition.

```
server# umount /dev/ndl1
```

2.6. How to Restart the Upgrade Procedures

If for any reason `UPGRADE` terminates before completion, you can restart the upgrade procedure by doing the following:

1. Make sure that you are in single user mode, with your regular root file system (not `miniroot`) mounted.

2. If the system is a server, make sure the `pub` file system is mounted.

3. Restart the upgrade script by entering the following:

```
# cd /etc
# 3.4UPGRADE
```

3

Reconfiguring Your Kernel

3

Reconfiguring Your Kernel

This chapter explains how to reconfigure the kernel. You do not have to reconfigure your kernel because a GENERIC kernel is now provided on the upgrade tape. However, it is recommended that you reconfigure in order to save space, customize the kernel to recognize your hardware, or change MAXUSERS to a larger size to accommodate Suntools. You must reboot your system upon completion of the 3.4 upgrade, whether or not you reconfigure.

If you are doing kernel configuration for the first time, you can use the procedures in

> *Installing UNIX on the Sun Workstation* for Release 3.2
> (Part Number: 800-1521)

Note: See "General and Specific System Description Lines" sections of *Installing Unix on the Sun Workstation* for more detailed information concerning the annotated GENERIC file.

If you have previously configured a kernel, you can use the next sections to guide you through reconfiguration. The first section gives reconfiguration procedures for all architectures.. The second section contains an annotated copy of the new *GENERIC* kernel configuration file; read it carefully to make sure that you are including the correct device description lines for your system. The last two sections of this chapter contain specific *GENERIC* files for a Sun-2 or Sun-3 machine.

3.1. Procedures for Reconfiguring the Kernel

This section contains kernel reconfiguration procedures for all architecture types running Release 3.4.

Kernel Reconfiguration for Standalone Systems

For standalone machines, proceed as follows.

1. Change the current directory to `/usr/sys/conf`.

```
# cd /usr/sys/conf
```

2. Create a kernel configuration file.

   Copy the file *GENERIC* and comment out the lines that do not apply to your system. In the next examples, the new file is called *SYS_NAME* (representing the name of the system).

```
# cp GENERIC SYS_NAME
# chmod +w SYS_NAME
```

3. Edit /usr/sys/conf/*SYS_NAME* to reflect your system configuration. Use the annotated copy of *GENERIC* provided in the following section for an explanation of these changes. Make sure that you are including the proper device description lines for your system.

4. Still in the /usr/sys/conf directory, run /etc/config. Then change to the new configuration directory, as shown below, and make the new system. (Remember to substitute your actual system image name for *SYS_NAME*.)

```
# /etc/config SYS_NAME
# cd ../SYS_NAME
# make
[ lots of output ]
```

5. Now you can save your old kernel and install the new one as follows:

```
# mv /vmunix /vmunix.old
# cp vmunix /vmunix
# /etc/shutdown -h now
     The system goes through the halt sequence, then
     the monitor displays its prompt, at which point you
     can boot the system:
>b vmunix
```

6. If the system appears to work, this completes the upgrade procedure. If the new kernel does not seem to function properly, boot /vmunix.old, as shown below, copy it back to /vmunix, and go about fixing your new kernel.

Note: It is very useful to save the old kernel until the new one is working smoothly, and in case you need it for diagnostic purposes.

```
# /etc/shutdown -h now
>b vmunix.old -s
# mv /vmunix /vmunix.oops
# mv /vmunix.old /vmunix
# ^D       [ Brings the system up multiuser ]
```

Kernel Reconfiguration for Servers

For server machines, proceed as follows.

1. Change the current directory to /usr/sys/conf.

```
# cd /usr/sys/conf
```

2. Create a kernel configuration file.

   Copy the file *GENERIC* and comment out the lines that do not apply to your system. The new file is called *SYS_NAME* (representing the name of the system). For example,

```
# cp GENERIC SYS_NAME
# chmod +w SYS_NAME
```

3. Edit `/usr/sys/conf/`*SYS_NAME* to reflect your system configuration. Use the annotated copy of *GENERIC* provided in the next section for an explanation of these changes. Make sure that you include the proper device description lines for your system.

4. Still in the `/usr/sys/conf` directory, run `/etc/config`. Then, change to the new configuration directory, and make the new system. (Remember to substitute your actual system image name for *SYS_NAME* .)

```
# /etc/config SYS_NAME
# cd ../SYS_NAME
# make
[ lots of output ]
```

5. Now, prepare a kernel for your clients in the same way. When editing the configuration file (*CLIENT_KERNEL_NAME* in the following example), remember to include the entire set of devices used by client machines. For example, substitute 68010 (for Sun-2s) or 68020 (for Sun-3s) for *client_arch* in the commands below.

```
# cd /usr.MCclient_arch/sys/conf
# cp GENERIC CLIENT_KERNEL_NAME
# chmod +w CLIENT_KERNEL_NAME
[ Edit CLIENT_KERNEL_NAME to reflect all clients' systems.
  Be especially careful with the device description lines. ]
# /etc/config CLIENT_KERNEL_NAME
# cd ../CLIENT_KERNEL_NAME
# make
[ lots of output ]
```

6. Now you can go to the directory containing the server's kernel, save your server's old kernel, install your new one, and try everything out as follows:

```
# cd /usr/sys/SYS_NAME
# mv /vmunix /vmunix.old
# cp vmunix /vmunix
```

7. Next, install the appropriate client kernel in /pub for the architecture. To install the clients' kernel, **make sure all the clients are halted**; save the original kernel (if there is one); install the new kernel image in the appropriate /pub; then test it out by booting up one of the clients.

   ```
   # cd /usr.MCclient_arch/sys/CLIENT_KERNEL_NAME
   # cp vmunix /pub.client_arch/vmunix
   ```

   On the client machine, enter

   ```
   >b vmunix
   ```

8. Since at this point normal system performance is a highly, but not absolutely, certain indicator of a trouble-free kernel, if your system(s) appears to work, you may proceed with some confidence. You have successfully completed installation. Congratulations!

   If, on the other hand, any new kernels do not seem to function properly, halt all systems, and boot from the original kernel. Then, move the faulty kernel away, and reinstall the original in its place. Once you are booted up on the original, you can try to fix the faulty kernel. For example, try the following on the server:

   ```
   # /etc/halt
   >b vmunix.old -s
   # cd /
   # mv vmunix vmunix.bad
   # mv vmunix.old vmunix
   # ^D              [ Brings the system up multi-user ]
   ```

   For clients, halt all the clients on the server. You have to correct the problem from the server.

   Enter the following on the server:

   ```
   # cd /pub.MCclient_arch
   # mv vmunix vmunix.bad
   # mv vmunix.old vmunix
   ```

You may now boot up the clients and allow them to run while, or until, a new client kernel is made and ready to install. Or, if the clients can remain down, build and install a new client kernel now.

3.2. Kernel Reconfiguration — an Annotated Copy of a *GENERIC* File

Note: For the specific GENERIC files for a Sun-2 or Sun-3 machine, see Section 3.3 (Sun-2) or Section 3.4 (Sun-3) of this chapter.

The next page contains an *example* of an annotated copy of a *GENERIC* file to help you identify the lines you need to include in your own system configuration file.

The comments explain the device and pseudo-device lines, and may also refer you to the reference manual entry that covers the device in question. If the comments say the line is **mandatory**, the line *must* be included in every system configuration file, either exactly as it stands, or, if commentary indicates variables, with the variables adjusted to fit your system.

A number of parameters relating to the System V Inter-Process Communication (IPC) extensions may also be tuned in the configuration file. These parameters do not appear in the GENERIC file but are documented in the System V Overview.

Configuration Line	*Comments*	*Description*
`#`		
`# GENERIC SUN-3`		
`#`		
`machine "sun3"`	**mandatory**	Identifies the specific machine.
`cpu "SUN3_160"`	**mandatory***	Identifies CPU type (Sun-3/160, Sun-3/180, Sun-3/140, Sun-3/75).
`cpu "SUN3_50"`	**mandatory***	Identifies the specific CPU type.
`cpu "SUN3_260"`	**mandatory***	Identifies the specific CPU type (Sun-3/260 or Sun-3/280).
`cpu "SUN3_110"`	**mandatory***	Identifies the specific CPU type (Sun-3/110).
`ident GENERIC`	**mandatory**	See *General* and *Specific System Description Lines* for information. Finally, if *SYS_NAME* contains both alpha and numeric characters (for example, SDST120), you must enclose the name in double quotes ("SDST120"), or you will get a syntax error when you run `/etc/config`.
`timezone 8 dst`	**mandatory**	Specifies your time zone and adjusts values accordingly. It can also use half hour designations.
`maxusers 4`	**mandatory**	Number may vary. For most systems, "4" is the proper value for maxusers. See the section, *General System Description Lines*, for information.
`options INET`	**mandatory**	Controls inclusion of Internet code — see `inet`(4). You must also include the "pseudo-device loop" lines below.
`options SYSACCT`	*optional*	Controls inclusion of code to do process accounting — see `acct`(2) and `acct`(5).
`options QUOTA`	*optional*	Controls the disk quota checking system.
`options NFS`	*optional*	Inclusion of NFS code.
`options NIT`	*optional*	Inclusion of network interface tap code.
`options IPCMESSAGE`	*optional*	Controls inclusion of code for SystemV IPC Message Facility.
`options IPCSEMAPHORE`	*optional*	Controls inclusion of code for SystemV IPC Semaphore Facility.
`options IPCSHMEM`	*optional*	Controls inclusion of code for SystemV IPC Shared-Memory Facility.
`config vmunix swap generic`	**mandatory**	Specifies kernel name and configuration clauses. Please see *Specific System Description Lines* for information.
`pseudo-device pty`	*optional*	Pseudo-tty's. These are needed for network or window system.
`pseudo-device bk`	*optional*	Berknet line discipline for high speed tty input — see `bk`(4).
`pseudo-device ether`	*optional*	ARP code. Must include if using Ethernet — see `arp`(4).
`pseudo-device loop`	**mandatory**	Software loop back network device driver — see `lo`(4). Must include with 'options INET'.
`pseudo-device nd`	*optional*	Network disk. Necessary for servers and diskless clients, and for machines serving as remote hosts for remote installation — see `nd`(4).
`pseudo-device win128`	*optional*	Window system. Number indicates maximum number of windows. If you include this line, you must also include the "pseudo-device dtop," "ms," and "kb" lines just below.
`pseudo-device dtop4`	*optional*	Maximum number of screens ("desktops"). Required for window system.
`pseudo-device ms3`	*optional*	Maximum number of mice. Required for window system — see `ms`(4).
`pseudo-device kb3`	*optional*	Maximum number of Sun keyboards. Required if using any Sun keyboard, and for the window system.

*** *You need not include all machine types, only the machine type(s) that you may be running.***

The following are connections for machine types. These connections, in conjunction with controllers, devices, and disks for a structure, enable your system to recognize various hardware and software attached to it. For each device or controller on a bus, you need to have the bus type it is connected to listed under connections for machine type. It easiest to leave all lines for machine types that way. As you add controllers and devices, the connections are already in place and will be recognized by your system.

```
# connections for machine type 1 (SUN3_160)
 controller      virtual 1 at nexus ?
 controller      obmem 1 at nexus ?
 controller      obio 1 at nexus ?
 controller      vme16d16 1 at nexus ?
 controller      vme24d16 1 at nexus ?
 controller      vme32d16 1 at nexus ?
 controller      vme16d32 1 at nexus ?
 controller      vme24d32 1 at nexus ?
 controller      vme32d32 1 at nexus ?
```

```
# connections for machine type 2 (SUN3_50)
 controller      virtual 2 at nexus ?
 controller      obmem 2 at nexus ?
 controller      obio 2 at nexus ?
```

```
# connections for machine type 3 (SUN3_260)
 controller      virtual 3 at nexus ?
 controller      obmem 3 at nexus ?
 controller      obio 3 at nexus ?
 controller      vme16d16 3 at nexus ?
 controller      vme24d16 3 at nexus ?
 controller      vme32d16 3 at nexus ?
 controller      vme16d32 3 at nexus ?
 controller      vme24d32 3 at nexus ?
 controller      vme32d32 3 at nexus ?
```

```
# connections for machine type 4 (SUN3_110)
 controller      virtual 4 at nexus ?
 controller      obmem 4 at nexus ?
 controller      obio 4 at nexus ?
 controller      vme16d16 4 at nexus ?
 controller      vme24d16 4 at nexus ?
 controller      vme32d16 4 at nexus ?
 controller      vme16d32 4 at nexus ?
 controller      vme24d32 4 at nexus ?
 controller      vme32d32 4 at nexus ?
```

The following are controllers and devices (devices, disks, and tapes) that connect to bus types. Bus types and devices must hang off the appropriate controller, which, in turn, hangs off another controller until a configuration is formed that gets you to a bus type that hangs off a nexus . On Sun workstations, all bus types are considered to hang off a nexus. For example, disk

```
    xy0 at xyc0 drive 0
```

hangs off controller

```
    xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x
```

which hangs off bus type

```
    controller      vme16d16 1 at nexus ?
```

In order to determine and note what devices are present on your machine, boot the GENERIC kernel after you have executed the 3.4UPGRADE. If you want, you can delete those lines that pertain to devices not on your machine. Or, you can configure your file with devices on other machines that may want to boot from the same kernel.

Do not remove the zs lines, which represent UARTS, from the configuration file. If they are removed, the system does not recognize the presence of the keyboard, mouse, or serial ports.

The following is an example of controllers and devices you find in a Sun-3 configuration file.

```
controller      xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller      xyc1 at vme16d16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk            xy0 at xyc0 drive 0
disk            xy1 at xyc0 drive 1
disk            xy2 at xyc1 drive 0
disk            xy3 at xyc1 drive 1
controller      sc0 at vme24d16 ? csr 0x200000 priority 2 vector scintr 0x40
disk            sd0 at sc0 drive 0 flags 0
disk            sd1 at sc0 drive 1 flags 0
tape            st0 at sc0 drive 32 flags 1
disk            sd2 at sc0 drive 8 flags 0
tape            st1 at sc0 drive 40 flags 1
#disk           sf0 at sc0 drive 8 flags 2
controller      si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
controller      si0 at obio ? csr 0x140000 priority 2
disk            sd0 at si0 drive 0 flags 0
disk            sd1 at si0 drive 1 flags 0
tape            st0 at si0 drive 32 flags 1
disk            sd2 at si0 drive 8 flags 0
tape            st1 at si0 drive 40 flags 1
#disk           sf0 at si0 drive 8 flags 2
device          zs0 at obio ? csr 0x20000 flags 3 priority 3
device          zs1 at obio ? csr 0x00000 flags 0x103 priority 3
device          mti0 at vme16d16 ? csr 0x620 flags 0xffff priority 4
                vector mtiintr 0x88
device          mti1 at vme16d16 ? csr 0x640 flags 0xffff priority 4
                vector mtiintr 0x89
```

```
device          mti2 at vme16d16 ? csr 0x660 flags 0xffff priority 4
                vector mtiintr 0x8a
device          mti3 at vme16d16 ? csr 0x680 flags 0xffff priority 4
                vector mtiintr 0x8b
device          ie0 at obio ? csr 0xc0000 priority 3
device          ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75
device          le0 at obio ? csr 0x120000 priority 3
controller      tm0 at vme16d16 ? csr 0xa0 priority 3 vector tmintr 0x60
controller      tm1 at vme16d16 ? csr 0xa2 priority 3 vector tmintr 0x61
tape            mt0 at tm0 drive 0 flags 1
tape            mt1 at tm1 drive 0 flags 1
controller      xtc0 at vme16d16 ? csr 0xee60 priority 3 vector xtintr 0x64
controller      xtc1 at vme16d16 ? csr 0xee68 priority 3 vector xtintr 0x65
tape            xt0 at xtc0 drive 0 flags 1
tape            xt1 at xtc1 drive 0 flags 1
device          gpone0 at vme24d16 ? csr 0x210000
device          cgtwo0 at vme24d16 ? csr 0x400000
device          cgfour0 at obmem 4 csr 0xff000000 priority 4
device          bwtwo0 at obmem 1 csr 0xff000000 priority 4
device          bwtwo0 at obmem 2 csr 0x100000 priority 4
device          bwtwo0 at obmem 3 csr 0xff000000 priority 4
device          bwtwo0 at obmem 4 csr 0xff000000
device          vpc0 at vme16d16 ? csr 0x480 priority 2 vector vpcintr 0x80
device          vpc1 at vme16d16 ? csr 0x500 priority 2 vector vpcintr 0x81
device          des0 at obio ? csr 0x1c0000
device          fpa0 at virtual ? csr 0xe0000000
```

3.3. Sun-2 GENERIC Configuration File

The following is the GENERIC configuration file for a Sun-2 system.

```
#
# GENERIC SUN2
#
machine           "sun2"
cpu               "SUN2_120"                generic for machine type 1 (Multibus)
cpu               "SUN2_50"                 generic for machine type 2 (VMEbus)
ident             GENERIC
timezone          8 dst
maxusers          4
options           INET
options           SYSACCT
options           QUOTA
options           NFS
options           NIT
options           IPCMESSAGE                SystemV IPC Message Facility
options           IPCSEMAPHORE              SystemV IPC Semaphore Facility
options           IPCSHMEM                  SystemV IPC Shared-Memory Facility

config            vmunix                    swap generic

pseudo-device     pty
pseudo-device     bk
```

```
pseudo-device     ether
pseudo-device     loop
pseudo-device     nd
pseudo-device     win128
pseudo-device     dtop4
pseudo-device     ms3
pseudo-device     kb3

# connections for machine type 1 (SUN2_120)
controller        virtual 1 at nexus ?    # virtual preset
controller        obmem 1 at nexus ?      # on board memory
controller        obio 1 at nexus ?       # on board io
controller        mbmem 1 at nexus ?      # Multibus memory
controller        mbio 1 at nexus ?       # Multibus io

# connections for machine type 2 (SUN2_50)
controller        virtual 2 at nexus ?    # virtual preset
controller        obmem 2 at nexus ?      # on board memory
controller        obio 2 at nexus ?       # on board io
controller        vme16 2 at nexus ?      # 16 bit address VMEbus (16 bit data)
controller        vme24 2 at nexus ?      # 24 bit address VMEbus (16 bit data)

controller        ipc0 at mbio ? csr 0x40 priority 2
controller        ipc1 at mbio ? csr 0x44 priority 2
disk              ip0 at ipc0 drive 0
disk              ip1 at ipc0 drive 1
disk              ip2 at ipc1 drive 0
disk              ip3 at ipc1 drive 1
controller        xyc0 at mbio ? csr 0xee40 priority 2
controller        xyc0 at vme16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller        xyc1 at mbio ?  csr 0xee48 priority 2
controller        xyc1 at vme16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk              xy0 at xyc0 drive 0
disk              xy1 at xyc0 drive 1
disk              xy2 at xyc1 drive 0
disk              xy3 at xyc1 drive 1
controller        sc0 at mbmem ? csr 0x80000 priority 2
controller        sc0 at vme24 ? csr 0x200000 priority 2 vector scintr 0x40
disk              sd0 at sc0 drive 0 flags 0
disk              sd1 at sc0 drive 1 flags 0
tape              st0 at sc0 drive 32 flags 1
disk              sd2 at sc0 drive 8 flags 0
tape              st1 at sc0 drive 40 flags 1
#disk             sf0 at sc0 drive 8 flags 2
controller        sc1 at mbmem ? csr 0x84000 priority 2
disk              sd2 at sc1 drive 0 flags 0
disk              sd3 at sc1 drive 1 flags 0
tape              st1 at sc1 drive 32 flags 1
#disk             sf1 at sc1 drive 8 flags 2
device            sky0 at mbio ? csr 0x2000 priority 2
device            sky0 at vme16 ? csr 0x8000 priority 2 vector skyintr 0xb0
device            zs0 at obio 1 csr 0x2000 flags 3 priority 3
device            zs0 at obio 2 csr 0x7f2000 flags 3 priority 3
device                  zs1 at obmem 1 csr 0x780000 flags 0x103 priority 3
device                  zs1 at obio 2 csr 0x7f1800 flags 0x103 priority 3
device                  zs2 at mbmem ? csr 0x80800 flags 3 priority 3
device                  zs3 at mbmem ? csr 0x81000 flags 3 priority 3
device                  zs4 at mbmem ? csr 0x84800 flags 3 priority 3
device                  zs5 at mbmem ? csr 0x85000 flags 3 priority 3
device                  mti0 at mbio ? csr 0x620 flags 0xffff priority 4
device                  mti1 at mbio ? csr 0x640 flags 0xffff priority 4
device                  mti2 at mbio ? csr 0x660 flags 0xffff priority 4
device                  mti3 at mbio ? csr 0x680 flags 0xffff priority 4
device                  mti0 at vme16 ? csr 0x620 flags 0xffff priority 4
                        vector mtiintr 0x88
device                  mti1 at vme16 ? csr 0x640 flags 0xffff priority 4
                        vector mtiintr 0x89
device                  mti2 at vme16 ? csr 0x660 flags 0xffff priority 4
                        vector mtiintr 0x8a
device                  mti3 at vme16 ? csr 0x680 flags 0xffff priority 4
                        vector mtiintr 0x8b
device                  ie0 at obio 2 csr 0x7f0800 priority 3
device                  ie0 at mbmem ? csr 0x88000 priority 3
device                  ie1 at mbmem ? csr 0x8c000 flags 2 priority 3
device                  ie1 at vme24 ? csr 0xe88000 priority 3 vector ieintr 0x75
device                  ec0 at mbmem ? csr 0xe0000 priority 3
device                  ec1 at mbmem ? csr 0xe2000 priority 3
controller              tm0 at mbio ? csr 0xa0 priority 3
controller              tm0 at vme16 ? csr 0xa0 priority 3 vector tmintr 0x60
controller              tm1 at mbio ? csr 0xa2 priority 3
controller              tm1 at vme16 ? csr 0xa2 priority 3 vector tmintr 0x61
tape                    mt0 at tm0 drive 0 flags 1
tape                    mt1 at tm1 drive 0 flags 1
controller              xtc0 at mbio ? csr 0xee60 priority 3
controller              xtc0 at vme16 ? csr 0xee60 priority 3 vector xtintr 0x64
controller              xtc1 at mbio ? csr 0xee68 priority 3
controller              xtc1 at vme16 ? csr 0xee68 priority 3 vector xtintr 0x65
tape                    xt0 at xtc0 drive 0 flags 1
tape                    xt1 at xtc1 drive 0 flags 1
device                  ar0 at mbio ? csr 0x200 priority 3
device                  ar1 at mbio ? csr 0x208 priority 3
device                  gpone0 at vme24 ? csr 0x210000 priority 4 vector gponeintr 0xa5
device                  cgtwo0 at vme24 ? csr 0x400000 priority 4 vector cgtwointr 0xa4
device                  cgone0 at mbmem ? csr 0xec000 priority 3
device                  bwtwo0 at obmem 1 csr 0x700000 priority 4
device                  bwtwo0 at obio 2 csr 0x0 priority 4
device                  bwone0 at mbmem ? csr 0xc0000 priority 3
device                  vp0 at mbio ? csr 0x400 priority 2
device                  vpc0 at mbio ? csr 0x480 priority 2
device                  vpc0 at vme16 ? csr 0x480 priority 2 vector vpcintr 0x80
device                  vpc1 at mbio ? csr 0x500 priority 2
device                  vpc1 at vme16 ? csr 0x500 priority 2 vector vpcintr 0x81
device                  pi0 at obio 1 csr 0x1800
device                  des0 at obio 1 csr 0x1000
device                  des0 at obio 2 csr 0x7f1000
device                  tod0 at obio 1 csr 0x3800
device            tod0 at vme24 ? csr 0x200800
```

3.4. Sun-3 GENERIC Configuration File

The following is the GENERIC configuration file for a Sun-3 system.

```
#
# GENERIC SUN3
#
machine           "sun3"
cpu               "SUN3_160"      # (Sun-3/160 or Sun-3/75 cpu)
cpu               "SUN3_50"
cpu               "SUN3_260"
cpu               "SUN3_110"
ident             GENERIC
timezone          8 dst
maxusers          4
options           INET
options           SYSACCT
options           QUOTA
options           NFS
options           NIT
options           IPCMESSAGE      # SystemV IPC Message Facility
options           IPCSEMAPHORE    # SystemV IPC Semaphore Facility
options           IPCSHMEM        # SystemV IPC Shared-Memory Facility

config            vmunix          swap generic

pseudo-device     pty
pseudo-device     bk
pseudo-device     ether
pseudo-device     loop
pseudo-device     nd
pseudo-device     win128
pseudo-device     dtop4
pseudo-device     ms3
pseudo-device     kb3

# connections for machine type 1 (SUN3_160)
controller        virtual 1 at nexus ?
controller        obmem 1 at nexus ?
controller        obio 1 at nexus ?
controller        vme16d16 1 at nexus ?
controller        vme24d16 1 at nexus ?
controller        vme32d16 1 at nexus ?
controller        vme16d32 1 at nexus ?
controller        vme24d32 1 at nexus ?
controller        vme32d32 1 at nexus ?

# connections for machine type 2 (SUN3_50)
controller        virtual 2 at nexus ?
controller        obmem 2 at nexus ?
controller        obio 2 at nexus ?
```

```
# connections for machine type 3 (SUN3_260)
controller        virtual 3 at nexus ?
controller        obmem 3 at nexus ?
controller        obio 3 at nexus ?
controller        vme16d16 3 at nexus ?
controller        vme24d16 3 at nexus ?
controller        vme32d16 3 at nexus ?
controller        vme16d32 3 at nexus ?
controller        vme24d32 3 at nexus ?
controller        vme32d32 3 at nexus ?

# connections for machine type 4 (SUN3_110)
controller        virtual 4 at nexus ?
controller        obmem 4 at nexus ?
controller        obio 4 at nexus ?
controller        vme16d16 4 at nexus ?
controller        vme24d16 4 at nexus ?
controller        vme32d16 4 at nexus ?
controller        vme16d32 4 at nexus ?
controller        vme24d32 4 at nexus ?
controller        vme32d32 4 at nexus ?

controller        xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller        xyc1 at vme16d16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk              xy0 at xyc0 drive 0
disk              xy1 at xyc0 drive 1
disk              xy2 at xyc1 drive 0
disk              xy3 at xyc1 drive 1
controller        sc0 at vme24d16 ? csr 0x200000 priority 2 vector scintr 0x40
disk              sd0 at sc0 drive 0 flags 0
disk              sd1 at sc0 drive 1 flags 0
tape              st0 at sc0 drive 32 flags 1
disk              sd2 at sc0 drive 8 flags 0
tape              st1 at sc0 drive 40 flags 1
#disk             sf0 at sc0 drive 8 flags 2
controller        si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
controller        si0 at obio ? csr 0x140000 priority 2
disk              sd0 at si0 drive 0 flags 0
disk              sd1 at si0 drive 1 flags 0
tape              st0 at si0 drive 32 flags 1
disk              sd2 at si0 drive 8 flags 0
tape              st1 at si0 drive 40 flags 1
#disk             sf0 at si0 drive 8 flags 2
device            zs0 at obio ? csr 0x20000 flags 3 priority 3
device            zs1 at obio ? csr 0x00000 flags 0x103 priority 3
device            mti0 at vme16d16 ? csr 0x620 flags 0xffff priority 4
                  vector mtiintr 0x88
device            mti1 at vme16d16 ? csr 0x640 flags 0xffff priority 4
                  vector mtiintr 0x89
device            mti2 at vme16d16 ? csr 0x660 flags 0xffff priority 4
                  vector mtiintr 0x8a
device            mti3 at vme16d16 ? csr 0x680 flags 0xffff priority 4
                  vector mtiintr 0x8b
```

```
device                  ie0 at obio ? csr 0xc0000 priority 3
device                  ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75
device                  le0 at obio ? csr 0x120000 priority 3
controller              tm0 at vme16d16 ? csr 0xa0 priority 3 vector tmintr 0x60
controller              tm1 at vme16d16 ? csr 0xa2 priority 3 vector tmintr 0x61
tape                    mt0 at tm0 drive 0 flags 1
tape                    mt1 at tm1 drive 0 flags 1
controller              xtc0 at vme16d16 ? csr 0xee60 priority 3 vector xtintr 0x64
controller              xtc1 at vme16d16 ? csr 0xee68 priority 3 vector xtintr 0x65
tape                    xt0 at xtc0 drive 0 flags 1
tape                    xt1 at xtc1 drive 0 flags 1
device                  gpone0 at vme24d16 ? csr 0x210000 priority 4 vector gponeintr 0xa5

device                  cgtwo0 at vme24d16 ? csr 0x400000 priority 4 vector cgtwointr 0xa4
device                  cgfour0 at obmem 4 csr 0xff000000 priority 4
device                  bwtwo0 at obmem 1 csr 0xff000000 priority 4
device                  bwtwo0 at obmem 2 csr 0x100000 priority 4
device                  bwtwo0 at obmem 3 csr 0xff000000 priority 4
device                  bwtwo0 at obmem 4 csr 0xff000000
device                  vpc0 at vme16d16 ? csr 0x480 priority 2 vector vpcintr 0x80
device                  vpc1 at vme16d16 ? csr 0x500 priority 2 vector vpcintr 0x81
device                  des0 at obio ? csr 0x1c0000
device                  fpa0 at virtual ? csr 0xe0000000
```

4

Changes to Release 3.2 in Release 3.3

4

Changes to Release 3.2 in Release 3.3

This chapter describes the changes and new features that were provided with the Sun System Release 3.3, including

- disconnect/reconnect
- subnets
- extensions to the Pascal compiler
- general bug fixes and documentation updates

4.1. disconnect/reconnect

The disconnect/reconnect facility available with the Sun System Release 3.3 can be used on Sun-3s with the following:

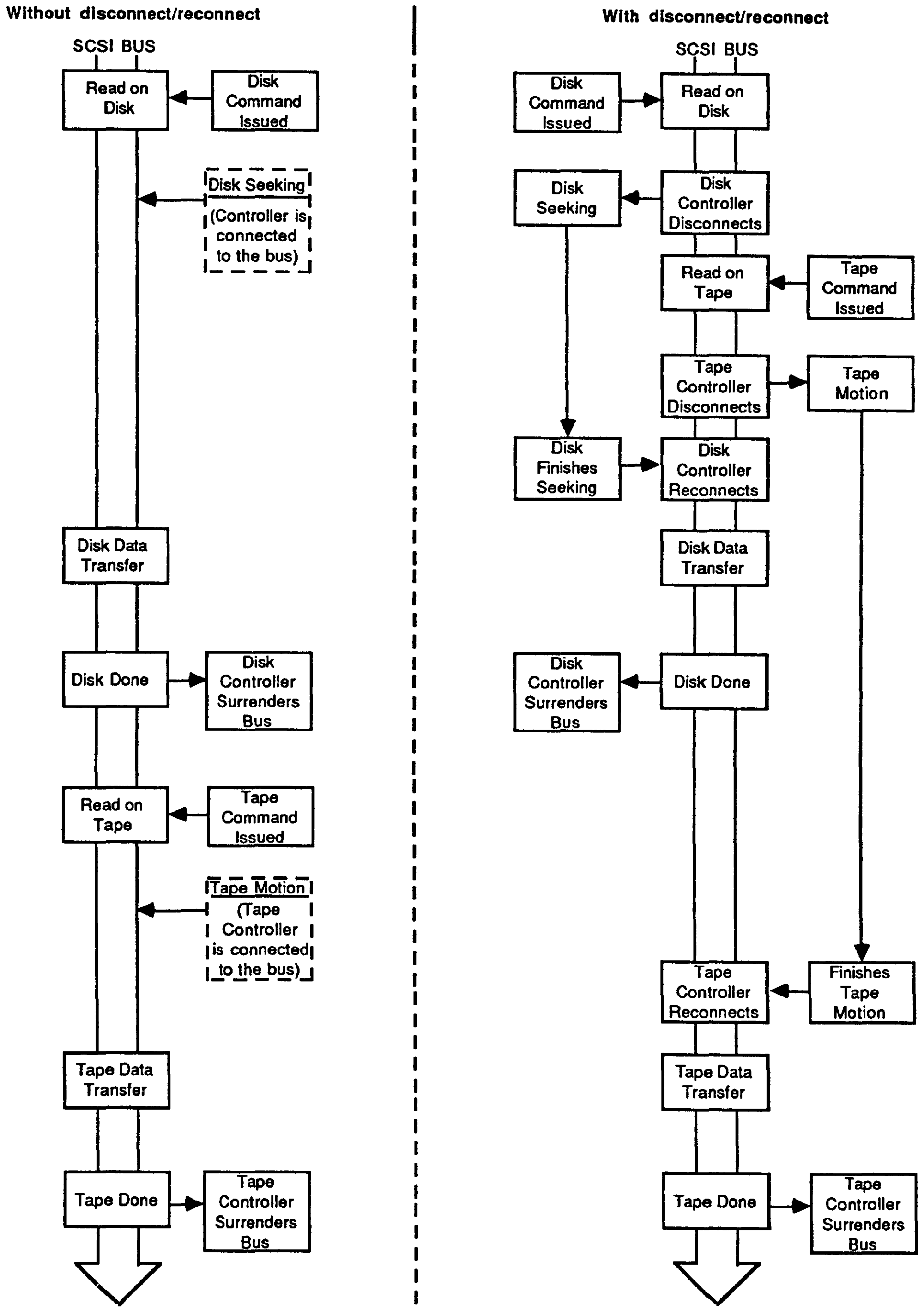
- SCSI-3 type host adapter (available on Sun 3/50s)
- SCSI disk controller(s)
- MT-02 tape controller(s)

Product Description

Disconnect/reconnect is a software facility that enables a disconnect-reconnect capable tape or disk controller to surrender access to (disconnect from) a SCSI bus while performing an operation that does not require the bus's data transfer capabilities. The controller can then regain access to (reconnect) the bus when it needs to resume data transfer. This improves data transfer efficiency over systems without disconnect/reconnect, since the controller no longer is attached to the SCSI bus for an entire "read" or "write" operation. The bus is free to handle incoming commands intended for other controllers, while the disconnected controller performs operations that do not require access to the bus, such as disk seeks or tape movement.

The following chart gives a general picture of the interaction among controllers and a SCSI bus on a system that uses disconnect/reconnect and on one that does not.

Figure 4-1 *SCSI Bus Usage with and without disconnect/reconnect*


Without disconnect/reconnect
SCSI BUS
Read on Disk
Disk Command Issued
Disk Seeking
(Controller is connected to the bus)
Disk Data Transfer
Disk Done
Disk Controller Surrenders Bus
Read on Tape
Tape Command Issued
Tape Motion
(Tape Controller is connected to the bus)
Tape Data Transfer
Tape Done
Tape Controller Surrenders Bus
With disconnect/reconnect
SCSI BUS
Disk Command Issued
Read on Disk
Disk Seeking
Disk Controller Disconnects
Read on Tape
Tape Command Issued
Tape Controller Disconnects
Tape Motion
Disk Finishes Seeking
Disk Controller Reconnects
Disk Data Transfer
Disk Controller Surrenders Bus
Disk Done
Tape Controller Reconnects
Finishes Tape Motion
Tape Data Transfer
Tape Done
Tape Controller Surrenders Bus

As you can see, controllers that do not implement disconnect/reconnect tie up the bus. Once a controller attaches itself to the bus, it locks out all other processing, until data transfer has completed. This is usually not noticeable during disk reads. However, tape operations are significantly slower and noticeably "freeze" the system during the operation, because they prevent disk accesses, such as those that occur during swapping.

With disconnect/reconnect, however, the controller disconnects from the bus during the relatively time-consuming seeking process. Therefore, the bus can handle requests from other processes while the controller seeks to its desired location.

Advantages to the User

Disconnect/reconnect may improve performance on systems that have overlapped I/O operations—the typical UNIX–based system. It is not truly necessary for systems that perform all peripheral operations serially. The most noticeable improvement on a system with disconnect/reconnect is that the system does not seem to pause during a tape operation such as a rewind. Since the tape device no longer takes over the bus for the duration of a tape operation, the system does not "lock out" during that tape operation.

4.2. Subnets

Sun System Release 3.3 includes limited support for Internet standard subnets. These subnets are compatible with all SunLink products, particularly the SunLink inter–network router. You can use them with all Sun architectures; however, to perform a tapeless install on a subnet, you must use a tape server running Release 3.0 or later releases.

Subnets allow more flexibility in the assignment of network addresses. Normally, two bits of the 32 bit Internet Protocol (IP) address tell how much is a network number. There are 127 Class A networks with 24 bit host fields, 16,383 Class B networks with 16 bit host fields, and over two million Class C networks with eight bit host fields.

The problem is that routing can get very complicated as the number of networks grows. For example, a small organization might give each Ethernet a Class C number. As the organization grows, the administration of network numbers could get out of hand. A better idea is to allocate a few Class B network numbers: one for Engineering, one for Operations, one for Support, one for Sales, and so on. Then, divide each Class B network into physical Ethernets using subnets. In this way, machines are isolated from topology changes in remote parts of the organization.

The Sun System Release 3.3 Implementation

When setting up your network, a network-wide *network mask* must be selected. The network mask determines which bits will be the subnet number, and the rest is the host within the subnet. For example, an organization could be one Class B network, with each Ethernet (or SunLink connection) assigned a subnet number within that network. The 16 bits could be allocated as eight for subnet and eight for host, or nine for subnet and seven for host, and so on. However, this decision would be transparent to everyone outside that organization.

You can express network masks as a single hexadecimal number, or as a sequence of four decimal numbers in the IP "dotted notation." The default is a mask of 0xFF000000 (255.0.0.0) for Class A networks, 0xFFFF0000

(255.255.0.0) for Class B networks, and 0xFFFFFF00 (255.255.255.0) for Class C networks. Network masks must only be explicitly specified when they are "wider" (that is, have more one-bits) than the default values. One common case is a Class C mask on a Class B network.

Previously, the check for matching networks always applied the default masks. The check now allows you to use one non-standard mask when the network matches a saved value. Thus, Release 3.3 has one important limitation in its support of subnets: all interfaces with non-default subnet masks must be on the same IP network (but may be on different subnets). In other words, you cannot have more than one subnetted network interfaced to any machine. Usually, a workstation will be on only one subnet; a server will be a gateway between subnets of the same net, and possibly other non-subnetted networks. These cases are supported by 3.3. This temporary restriction should be removed in a future release.

Sun System Release 3.3 also fixes the broadcast problem. Some implementations of IP send broadcasts with a normal network (or subnet) field, but with a host field of all ones. This is correct, according to the IP specification, but may cause earlier Sun software to essentially bring down the network, by broadcasting ARP requests for Host 255. Note that all-one host numbers (for example, x.y.z.255 for Class C networks) work in previous Sun System releases, even though they are not allowed by the IP specification. Starting with this release, all-one host numbers are treated properly, as broadcast.

Enabling Subnets

In addition to the kernel modules needed to support subnets, two programs are involved. The `/etc/ifconfig` program has a new option to set the network mask, and a new `/etc/in.routed` is needed to dynamically maintain the routing tables.

Normally, you should edit the `/etc/rc.local` file on diskfull machines to add an `/etc/ifconfig netmask` command, or else edit the `/etc/rc.boot` file to add the `netmask` option to the `ifconfig` command. Workstations that are booted over the network should then set their masks from their file servers using a network protocol. You can also manually set the kernel variables `subnet_net` and `subnet_mask` using `adb`, although this is discouraged.

For example, consider the Class B network 128.32 with an eight-bit wide subnet field (and, therefore, an eight-bit wide host field), with a server that is Host 1 on Subnet 37, Host 100 on Subnet 3, and is also a gateway to Network 10. The `/etc/rc.boot` file could be as follows:

```
/etc/ifconfig ie0 128.32.37.1 netmask 255.255.255.0 -trailers up
/etc/ifconfig ie1 128.32.3.100 netmask 255.255.255.0 -trailers up
/etc/ifconfig imp0 10.2.0.78
```

Symbolic names for subnet addresses can be defined in the `/etc/hosts` file. These subnet names can be used instead of the numeric addresses as parameters to commands.

Converting to Subnets

Follow these steps to convert your network for using subnets.

1. Decide upon the new subnet topology, including considerations for subnet gateways and locations of hosts on the subnets.
2. Assign all subnet and host addresses.
3. Edit `rc.local` if the host is a diskfull server or standalone workstation.
4. Change hosts addresses.
5. Edit `/etc/hosts` files to change host address. If *yp* is running, change the *yp* database.
6. ND servers should be sure the rarpd runs on the interface connected to the ND client's subnet. The rarpd entry is in the `rc.local` file. Change the IP address of ND clients in the `/etc/tftpboot` directory.
7. *yp* servers should also follow the setup procedure described in *System Administration for the Sun Workstation* (800-1323-03).
8. Connect the physical networks and reboot all machines.

Examples of Subnets

The following examples show network installations where subnets are (and are not) in use:

```
128.32.0.0 Berkeley class B network      (subnetted) netmask 255.255.255.0
36.0.0.0   Stanford class A network      (subnetted) netmask 255.255.0.0
10.0.0.0   ARPAnet  class A network  (non-subnetted) netmask 255.0.0.0
```

All of the University of California at Berkeley is assigned the network number 128.32.0.0, so that any outside user only needs to know one route to access Berkeley. Within the campus, a class C subnet mask is used to give each individual Ethernet a subnet number, with 254 hosts on each of the 254 possible subnets. (Zero and all ones are reserved.) Stanford University uses a class A network number with a class B network mask, for 254 subnets of 65534 hosts each. The ARPAnet itself is a class A network without subnets; therefore, the default class A netmask is used.

Here are examples showing legal and illegal subnet configurations:

- Configuring a Sun with interface addresses of 128.32.1.1 and 128.32.2.1 is legal for Sun 3.3 (two subnets of same net).
- Configuring a Sun with interface addresses of 128.32.1.1 and 128.32.2.1 and 10.2.0.78 is legal for Sun 3.3 (two subnets of same net and non subnetted network).
- Configuring a Sun with interface addresses of 36.8.0.8, 36.10.0.1, and 10.2.0.11 is legal for Sun 3.3 (two subnets of same net and non subnetted network).
- Configuring a Sun with interface addresses of 128.32.1.1, 128.32.2.1, and 36.8.0.8 is **not** legal for Sun 3.3 (two subnets of the same net and another subnetted network).

4.3. Extensions to the Pascal Compiler

Release 3.3 includes two extensions to the Pascal compiler (`pc`). It also includes a new function, `GETFILE`, which has been added to `libpc.a`. Note that programs compiled with earlier versions of `pc` are upward–compatible with the new compiler.

`char` Data Type

The valid range of the `char` data type is now 0 to 255. Its previous range was 0 to 127.

Public or Private Declarations

You can now declare variables, procedures, or functions at the outer block level as either *public* or *private*. For example,

```
var
     total     :   real;
     quantity  :   integer;

private var
     score     :  integer;

private procedure eval(n : integer);
     begin
       .
       .
```

The default scope of an outer block variable, procedure, or function is public. If you declare a variable, function, or procedure as private, then its scope is restricted to the current compilation unit.

`GETFILE` Function

The Pascal runtime library, `/usr/lib/libpc.a`, now contains the `GETFILE` function, which simplifies using the C standard I/O library from Pascal programs.

To use `GETFILE`, declare the following in your Pascal program:

```
type
    iobptr = ^integer;

function GETFILE(var f: text): iobptr;
    external c;
```

`GETFILE(f)` returns a pointer to the C standard I/O descriptor associated with the Pascal file *f*. The result may be passed to `fprintf()`, `fread()`, or other C standard I/O library routines.

4.4. Bug Fixes

The following bugs were fixed in Release 3.3:

- record locking bugs
- `tbl` table layout preprocessor bug
- Sun-3/200 ECC error message bug

- problems with 1/2 inch tape on Sun 3/280s
- MT02 tape appending problems
- MT02 and VPC-2200 tape hangs
- documentation update

These changes are described below.

Locking Bugs

Two file locking bugs files have been fixed.

Using the System V record locking features `lockf` or `fcntl` on a remote file caused the end-of-file mark to be ignored. In such cases, a `read()` past the end-of-file marker always returned successfully. This condition, originally noted in Release 3.2, *Read This First*, has been corrected for Release 3.3.

Another locking–related bug in Release 3.2 caused NFS servers to crash when a client read from a file locked by the System V–compatible `fcntl` or `lockf` routines. The problem only occurred when the client read started at an offset that was not a multiple of four, and the data types crossed a block boundary in the server's file system. This bug was fixed in Release 3.3.

`tbl` Layout Processor Bug

In Release 3.2, `tbl` did not correctly format numeric fields in tables. This was fixed in Release 3.3.

Sun-3 ECC Error Message Bug

In Release 3.2, if a Sun-3/200 had more than one memory board and a correctable memory error occurred, the ECC address of the error, as displayed in the error message, was computed incorrectly.

The old message format was

```
mem1: soft ecc addr 20000000 + bfe980 = 20bfe980 syn 97 <S32,
S4, S1, S0, SX>
```

This error was corrected. The new format of the address is

```
mem1: soft ecc addr 394980 syn 97 <S32, S4, S1, S0, SX> 59 U1459
```

Problems with 1/2 Inch Tapes on Sun 3/280s

Sun-3/280 Workstations configured with GCR 1/2 inch tape drives and 472 Xylogics 1/2 inch tape controllers running 3.2 or previous releases may have experienced failures when executing `tar` or `dump`. These commands failed with the following error message:

```
  xt: bad command synchronization
```

This was fixed in Release 3.3.

MT02 Tape Appending Problems

MT02 tape controller users sometimes were unable to append new files to the end of a tape because it was difficult to reliably reach the end of recorded media. The tape driver would report an error upon reaching the end of tape when the `mt` command was used. This error has been eliminated.

It is suggested that you use the following procedure for positioning to the end of the recorded media:

```
% mt -f /dev/nrst0 fsf nnnn
```

nnnn represents a large count—at least one more than the actual number of files on the tape. Do not worry about the large `fsf` count because any extra `fsf`'s are ignored by the driver. Note that you must use the no-rewind device `/dev/nrst0`. Otherwise, the tape rewinds when the `mt` command exits, undoing the positioning to the end of recorded media.

Please note that appending data is supported only at the end of recorded media. Writing data in the middle of the tape is not supported, due to hardware limitations.

MT02 and VPC-2200 Tape Hangs

Systems with both the MT02 tape controller and the Systech VPC-2200 parallel printer/plotter interface were experiencing tape hangs. When you removed a tape, this left the tape drive in a state that prevented initialization when you inserted the next tape, resulting in the message:

```
st0 : no cartridge loaded
```

This was fixed in Release 3.3

Interphase Controller Reference

Do not use the Interphase Controller reference `(ip0)` on Page 21 of *Installing UNIX on the Sun Workstation* for the 3.2 Release. This controller is no longer supported; therefore, the kernel does not recognize this device.

5

Enhancements in Release 3.4

5

Enhancements in Release 3.4

Release 3.4 contains a number of enhancements to previous releases in the following areas:

- graphics
- SunView
- kernel
- networking
- file formats

The new SunPro optional software provides two enhancements, the filemerge program and a new version of make. filemerge is discussed in its manual page in Appendix C, and make is discussed in a separate manual packaged with your Release 3.4 software.

5.1. Graphics Enhancements

This section describes enhancements that have been made to the *Pixrect*, *SunCGI* and *SunCore* graphics software.

Pixrect Enhancements

Release 3.4 contains the following additional functionality for the *Pixrect* library.

Unstructured Text Functions

The following unstructured functions for drawing text have been added to the *Pixrect* library:

```
pr_text(pr, x, y, op, font, text)
Pixrect *pr;
int x, y, op;
Pixfont *font;
char *text;

pr_ttext(pr, x, y, op, font, text)
Pixrect *pr;
int x, y, op;
Pixfont *font;
char *text;
```

These functions correspond to the pixwin functions pw_text and pw_ttext. prs_text and prs_ttext macros are also provided, although they are identical to `pf_text` and `pf_ttext`, respectively.

Graphics Utilities

This section describes the enhancements to graphics utility programs.

Ordered Dithering with `rasfilter8to1(1)`

In Release 3.4, an ordered dither option has been added to `rasfilter8to1(1)`. This technique often gives better results than thresholding when displaying color images on a monochrome screen. It accepts filename arguments on the command line and accepts any type of rasterfile as input. See the manual page for `rasfilter8to1(1)` in Appendix C.

Single Binary for Graphics Utilities

1) In Release 3.4, the utility programs `clear_colormap(1)`, `rastrepl(1)`, `rasfilter8to1(1)`, `screendump(1)`, and `screenload(1)` have been merged into a single binary. (In `/usr/bin`, `clear_colormap(1)`, `rastrepl(1)`, `rasfilter8to1(1)`, and `screenload(1)` are symbolic links to `screendump(1)`.)

`screendump(1)`

1) In Release 3.4, a no-copy option has been added to `screendump(1)`. This option reduces virtual memory usage when dumping a memory frame buffer at the cost of lengthening the time interval that the frame buffer contents must be stable to ensure a consistent output image.

2) In Release 3.4, the heuristic for finding a color frame buffer has also been improved.

See the manual page for `screendump(1)`.

`clear_colormap(1)`

In Release 3.4, `clear_colormap(1)` has several new options listed below. See the `clear_colormap(1)` manual page for more information.

1) `clear_colormap(1)` clears the overlay plane and initializes the overlay enable plane by default on `cgfour` frame buffers. This behavior can be disabled with the `-o` option.

2) `clear_colormap(1)` can specify a frame buffer device with the `-f` option.

3) `clear_colormap(1)` can disable screen clearing with the `-n` option.

SunCGI Enhancements

Release 3.4 contains the following additional functionality for the *SunCGI* library.

Signal Handling

SunCGI's signal handling has been changed in Release 3.4 to use the Notifier. *SunCGI* applications should now use the Notifier instead of `signal(3)`. As an example, it is easier for an application to request notification of SIGWINCH's than to use `set_up_sigwinch`. For compatibility with previous releases, `set_up_sigwinch` is still available and performs as documented.

Set VDC Extent

```
cgipw_set_vdc_extent(desc, c1, c2)
Ccgiwin *desc;
Ccoor *c1, *c2;
    /* bottom left and top right corner of VDC space */
```

In *CGIPW* mode, `cgipw_set_vdc_extent` changes the coordinate mapping from pixel coordinates to normalized VDC coordinates with origin in the lower-left corner. This function is the *CGIPW* "equivalent" of `vdc_extent`, and its arguments — `c1` and `c2` — are identical to those passed to the standard `vdc_extent` function.

Since all view surfaces share a single VDC mapping, you must use the same kind of coordinates for each surface. In Release 3.2, `cgipw_set_vdc_extent` was designed to work for a single view surface, but could work with multiple view surfaces if each were called discretely using the same arguments (*c1*,*c2*). You must call `cgipw_set_vdc_extent` for each surface (after creating the surface with `open_cgi_pw`) before setting any other attributes or drawing primitives. The sequence of these calls is unimportant — either (*Open*; *Set*; *Open*; *Set*; *Open*; *Set*) or (*Open*; *Open*; *Open*; *Set*; *Set*; *Set*) will work.

Once the `cgipw_set_vdc_extent` function is called, the only way to return to pixwin coordinates is to call `close_pw_cgi`, then `open_pw_cgi` and begin again. The *CGIPW* client should continue using the same *CGIPW* functions (`cgipw_polyline`, `cgipw_rectangle`), because `cgipw_set_vdc_extent` changes only the mapping of coordinate values passed to *CGIPW* functions, and nothing else in *CGIPW* mode.

Additional CGI Error Messages

The Release 3.4 version of *SunCGI* detects additional error conditions. Below is a list of the error codes and their associated error messages.

`EFILACC` [113] `Unable to access file.` An attempt to open or access a file or device failed due to its nonexistence, improper permissions, or other configuration-dependent errors.

`ECGIWIN` [114] `Ccgiwin descriptor is invalid.` A *CGIPW* function was called with a view surface descriptor that is NULL, or that refers to a view surface that is not open. This is probably caused by using the descriptor returned from a failed `open_cgi_pw` or `open_cgi_canvas` call.

`EBADDATA` [95] `Contents of input data record are invalid.` The `Cinrep` data structure passed by an application to an input function is incorrect.

Open a CGI Canvas

```
Cerror open_cgi_canvas(canvas, desc, name)
Canvas *canvas;
Ccgiwin *desc;
Cint *name;
```

In order to make *SunCGI* compatible with *SunView* canvases, the Release 3.4 version of *SunCGI* contains a new view surface initialization function. This must be used instead of `open_cgi_pw`, so *SunCGI* knows to handle coordinate transformation and window repainting in a way that is compatible with the *Canvas* package.

`open_cgi_canvas` is used in place of `open_cgi_pw` to initialize *SunCGI* to use a canvas. This is needed to give *SunCGI* the canvas handle, which is a higher-level object than a pixwin. After calling this initialization function, the resultant descriptor can be treated like that from `open_cgi_pw` for calling any *CGIPW* function, including `close_cgi_pw`.

With the exception of input functions, *CGIPW* functions should work correctly with canvases. In particular, the new *SunCGI* extension `cgipw_set_vdc_extent` will correctly map the VDC extent to the underlying canvas. *SunCGI* input should not be used with canvases, since the *Canvas* package handles all input events on the canvas by calling a client handler function. *SunCGI* has no knowledge of this handler, and would consume input events the *Canvas* package expects, thus interfering with scrollbars and tool border functions such as menus.

Figure 5-1 contains an example program that illustrates a simple use of canvases with *SunCGI*.

Figure 5-1 *CGI Canvas Example Program*

```
#include <suntool/sunview.h>
#include <suntool/canvas.h>
#include <cgipw.h>

Frame   frame;
Canvas  canvas;
Ccgiwin vpw;
int     name, canvas_event_proc();

main()
{
    frame = window_create(NULL, FRAME, 0);
    canvas = window_create(frame, CANVAS,
        CANVAS_AUTO_SHRINK,         FALSE,
        WIN_EVENT_PROC,             canvas_event_proc,
        CANVAS_WIDTH,               1000,
        CANVAS_HEIGHT,              1000,
        WIN_VERTICAL_SCROLLBAR,     scrollbar_create(0),
        WIN_HORIZONTAL_SCROLLBAR,   scrollbar_create(0),
        0);
    open_pw_cgi();
    open_cgi_canvas(canvas, &vpw, &name);
    window_main_loop(frame);
}

canvas_event_proc(window, event)
Window  window;
Event   *event;
{
    Ccoor   lr, ul;

    if (event_is_down(event))
        return;

    switch (event_id(event)) {

    case MS_LEFT:
        ul.x = event_x(event);
        ul.y = event_y(event);
        lr.x = event_x(event) + 10;
        lr.y = event_y(event) + 10;
        cgipw_rectangle(&vpw, &lr, &ul);
        break;

    case MS_RIGHT:
        window_done(frame);
        break;

    default:
        break;
    }
}
```

5.2. SunView Enhancements

Release 3.4 has several enhancements to SunView over Release 3.2, which, in turn, had numerous enhancements and performance improvements compared with Release 3.0. This section describes the enhancements.

Improvements to `cmdtool`

You can now run programs such as `more` and `vi` in a `cmdtool`(1). The changes that support this are as follows:

- `cmdtool` directly supports "raw," "cbreak," and "no-echo" modes (used by `rlogin`, `more`, `su`, and so on). These modes are documented in the `tty`(4) manual page in the *UNIX Interface Reference Manual.*

- □ In programs that perform cursor motion (such as vi, game programs, and any terminal-based applications that use curses), cmdtool's text subwindow is automatically overlaid by a tty subwindow. The cmdtool then responds to the cursor motion escape sequences sent out by the program (documented in console(4S)).

Scroll Mode

Normally, cmdtool only sends the line you are typing to the application when you press RETURN. This means that you can edit the current command as you type it using the mouse and standard text operations.

When an application takes cmdtool out of cooked, echo mode, every keystroke you make is immediately sent to the application. This means that you cannot move the caret[1] backwards to edit the line you are typing — those characters have already been sent to the application. This happens, for example, when you rlogin to another machine.

Of course you can still use the Put and Get commands, and, by selecting 'Enable Edit' from cmdtool's menu, you can edit anywhere in cmdtool's transcript at any time, but the changes will not be sent to the application.

TTY (Disable Scroll) Mode

Basically, well-behaved applications that go into cursor motion mode, such as vi, automatically set cmdtool to the tty mode. The window clears, the scrollbar vanishes, and the caret turns into a rectangular cursor to indicate that the program now controls the insertion point. Since the cursor is moved around under program control, you can not scroll or edit in tty mode, though Put and Get still work. See "Tips and Tricks" below for a way to get the benefit of scrolling while in tty mode.

When a well-behaved program is finished, it will automatically return you to scroll mode.

How TTY Mode Works

Many applications, such as full-screen editors, want to control text display all over their window and position the cursor anywhere within it, instead of interacting with the user on a line-by-line basis. Most such applications use termcap(5) or curses(3X) (which uses termcap) to interact with the display.[2] Some terminals have different modes for the two kinds of interaction, just as cmdtool does, so termcap has always allowed a terminal entry to specify

- □ a ti escape sequence to send to the terminal when entering cursor motion mode

[1] In terminal-based programs, the application controls where new characters that you type will appear. In tty mode, this *insertion point* is marked by a black rectangle which is usually referred to as the *cursor*. In SunView, the *cursor* is also the pointer moved by the mouse independently of each window's insertion point. The SunView text facility lets you set the insertion point for editing using the mouse. The triangle that marks the insertion point in panels and text subwindows is called the *caret*; the active caret flashes by default, though you can turn this off in defaultsedit.

[2] Similar comments apply if you are using System V's terminfo(5V) or curses(3V), both in the optional System V compatibility package. In this the capabilities to enter and exit cursor mode are called smcup and rmcup, respectively.

- a te escape sequence to send to the terminal when exiting cursor motion mode.

In Release 3.4, there is a new entry in termcap, called sun-cmd, for cmdtool. This is the same as the ordinary Sun console termcap entry (sun), except that it defines ti and te escape sequences. The 3.4 cmdtool understands these escape sequences, and changes from scroll mode to tty mode and back as described above.

Compatibility of Applications with cmdtool

Well-behaved applications that want to use cursor motion should send ti and te sequences to the terminal they are running on to set it into cursor mode and out of it, provided that the termcap entry for the terminal specifies them. Thus they will work without modification in cmdtool.

CAUTION **Applications that use cursor motion must send these escape sequences for tty mode to work.** There are several ways this can happen:

- If an application use curses, then curses will send the escape sequences automatically.
- If an application uses termcap directly, it must send out the ti and te sequences itself when entering and exiting cursor motion mode.
- If an application uses a private database of terminal capabilities, a special entry for cmdtool must be added, with the private database's equivalents for ti and te defined.
- If an application doesn't use termcap or its own equivalent, it must be changed to send out the escape sequences described by cmdtool's ti and te sequences; if the application is running in a shelltool, the escape sequences have no effect.

Compatibility of SunOS Releases with cmdtool

If you are in cmdtool and you rlogin to another machine that has an older termcap, you will get the message

```
Type sun-cmd unknown
```

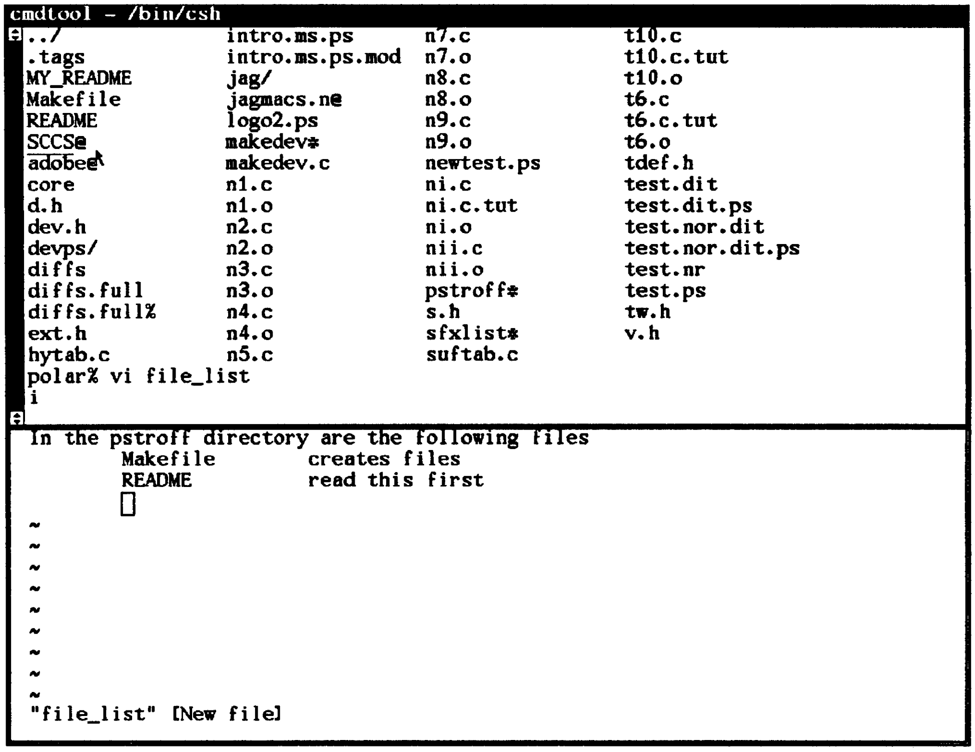
and tty mode will not work. The work-around is to set the terminal type to sun and manually enter tty mode when desired. (see *Changing Modes Manually* below); The solution is to merge the sun-cmd entry into the other machine's termcap, or copy the entire 3.4 /etc/termcap file to the other machine.

Changing Modes Manually

It is sometimes useful to be able to set cmdtool to tty mode or scroll mode yourself. In Release 3.4, the menu in each mode has a new menu item to change to the other mode: in scroll mode there is a menu item to 'Disable Scroll' and in tty mode there is a menu item to 'Enable Scroll.' Remember that in normal use, cmdtool jumps between the two modes automatically.

Tips and Tricks

You can 'Split' `cmdtool` into two or more separate subwindows using the menu command of the same name. Each subwindow can be scrolled independently. When a program sends `cmdtool` into tty mode, only one subwindow is overlaid by a tty subwindow. It is the one with the *keyboard focus* at the time; by default, the subwindow that the cursor is in has the keyboard focus. The other subwindows can still be scrolled to view and select previous commands, as shown in this example:



```
cmdtool - /bin/csh
../              intro.ms.ps      n7.c            t10.c
.tags            intro.ms.ps.mod  n7.o            t10.c.tut
MY_README        jag/             n8.c            t10.o
Makefile         jagmacs.ne       n8.o            t6.c
README           logo2.ps         n9.c            t6.c.tut
SCCSe            makedev*         n9.o            t6.o
adobee           makedev.c        newtest.ps      tdef.h
core             n1.c             ni.c            test.dit
d.h              n1.o             ni.c.tut        test.dit.ps
dev.h            n2.c             ni.o            test.nor.dit
devps/           n2.o             nii.c           test.nor.dit.ps
diffs            n3.c             nii.o           test.nr
diffs.full       n3.o             pstroff*        test.ps
diffs.full%      n4.c             s.h             tw.h
ext.h            n4.o             sfxlist*        v.h
hytab.c          n5.c             suftab.c
polar% vi file_list
i
```

```
In the pstroff directory are the following files
        Makefile        creates files
        README          read this first
        □
~
~
~
~
~
~
~
~
~
"file_list" [New file]
```

Figure 5-2 *A Split* `cmdtool`

Note: In normal use, you do not need to use these techniques to go from one mode to the other — it happens automatically.

In addition to the menu items provided to change modes, it is also possible to change from one mode to the other via a function key or an alias. The two escape sequences are `ti` and `te` in the `sun-cmd` entry in `/etc/termcap`:

`ti` to begin tty mode: `\E[>4l`
and
`te` to return to scroll mode: `\E[>4h`

`\E` means the Escape character.

So, to go from one mode to the other, you can map the two escape sequences to keys in your `~/.ttyswrc` file. The following example maps them to top function keys F8 and F9:

```
# go to tty mode
mapo T8 \E[>4l
# go to scroll mode
mapo T9 \E[>4h
```

You can also set these up as aliases in your `~/.cshrc` file as follows:

```
if ($?WINDOW_PARENT) then
        alias gotty 'echo -n "^[[>4l"'
        alias goscroll 'echo -n "^[[4h"'
endif
```

(If you are using the C-shell, to get the Escape character `^[` in the file, you may have to precede it with [CTRL-V].)

Things to Watch For

File completion in the C-shell (`set filec`) does not work properly in a `cmdtool`.

You must be careful when typing ahead, because characters typed while `cmdtool` changes modes may be lost or sent to the other mode's window.

The scroll mode of `cmdtool` picks up most of the *Text* defaults in `defaultsedit`. However, if *Text/Font* is different than the font used in tty subwindows (because *SunView/Font* is different, or because the `-Wt` *fontname* frame command line argument was used), then the font will *not* change when the subwindow goes from one mode to the other. Otherwise, for example, the number of rows and columns will be different in different modes.

In normal use, you should not have to change modes manually. However, there are a few reasons why you may need to:

- Some programs do not output the correct escape sequence to go into and out of tty mode.
- In Release 3.4, only the tty subwindow supports file completion (**`set filec`**) in the C-shell.
- In Release 3.4, only the tty subwindow can embolden characters (used, for example, by `man(1)` to highlight words).
- When the root file system fills up so that `cmdtool` cannot write to its transcript file (`/tmp/tty.txt.`*zProcess_id*) or its edit log (`/tmp/Text`*Process_id*`.`*counter*), `cmdtool` switches into tty mode. It does not detect when you have freed up enough disk space; you have to explicitly put it back into scroll mode.

Menu Enhancements

Stay-up Menus

You can change the behavior of pop-up menus so that they stay up even when you take your finger off the menu button on the mouse. Normally, a menu pops up as soon as you press the mouse's menu button, and disappears when you release it. However, if you set *Stay_up* in the *Menu* category of `defaultsedit(1)` to *True*, then the first click of the menu button displays the menu. The second click selects the highlighted menu item, if any, and makes the menu go away. Between the two clicks, the menu remains visible, and grabs all window input — you cannot type or select in any window while the menu is up.

Note: The right button on the mouse is usually the menu button, although this can be changed using the *Input* category of `defaultsedit`.

If you hold the menu button down in *Stay_up* mode, the cursor changes to the following shape to indicate that the menu will appear when you release the button:

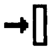


Figure 5-3 *Cursor Indicating "Stay Up" Menu Imminent*

Another new attribute in the same category lets you set your menus so that menu strings come up centered; set *Center_string_items* in the *Menu* category of `defaultsedit(1)` to *True*. This helps to differentiate different menu options. Try it to see if you like it. Here is a sample.

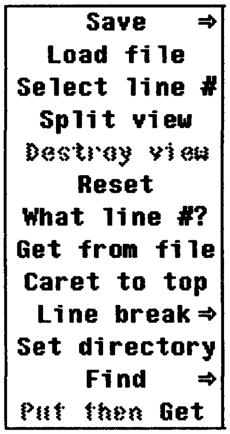


Figure 5-4 *Sample Centered Menu*

Row-Column Order in Menus

Some applications put up menus in several columns, for example the menu for `mailtool`'s folder button. The default order for the items is row-major order, so that adjacent items are in the same row and different columns. But if you set *Items_in_column_major* in the *Menu* category of `defaultsedit(1)` to *True*, then items are laid out in columns. This is the way `ls(1)` displays files.

Menu Compatibility

These effects (and many others available in the *Menu* category of `defaultsedit`) only work if you have *Walking_Menus* set to *Enabled* in the *SunView* category of `defaultsedit`.

In Release 4.0, the default will be walking menus, although Sunwindows-style menus will still be provided for compatibility with releases prior to 3.0.

Corresponding New Menu Attributes

The menu attributes that correspond to these new features are `MENU_STAY_UP`, `MENU_CENTER` and `MENU_COLUMN_MAJOR` in `<suntool/walkmenu.h>`. Each takes a Boolean value that turns the feature on or off.

Text Subwindow Enhancements

A variety of bug fixes and enhancements have been made to the text subwindow package (used in `textedit(1)` and many other programs):

''Quad-clicking'' Now Selects the Entire Document

The fourth click in succession of the Select mouse button selects the entire contents of the text subwindow. As before, the first click selects a character (and sets the insertion point); the second click selects a word; and the third click selects a line. The defaults *Multi_click_space* and *Multi_click_timeout* in the *Text* category of `defaultsedit` determine the maximum distance in space and time between clicks for them still to be considered a multi-click.

Positive Confirmation When 'Find' Fails

When 'Find' in a text subwindow fails to find characters matching the selection, the text subwindow flashes and beeps.[3]

The Upper Context is Set Correctly After 'Find'

When 'Find' succeeds and displays the matching characters, it now displays the lines above the match (the ''upper context'') instead of positioning the match at the top of the window. *Upper_context* in the *Text* category of `defaultsedit` determines the number of lines above the selection that are displayed.

Text Subwindow's Temporary File Name Shortened

The temporary file that the text subwindow uses to store your edits to the file has been shortened to `/tmp/Text`*Process_id*`.`*counter* from `/tmp/EtHost`*hostnumber*`Process`*Process_id*`Counter`*counter*.

Specifying Different Defaults

The `defaultsedit` program maintains a personal `~/.defaults` file that records your SunView preferences; this defaults file is examined when tools start up. Often you want different tools to use different defaults. For example, you might want a `textedit` used to edit multiple files to have a small font and normal menus, while you want `mailtool` to use a large font and *Stay_up* menus.

In 3.4, tools now look at the `DEFAULTS_FILE` environment variable (if it exists) to determine where to read the user's preferences. You can set this to point to a special defaults file before you invoke a tool. If you are a C-shell user, you would go through steps like the following to create this special file:

[3] Flashing and beeping can be individually turned off by setting *Audible_Bell* and *Visible_Bell* in the *SunView* category of `defaultsedit` to *Disabled*.

```
% mv ~/.defaults ~/.defaults.hold
% defaultsedit
(Make a special set of defaults.)
Save them and quit defaultsedit.)
% mv ~/.defaults ~/.defaults.special
% mv ~/.defaults.hold ~/.defaults
```

Then, when you want to start up a tool, `mailtool` for example, using the special defaults, type:

```
% (setenv DEFAULTS_FILE ~/.defaults.special; mailtool) &
```

This is especially useful if you use several machines with different size and resolution displays — you can make a second `.defaults` file that specifies larger fonts for text and menus, larger scrollbars, and the like. You can select this automatically from your `~/.login` file.

```
if ('hostname' == "high-res_machine") then
        setenv DEFAULTS_FILE ~/.defaults.hires
else
        setenv DEFAULTS_FILE ~/.defaults
endif
```

Miscellaneous Enhancements Visible to the User

If you make changes to one of the categories in `defaultsedit`, then try to change to another category, the program warns you that your changes will be lost if you do not 'Save' them.

New TTY Subwindow Attributes

New attributes for tty subwindows give programmers more control over when and whether a process is forked to run in a frame's tty subwindow.

Table 5-1 *New TTY Subwindow Attributes*

Attribute	*Type*	*Description*
TTY_TTY_FD	int	File descriptor of the tty associated with the tty subwindow. Get only.
TTY_PID	int	Process ID of tty subwindow child or handle of client running in the tty subwindow. A value of TTY_INFINITY says there is no program or client running in the tty subwindow. If get returns a value of -1, the child was not forked successfully.

TTY_TTY_FD

You can use `TTY_TTY_FD` to read and write to the pseudotty in the tty subwindow using standard UNIX I/O routines. Note that `TTY_TTY_FD` is the file descriptor of the pseudotty, not the file descriptor of the tty subwindow. The latter is used for some low-level window manipulation procedures.

TTY_PID

You can use TTY_PID to monitor the state of the child process running in the tty window via the Notifier using notify_set_wait3_func(). The client's wait3() function gets called when the state of the process in the tty subwindow changes. It can then do something useful, such as destroying the tty window or starting up another process in the tty subwindow.

You can set TTY_PID as well as get it, but if you set it then you are responsible for setting the notify_set_wait3_func() to catch the child's death, and for making the standard input and standard output of the child go to the pseudotty.

TTY_ARGV_DO_NOT_FORK

There is a new potential value for TTY_ARGV. Setting a value of TTY_ARGV_DO_NOT_FORK tells the system not to fork a child in the tty subwindow. In combination with TTY_FD, this allows the tool to use standard I/O routines to read and write to the tty subwindow. This simplifies porting terminal-oriented graphics programs, which interact with the user on the model of *write a prompt... read a reply*, to SunView. However, in most cases you should redesign programs to use a real windowing interface made up of SunView components.

This new capability makes obsolete the work-around required in previous releases of SunView if you wanted a window program to read and write from its own tty subwindow. The old approach was demonstrated in the example programs typein.c and loopback.c listed in the *Example Programs* chapter of the *SunView Programmer's Guide* and included in optional software in /usr/src/sun/suntool. A revised version of typein.c is in Section 7.7, ''Errata and Addenda for the SunView Programmer's Guide,'' of this manual, and is also on the release tape.

Sample Uses of New TTY Attributes

Here are some other sample uses of the new attributes.

Example Using TTY_TTY_FD to Read/Write Standard Input/Output

The following program is the framework for the same kind of *write a prompt... read a reply* kind of program as typein, but is a more straightforward conversion to SunView. Instead of rewriting applications to move their main flow of control into window_main_loop(), you can call notify_do_dispatch() to tell the Notifier to run when it can. The Notifier will handle selections, menu processing, window resizing, and so on, each time the program does a read() or write(). Read ''Porting Programs to SunView'' in the *Notifier* chapter of the *SunView Programmer's Guide* for more information.

```
#include <stdio.h>
#include <sys/wait.h>
#include <suntool/sunview.h>
#include <suntool/tty.h>
#include <suntool/textsw.h>

#define BUFSIZE 1000
static int      my_done;
static Notify_value
my_notice_destroy(frame, status)
        Frame           frame;
        Destroy_status  status;
{
        if (status != DESTROY_CHECKING) {
            my_done = 1;
            (void)notify_stop();
        }
        return (notify_next_destroy_func(frame, status));
}

main(argc, argv)
        int      argc;
        char    *argv[];
{
        Frame    base_frame;
        Tty      ttysw;
        int      tty_fd;
        char     buf[BUFSIZE];

        my_done = 0;
        base_frame = window_create(NULL, FRAME,
                        FRAME_ARGC_PTR_ARGV,    &argc, argv,
                        0);

        ttysw = window_create(base_frame,       TTYSW,
                        TTY_ARGV,               TTY_ARGV_DO_NOT_FORK,
                        0);
        tty_fd = (int)window_get(ttysw, TTY_TTY_FD);
        dup2(tty_fd, 0);
        dup2(tty_fd, 1);

        (void)notify_interpose_destroy_func(base_frame, my_notice_destroy);
        window_set(base_frame, WIN_SHOW, TRUE, 0);
        (void)notify_do_dispatch();
        puts(prompt_to_user);
        while (gets(buf)) {
            if (my_done)
                break;
            /*
             *  This is where the meat of the program
             *  would be if this were a real program.
             */
            puts(buf);
        }
        exit(0);
}
```

Example Using TTY_PID to Fork a New Process on Child Death

This program is notified of the death of its tty subwindow's child and forks a new one. To do this, you interpose a wait3() function using notify_interpose_wait3_func(), which "watches" the process ID of the child program running in the tty subwindow.

```
#include <stdio.h>
#include <sys/wait.h>
#include <suntool/sunview.h>
#include <suntool/tty.h>
#include <suntool/textsw.h>

static char     *my_argv[] = {"my_prog", 0};

static Notify_value
my_wait3(ttysw, pid, status, rusage)
        Tty              ttysw;
        int              pid;
        union wait      *status;
        struct rusage   *rusage;
{
        int     child_pid;

        notify_next_wait3_func(ttysw, pid, status, rusage);
        if (! (WIFSTOPPED(*status))) {
            window_set(ttysw,
              TTY_QUIT_ON_CHILD_DEATH,  FALSE,
              TTY_ARGV,                 my_argv,
              0);
            child_pid = (int)window_get(ttysw, TTY_PID);
            notify_interpose_wait3_func(ttysw, my_wait3, child_pid);
        }
        return NOTIFY_DONE;
}

main(argc, argv)
        int     argc;
        char    *argv[];
{
        Frame   base_frame;
        Tty     ttysw;
        int     child_pid;
        int     tty_type_flag = 0;

        base_frame = window_create(NULL, FRAME,
                        FRAME_ARGC_PTR_ARGV,    &argc, argv,
                        0);
        ttysw = window_create(base_frame,       TTY,
                  TTY_QUIT_ON_CHILD_DEATH,      FALSE,
                  TTY_ARGV,                     my_argv,
                  0);
        child_pid = (int)window_get(ttysw, TTY_PID);
        notify_interpose_wait3_func(ttysw, my_wait3, child_pid);
        window_main_loop(base_frame);
        exit(0);
}
```

SunView Library Changes

lint Libraries

New lint libraries for libsuntool and libsunwindow have been created. You can run your SunView applications against these using the lint(1) program to find potential bugs in them at compile time. The lint libraries are

```
/usr/lib/llib-lsuntool.ln
/usr/lib/llib-lsunwindow.ln
```

To run lint against the program *mysunview.c*, you would type something like:

```
% lint mysunview.c -lsuntool -lsunwindow -lpixrect
```

Profiled Libraries

Profiled versions of libsuntool, libsunwindow, and libtoolmerge[4] are available in Release 3.4 if you want to analyze the execution of your SunView applications. The profiled libraries are

```
/usr/lib/libsuntool_p.a
/usr/lib/libsunwindow_p.a
/usr/lib/libtoolmerge_p.a
```
(there is no profiled version of libpixrect, *yet)*

The profiled libraries are only installed if during installation you answer **y** to the following question:

```
Do you want to install the new window profiled libraries ? :
```

during the 3.4 upgrade procedure; see step 10 in section 2.4. To compile the program *mysunview.c* in preparation for profiling with gprof(1), you would type something like

```
% cc -pg mysunview.c -lsuntool_p -lsunwindow_p -lpixrect
```

The gprof(1) and cc(1) manual pages have more information on profiling.

Improved Attribute-Value List Error Handling

The maximum length of attribute-value lists supported by the SunView packages (see ATTR_STANDARD_SIZE in <sunwindow/attr.h>) has always been 250. In this version of SunView, if the number of attributes in a list you pass to SunView exceeds this size, the attribute package now prints

[4] libtoolmerge contains the .o files for all the tools in the suntools and othertools tool merges. It is used only when you build your own custom-merged tool set, as documented in Section 4.8, "SunView Enhancements," of the *Release 3.2* manual.

```
Number of attributes(nnn) in the attr list exceeds
the maximum number(nnn) specified.  Exit!
```

on standard error and exits with exit status 1.

There is an addendum for this in Section 7.7, ''Errata and Addenda for the SunView Programmer's Guide,'' of this manual.

5.3. Kernel Enhancements

Release 3.4 changes the way in which soft ECC errors are handled by Sun-3/260s. Previously, these errors appeared on the console. Now they only appear in the file /usr/adm/messages.

5.4. Networking Enhancements

Release 3.4 includes the following enhancements to networking capabilities.

Name Resolver Library

The Internet name resolver library, which was described in the `resolver`(3) and `resolver`(5) manual pages, is now included in the file /usr/lib/libresolv.a.

New IP, TCP, and ICMP from 4.3BSD

The kernel network code that implements IP (Internet Protocol), TCP (Transmission Control Protocol), and ICMP (Internet Control Message Protocol), has been updated to include many features of 4.3BSD for better congestion control.

RPC Programs Read Broadcast Address

Release 3.3 introduced the ability to set the broadcast address for any interface and accept any of the several "standard" broadcast addresses now in use. Release 3.4 includes new copies of RPC programs, such as `ypbind`, `rup`, and `rusers`, that read the broadcast address from the interface instead of always using the network number plus a host number of all zeros.

5.5. File Formats

The `termcap` database now contains a new entry , `sun-na`, to describe Sun workstations with arrow keys disabled. This was done to support SunSimplify database screens.

6

Bug Fixes Since Release 3.3

6

Bug Fixes Since Release 3.3

Release 3.4 is a major bug fix release. This chapter describes bugs from Releases 3.2 and 3.3 that have been fixed in 3.4. The bug fix descriptions are organized into the following sections:

- languages
- graphics
- SunView
- SunOS
- networking
- system administration

6.1. Language-related Bugs Fixed

This section describes the languages bugs that were fixed in Release 3.4.

as Bugs

68020 Addressing Bug

Previous versions of as would not properly assemble statements of the form

```
movl     @(10:w,d3:1:4), d2
```

on Sun-3s.

The problem arose because as permitted you to specify a coprocessor ID or an FPA base register after the opcode, but did not require it to come *immediately* after the opcode.

This bug has been fixed in Release 3.4.

Assembler Long Jump Bug

as sometimes generated many lines like the following:

```
as: error (t.s:36277): Multiply defined symbol (phase error)
```

if a program contained long jmp instructions.

If you used the `-j` option of `as` to correct this during the assembly, the resulting object file did not work with the loader. The problem with `-j` was due to the processing of link instructions. This has been fixed in 3.4.

C-related Bugs

The following C language-related bugs have been fixed in Release 3.4.

op REG Compiler Error

Previous versions of `cc` could generate an error message like

```
"interp.c", line 14: compiler error: no table entry for op REG
```

if you wrote a program with chained structure assignments in a single statement.

The bug has been fixed in this release.

Long Filenames Crashed `cc`

Previous versions of `cc` limited filenames to 100 characters. If you invoked the compiler with a filename argument exceeding 100 characters, `ccom` would dump core with a segmentation fault.

This bug has been fixed.

C Compiler Error with `-f68881` Set

In previous versions of `cc`, compiling a C program that cast an unsigned value to a double would generate

```
  compiler error: expression causes compiler loop: try simplifying
```

if the `-f68881` compiler flag was set. The bug has been fixed.

Casting Error Bug

Previously, `cc` sometimes incorrectly performed type conversions with unsigneds and bit fields. In binary operations involving an integer and an unsigned, sometimes the unsigned quantity would be converted to integer, rather than the integer being converted to unsigned.

The bug has been fixed in this release.

Optimizer Failure

Before the 3.2 release, the C code optimizer sometimes generated code that used one or more incorrect registers, usually when doing compares.

The bug has been fixed in this release.

Optimizer Lost Address Register

In Release 3.1, `cc` lost track of address register offsets if you used the `-O` option. A workaround was to disable one of the `c2` optimizations, at a cost of a slight reduction in code quality, as follows:

```
  cc -Qoption c2 -dcoalesce
```

The bug has been fixed in this release.

cc Mishandled ',' Operator

Previous versions of cc mishandled C's "comma" operator during code generation. The problem resulted in a

```
compiler loop: try simplifying
```

error message. The bug is fixed in this release.

3.1 Optimizer Bug

The Release 3.1 optimizer could generate bad code losing store instructions and violating register d6.

The error was caused by the optimizer's improper specification of a redundant operand pattern. This caused a3@+ in some code to be deleted because the pattern included it in the same category as a3.

cc -f68881 Bug

In Release 3.2, if the -f68881 flag was set, cc would not compile unary functions except floats or doubles.

The problem occurred because when the -f68881 flag was set, cc incorrectly evaluated a double precision operand in a d-register pair for condition codes. The same code compiled properly if either -fsoft or -fswitch were set.

The bug has been fixed in this release.

Division by Powers of 2 Bug

In Release 3.2, the C compiler's special-case code to handle division by powers of 2 on Sun-3s (MC68020s) did not properly manage register use. The equivalent code generated for Sun-2s (MC68010s) was correct.

The bug has been fixed in this release.

doscan Bug

In previous releases, the 4.2BSD version of doscan did not properly handle bad hexadecimal input, passing it through to the calling procedure. The bug has been fixed in this release.

Coercion Bug

The Release 3.1 C compiler had a bug that caused it to generate incorrect ints that have been cast to shorts. A segment of code like the following:

```
    int  x,y;
    if ((short) x == (short) y)...
```

did not work properly; instead of masking off the high-order bits and doing a 16-bit compare, it performed a 32-bit compare.

The bug has been fixed in this release.

Unneeded File Bug

Previously, cc would invariably access /usr/lib/pcexterns.o, even if it did not need to. This meant that /usr/lib/pcexterns.o had to exist whenever the compiler driver was run.

The bug has been fixed in this release.

cc Conversion Bug

In a previous release, cc incorrectly evaluated compile-time unsigned-to-double conversions.

The bug has been fixed in this release.

Type-casting Bug

Previously, the C compiler ignored type casting in if statements. Type casting outside of if statements was done correctly. The bug has been fixed in this release.

Integral-conversion Bug

Previously, the C compiler sometimes deleted necessary small integral conversions in programs. The bug has been fixed in this release.

No Overwrite Warning Bug

In Release 3.2, cc did not warn you if your source file would be overwritten. The bug has been fixed in this release.

Peephole Optimizer Bug

Previously, the peephole optimizer /lib/c2 generated incorrect code if it encountered a structure passed by value and the total size of the argument list was not a multiple of 4.

The bug has been fixed in this release.

Optimizer Register Use Bug

Previously, the peephole optimizer /lib/c2 lost track of actual register usage when overlapping memory operands were modified. If the operand in memory was modified, the value used by compiled code (a copy saved in a register) would not be changed and would be incorrect.

The bug has been fixed in this release.

Register Variable Bug

Previously, cc would issue the following error message:

```
no table entry for op REG
```

error message when compiling a program with code like

```
int test[100];

main()
 {
   register int a, b, c, d, e, f;

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

because it allocated too many registers for double indexing, leaving too few for the "&" operation.

The bug has been fixed in this release.

Debugger-related Bugs

The following debugger-related bugs were fixed in Release 3.4.

dbx #include Bug

In some previous releases, dbx would not stop at any lines in a program that had #include'd initialized variables, such as sccs id's in the file header.

The bug has been fixed in this release.

dbx Printing Bug

Previously, dbx printed incorrect values for odd-length char arrays in structures. For example, dbx would print that the value of c1 in the example below as “\0”, rather than “abcde” at the last line.

```
struct {
        char c1[5];
} s;
main()
{
        s.c1[0] = 'a';
        s.c1[1] = 'b';
        s.c1[2] = 'c';
        s.c1[3] = 'd';
        s.c1[4] = 'e';
}
```

The bug was fixed in this release.

dbx Stack Trace Bug

Previously, dbx would fail to get a correct stack trace of a program that died in the middle of any routine that did not do a link to establish a stack frame. The bug has been fixed in this release.

dbx Function Pointer Bug

Previously, the dbx command

```
print &gen_replrop
```

returned

```
&gen_replrop = gen_replrop()
```

The command now prints out the address of the function in hexadecimal, in addition to the name of the function referred to.

dbx Casting Bug

Previously, dbx would dump core rather than cast a negative floating-point constant to type double. The bug has been fixed in this release.

Incorrect Help Bug

Previously, the `dbx help` command's description of `dbx's clear` command was wrong, implying that you could give more than one line number to be cleared. The bug has been fixed in this release.

`dbx` Symbol Table Bug

In a previous release, `dbx` would dump core if you tried to debug a file without a symbol table. The bug has been fixed in this release.

`dbx` Error Message Bug

Previously, `dbx` gave an incorrect error message if you tried to debug a program without a core file and tried to list the beginning of the program.

The bug has been fixed in this release.

`dbx` Prefix Bug

Previously, `dbx` failed to find source files in some situations if you specified them with prefix paths. It would work properly if you did your debugging in the object files' directory. The bug has been fixed in this release.

`where` Command Bug

Previously, `dbx`'s `where` command sometimes looped when dealing with complicated expressions in FORTRAN programs with the `-f68881` flag set. Either simplifying the program or using a different floating-point option got around the problem.

The bug has been fixed in this release.

`catch` and `ignore` Report Bug

Previously, `dbx`'s `catch` and `ignore` commands did not properly report the status of signals that are to be caught or ignored. The bug has been fixed in this release.

`dbx clear` Command Bug

Previously, if you issued `clear` while in `dbx`, you would get the following message:

```
<file "" was not compiled ... "-g">
```

instead of clearing the current breakpoint.

The bug has been fixed in this release.

`dbx` "No Program" Bug

Previously, if you invoked `dbx` with no arguments, then issued `run` *with* arguments, `dbx` died with the following error message:

```
dbx: fatal error: bad address
```

The bug has been fixed in this release.

`dbx` Array Index Bug

Previously, `dbx` refused to index an array with a variable of type `char`, giving the following message:

```
subscript charvar is the wrong type
```

This type of operation is permitted in Release 3.4.

Private Procedure Bug

Previously, dbx would not recognize Pascal private procedures or functions, giving the error message

```
dbx: internal error: level error in symbol entry for outer
```

instead.

The bug has been fixed in this release.

Symbolic Link Bug

Previously, dbx would not find an executable to debug if it was symbolically linked. The bug has been fixed in this release.

Double Negative Bug

Previously, dbx would die if you told it to print the negative of a double or do an assign where the variable being assigned to and the value being assigned are double type and the value is negative. The bug has been fixed in this release.

Numeric Conversion Bug

Previously, dbx would sometimes fail to convert integers properly to floating-point. The bug has been fixed in this release.

FORTRAN-related Bugs

The following FORTRAN bugs were fixed in Release 3.4.

Complex Number Compare Bug

Previous versions of f77 would return the following message from the loader:

```
undefined:
        Fz_eq
        Fz_ne
```

if the program being compiled compared two complex or doublecomplex numbers.

The bug has been fixed in this release.

INCLUDE Bug

Previously, an INCLUDE in a FORTRAN program would result in the compiler failing and dumping core. This was caused by the compiler getting confused and losing track of the name of the source file being compiled.

The bug has been fixed in this release.

f77 'op REG' Error Bug

Previously, f77 would sometimes generate

```
compiler error: no table entry for op REG
```

when you tried to compile a program using doublecomplex numbers.

The bug has been fixed in this release.

Number Scaling Error Bug

Previously, the following code produced a segmentation violation at runtime if the number was 0.0 or less:

```
        write (*, " (-6pf7.3)" 0.0
        stop
        end
or
        write (*, 10) 0.0
10      format (-6pf7.3)
        stop
        end
```

The bug has been fixed in this release.

`f77 read` Bug

Previously, the FORTRAN list-directed runtime routines did not correctly test for invalid input. This resulted in the `read` statement, in some cases, correctly detecting the error and stopping, but not flagging an error.

The bug has been fixed in this release.

FORTRAN Uppercase Bug

`f77` accepts uppercase source and converts it to lowercase by default. In such a case, neither `dbx` or `dbxtool` recognize uppercase if pointed at.

Using the `-U` option resulted in compiler options, such as `OPEN`, not being recognized.

The bug in `-U` operation has been fixed in this release.

`f77 read` Bug

Previously, the FORTRAN list-directed runtime routines did not correctly test for invalid input. This sometimes resulted in a program dumping core after receiving invalid input.

The bug has been fixed in this release.

Bug in Assembly Phase of Inline Expansion

Previously, some `f77` programs compiled on Sun-2s would fail during the assembly phase of inline expansion if the program called for converting a FORTRAN character variable to an integer. If the optimizer was invoked, some MC68020 instructions were included in the expanded code.

The bug has been fixed in this release by removing all non-MC68010 instructions from the files `/usr/lib/{fsoft, fswitch, fsky}.il`.

Inline Expansion Bug

In Release 3.2, using

```
f77 -O
```

automatically resulted in inline code expansion.

The bug has been fixed in this release. Inline expansion only occurs now when an inline expansion template file is listed in the command line invocation.

f77 -a Bug

In Release 3.2, compiling a FORTRAN program with the -a flag set would cause a core dump when you tried to run the program. The bug has been fixed in this release.

f77 Compiler Bug

In previous releases of f77, statements of the form

```
WRITEFRMAT4),300, 'ERR=10)NC
```

caused the optimizer to fail. The problem occurred if any variable in the control list of an I/O statement was a constant substring or had a subscript greater than 1. You received a message like the following:

```
f77 -c -O -ffpa - Compiler generated error
    find_parent: couldn't find parent of T[n]
   ***Error code 1
```

If the -O option had been specified, the following error message would be generated:

```
compiler (iropt) error: find_parent: couldn't find parent of T[#]
```

This bug has been fixed in Release 3.4.

Incorrect Handling of Backslash Characters

Previously, f77 did not correctly handle the backslash character (\). It converted a double backslash (\\) to a single backslash, requiring you to type four backslashes (\\\\) for a double backslash. The compiler converted \0 to the ASCII character 0, which is the string terminator for C, rather than FORTRAN. This has been fixed in Release 3.4.

Problems with Profiling Options

Previously, f77 occasionally generated

```
as: warning (/tmp/f77pass1.1984.d.2.s:70): Undefined L-symbol
Undefined:
LPG1_158098785
```

messages from the assembler if you used the -p or -pg profiling options. The problem was related to very long jumps in the code sent to the assembler. This assembler bug was fixed in Release 3.4 and is documented in Chapter 6 of the release manual.

Redundant I/O-List Parentheses Bug

Previously, the compiler reported an error if an expression in the I/O list of a write statement was enclosed in a redundant set of parentheses.

The bug has been fixed in this release.

system and fork Error Bug (1003162)

Previously, the system and fork routines in the FORTRAN library could dump core at runtime.

The bug has been fixed in this release.

Pascal-related Bugs

The following Pascal bug was fixed in Release 3.4.

Pascal Filename Bug

In Release 3.2, pc failed to compile a program if you invoked it with the -i option set and passed it a source filename containing a period.

The bug has been fixed in this release.

Library Bugs Fixed

The following bugs in the libraries were fixed in Release 3.4.

prof.h Failure

The include file prof.h, described in prof(3), did not work as described in the manual page. Trying to use it so resulted in "undefined loop" error messages.

The bug has been fixed in this release.

Missing Library Routine

In Release 3.2, the C library routine putpwent was in the System V library but was missing from the 4.2BSD library. putpwent writes out a password structure to be placed in /etc/passwd.

That omission has been corrected in this release.

System V putpwent Bug

The UNIX System V version of putpwent incorrectly handled negative UIDs or GIDs as unsigned quantities.

The bug has been fixed in this release.

System V ioctl Bug

In an earlier release, ioctl could set incorrect terminal modes in the UNIX System V environment if you tried to set them to support a full eight-bit data path.

The bug has been fixed in this release.

System V-related curses Bug

The UNIX System V version of curses dumped core rather than just not working correctly, if the value of term TERM was incorrectly set. The bug has been fixed in this release.

Duplicate Library Routine

In Release 3.2, the _doprnt routine appears twice in /lib/c.a.

The bug has been fixed in this release.

Utility Bugs Fixed

The following bugs were fixed in the language-related utilities.

`tcov` Failed on SIGCHILD

In previous releases, the `tcov` block count statistics coverage were incorrect if the program received ANY signal.

During normal execution, the parent process receives a SIGCHILD when the child process exits; if *any* module in the parent was compiled with the `-a` switch, the parent immediately executes `exit`(2) and writes out the test coverage results.

The bug has been fixed.

`lint` Error Output

Release 3.2's version of `lint` sent its list of complaints to `stderr` rather than `stdout`, as did earlier versions of `lint`.

This bug has been fixed.

Miscellaneous Bugs

Below are miscellaneous language-related bugs that were fixed in Release 3.4.

`pxp` Bug

In Release 3.3, Pascal included a new extension implementing public and private procedure declarations. However, `pxp` was not concurrently enhanced to recognize the keywords `private` and `public`, flagging them as syntax errors in a warning message.

The bug has been fixed in this release.

Compiler "Invalid" Options Bug

In previous releases, some invalid combinations of floating-point options were incorrectly flagged as warnings, rather than fatal errors. These combinations were `-fsky` with `-m68020` and `-ffpa` with `-m68010`.

The bug has been fixed in this release.

Double-Precision Transcendental Bug

In Release 3.2, double-precision trigonometric functions produced incorrect results for certain multiples of $\pi/4$ when using the `-fsoft` or `-fsky` options. The problem does not occur with either `-f68881` or `-ffpa`.

6.2. Graphics Bug Fixes

This section describes bugs that have been fixed in the *Pixrect*, *SunCGI* and *SunCore* graphics software.

Pixrect Bug Fixes

This section describes the bug fixes for the *Pixrect* library.

File I/O

The following subsections describe bug fixes to the file I/O facilities of the *Pixrect* graphics library.

Pixrect File I/O Rewrite

In Release 3.4, the pixrect file I/O facilities have been completely rewritten. A number of bugs, including memory, file descriptor, and process leaks have been fixed. Chapter 7 of this manual contains a new version of the *File I/O* chapter from the *Pixrect Reference Manual* that reflects changes to the pixrect I/O facilities.

Type Declarations Added

In Release 3.4, type declarations of the file I/O functions have been added to the include file <pixrect/pr_io.h>. It is no longer necessary for a *Pixrect* application to declare them explicitly.

pr_load

In Release 3.4, it is no longer necessary for *Pixrect* applications to determine the colormap type of a rasterfile in order to load the file's colormap with pr_load or pr_load_colormap. These functions can automatically load any valid colormap.

RMT_RAW Colormaps

In Release 3.4, colormaps of the RMT_RAW type can be dumped and loaded; they are treated as a single array of bytes. In previous releases, the RMT_RAW type was documented and defined but unimplemented.

Writing to a Rasterfile

In Release 3.4, pr_dump, pr_dump_header, pr_dump_image and pr_dump_init properly handle any type of pixrect. In previous releases, it was necessary to set the *copy_flag* argument passed to pr_dump or pr_dump_init if there was any possibility that the pixrect being dumped would be something other than a primary memory pixrect.

RT_BYTE_ENCODED Rasterfiles

1) In Release 3.4, the run length encoding and decoding functions used to implement the RT_BYTE_ENCODED rasterfile type no longer operate in place. Using pr_dump to write an RT_BYTE_ENCODED rasterfile will always succeed and will not destroy the input pixrect, even if the *copy_flag* argument is not set.

2) In Release 3.4, pr_dump_image can write RT_BYTE_ENCODED rasterfiles. In previous releases, pr_dump_image could only write RT_OLD and RT_STANDARD formats.

3) In Release 3.4, pr_load no longer fails on some correctly encoded input files.

Rasterfile Filters

1) In Release 3.4, the directories in the user's $PATH environment variable are searched for all rasterfile filters. In previous releases, all filters except convert.65535 had to be installed in the directory /usr/lib/rasfilters, and convert.65535 would only be found in the process's current directory.

2) In Release 3.4, the filter code waits indefinitely for the filter process to exit. This prevents race conditions and the accumulation of zombie processes, but may cause *Pixrect* applications to mysteriously hang if undebugged rasterfile filters are installed in /usr/lib/rasfilters or other shared directories.

lint Library

The following section describes bug fixes to the lint(1) facilities of the *Pixrect* graphics library. You should rerun lint(1) on all *Pixrect*-based applications.

1) In Release 3.4, the *Pixrect* lint(1) libraries (llib-lpixrect and llib-lpixrect.ln in /usr/lib/lint) and the header file <pixrect/pixrect.h> have been improved so that the types of arguments to all pixrect functions and macros will be checked.

2) The arguments to pf_text in the lint(1) library have been changed to match the documented calling sequence.

3) Errors in the arguments for pf_textbatch and pr_load_std_image have been corrected.

4) New entries have been added for the GP support functions gp1_alloc, gp1_post, and gp1_sync, and for the new text functions pr_text, pr_ttext, prs_text, and prs_ttext.

Macros

The following section describes bug fixes to macros in the *Pixrect* graphics library.

1) In Release 3.4, the prs_replrop macro was added to the <pixrect/pixrect.h> include file. In previous releases, this macro was documented but not present.

2) A spurious semicolon following the declaration of prs_destroy was deleted.

3) pr_close is a macro as documented, instead of a function, as it was in previous releases.

rop_fastloop Macro

The rop_fastloop macro in the include file <pixrect/pr_util.h> is conditionally defined to generate an unrolled loop only on 68010 systems, and the supporting cases8 macro is only defined in that case. Rolled loops usually provide better performance on systems with an instruction cache, including all 68020 systems.

Memory Pixrects

The following sections describe bug fixes to the memory pixrect facilities of the *Pixrect* graphics library.

Private Data Format for Memory Pixrects

A new memory pixrect private data format has been defined, which includes a bit plane mask to support the cgfour(4S) color memory frame buffer. The document *Release 3.2 Manual for the Sun Workstation* contains a more detailed description of the cgfour(4S) frame buffer. All memory pixrect functions have been modified to support this new format, which is defined by struct mprp_data in <pixrect/memvar.h>. The plane mask is normally accessed with pr_putattributes and pr_getattributes. Most pixrect client programs will not have any reason to directly create memory pixrects with plane masks. But if this is necessary for some special purpose, the function in Figure 6-1 should be used.

Figure 6-1 *Memory Pixrect Program Example*

```
#include <sys/types.h>
#include <pixrect/pixrect.h>
#include <pixrect/pr_util.h>
#incude <pixrect/memvar.h>

Pixrect *mem_create_with_planemask(w, h, depth)
int w, h depth;
{
    Pixrect *pr;
    struct mprp_data *mprd;

    /*
     * Create a normal memory pixrect with no plane mask,
     * replace its private data (struct mpr_data) with
     * a struct mprp_data, set the plane mask flag,
     * and initialize the plane mask.
     */
    if (pr = mem_create(w, h, depth))
        if (mprd = alloctype(struct mprp_data)) {
            mprd->mpr = *mpr_d(pr);
            free(mpr_d(pr));
            pr->pr_data = (caddr_t) mprd;
            mprd->mpr.md_flags |= MP_PLANEMASK;
            mprd->planes = ~0;
        } else {
            pr_destroy(pr);
            pr = 0;
        }
    return pr;
}
```

8, 16, and 32-bit Memory Pixrect Support

In Release 3.4, support for memory pixrects with a depth other than 1-bit has been improved. All pixrect functions except pr_vector support 16 and 32-bit memory pixrects. pr_line can be used to draw vectors in these pixrects.

pr_vector

1) In Release 3.4, vectors drawn into memory pixrects with pr_vector properly wrap polygons drawn with pr_polygon_2.

2) In Release 3.4, drawing vertical vectors into an 8-bit memory pixrect is reliable. In previous releases, pr_vector sometimes scribbled on random memory, often causing a segmentation violation.

pr_get

In Release 3.4, pr_get returns the correct result when applied to a reverse video memory pixrect.

`pr_put`

1) In Release 3.4, using `pr_put` to write to a pixel in a memory pixrect correctly truncates its *value* argument as documented. In previous releases, any non-zero argument caused a *value* of 1 to be written to a 1-bit memory pixrect.

2) In Release 3.4, using the `pr_put` function with the *value* argument set to 0 on a reverse video memory pixrect works properly. In Release 3.2, *value* was always treated as 1.

`pr_rop`

1) In Release 3.4, the semantics of operations involving a 1-bit source pixrect and an 8, 16, or 32-bit destination pixrect are consistent with the documentation and the behavior of `cgone` pixrects.

2) In Release 3.4, use of the `op` PIX_NOT(PIX_DST) with a non-NULL source pixrect is handled properly. In previous releases, it was treated as PIX_DST.

3) In Release 3.4, operations involving small, 8-bit memory pixrects are reliable.

4) In Release 3.2, when `pr_rop` was called with a 1-bit memory pixrect destination and NULL source, the least significant bit of the color encoded in the `op` was used to determine whether the source value should be 0 or 1. In Release 3.4, any non-zero color is treated as 1. This is consistent with pre-3.2 releases and the other memory pixrect functions.

5) In Release 3.4, using the `pr_rop` function on a 1-bit source and a narrow (16 bits-per-line) 8 or 16-bit memory pixrect destination is reliable.

Text

The following sections describe bug fixes to the text facilities of the *Pixrect* graphics library.

Memory Leaks with `pf_open`

In previous releases, a memory leak affecting repeated calls to `pf_open` and `pf_close` has been fixed. In Release 3.4, `pf_close` releases all resources allocated by `pf_open`.

`pr_close`

In previous releases, if `pf_close` was given a font argument which referred to a resident (compiled in) font that was not the default font, it would try to deallocate that font's data. This caused segmentation violations some time. In Release 3.4, any valid font can be passed to `pf_close`.

Opening `vfont` (5) Fonts

In Release 3.4, when `pf_open` loads a `vfont`(5) file, it normally sets the default character width to the width of (lowercase) `a` and the default character height to 1.5 times the height of (uppercase) `A` above the baseline. If `a` or `A` are not present in the font or have zero *width* and *up* values respectively, the `vfont`(5) header *maxx* and *maxy* values are used instead. In previous releases, `pf_open` would fail if `a` and `A` were present but had zero *width* or *up* values, making it impossible to load some valid fonts (primarily symbol and rotated fonts).

pf_ttext String Lengths

In Release 3.4, the pf_ttext function properly displays text strings of any length. In previous releases, the 81st and subsequent characters were positioned incorrectly.

Miscellaneous

The following sections describe miscellaneous bug fixes to the *Pixrect* graphics library.

pr_getcolormap with bwone Pixrects

In previous releases, pr_getcolormap returned incorrect values when applied to a bwone pixrect. In Release 3.4, this bug has been fixed and the colormap behavior of bwone pixrects is identical to that of bwtwo pixrects.

pr_batchrop

In Release 3.4, any valid pixrect can be passed as a source pixrect to pr_batchrop. In previous releases, use of anything other than a primary 1-bit memory pixrect caused undefined results in some cases.

Global Definitions Removed from pr_traprop

In previous releases, several utility functions were inadvertently declared globally in the source file for pr_traprop, sometimes causing collisions with user-defined names when this function was linked into a pixrect program. The most notable example was a function called init. In Release 3.4, the utility functions are static.

pr_vector

In previous releases, pr_vector ignored the most significant 16 bits of the coordinates passed to it. In Release 3.4, it performs clipping calculations on the full 32-bit values. (Most other pixrect functions will not function properly when given coordinates that cannot be represented in 16 bits.)

pr_polypoint

In Release 3.4, the pr_polypoint function produces correct results on bwone pixrects for all op values. This function is described in the document *Release 3.2 Manual for the Sun Workstation.*

pr_polygon_2

1) In Release 3.4, the pr_polygon_2 function replicates the source pixrect as necessary to tile an entire polygon. It is no longer necessary for the source pixrect to be as large as the destination area covered by a polygon drawn by pr_polygon_2.

2) In Release 3.4, when pr_polygon_2 returns an error, it will consistently be PIX_ERR rather than various small integer values.

3) In Release 3.4, the pr_polygon_2 function produces correct results when the destination is a GP pixrect and the source is not a 1-bit memory pixrect. In Release 3.2, pixels on the right edge of the destination were not drawn reliably, and other types of source pixrects gave indeterminate results.

pr_replrop

The pr_replrop function was rewritten to handle all combinations of destination and source coordinates properly, even if clipping is disabled, and to accept NULL source pixrects.

pr_rop

In Release 3.4, the semantics of operations with pr_rop involving a 1-bit source pixrect and a cgtwo or gpone destination pixrect are consistent with the pixrect documentation and the behavior of cgone and memory pixrects. In previous releases, each destination pixel was operated on using a value of either 0 or 255 (corresponding to 0 and 1 source pixrect values), then logically AND-ed with the color encoded in the op.

pr_stencil

1) In previous releases, pr_stencil operations from an 8-bit memory source pixrect to a cgtwo or gpone destination pixrect did not always correctly draw pixels on the right edge of the destination pixrect. This bug has been fixed in Release 3.4.

2) In previous releases, pr_stencil did not bother to synchronize with the GP when the source was a memory pixrect, sometimes causing spectacular screen corruption. This bug has been fixed in Release 3.4.

pr_line

1) In Release 3.2, pr_line and pr_polyline could overwrite an internal buffer in rare circumstances when drawing horizontal un-clipped textured vectors. This bug has been fixed in Release 3.4.

2) In previous releases, attempting to draw clipped textured polylines with the *balance* option set produced improper textures. This bug has been fixed in Release 3.4.

3) In previous releases, single segment (solid) textured vectors were not always drawn. This bug has been fixed in Release 3.4.

Frame Buffer Device Drivers

The following sections describe bug fixes to the frame buffer device drivers.

gpone (4S)

In Release 3.4, the gpone(4S) driver zeros the static blocks handed out by the GP1IO_GET_STATIC_BLOCK ioctl.

FBIOGVIDEO ioctl

In Release 3.4, the FBIOGVIDEO ioctl always returns the correct video enable status for all frame buffers except the bwone(4S). In Release 3.2, it was also possible for the software state to disagree with the hardware state for bwtwo(4S) and cgfour(4S) frame buffers.

cgfour 4S Support for the Sun-3/60 Frame Buffer

In Release 3.4, the cgfour pixrect driver includes support for the Sun-3/60 color frame buffer.

Graphics Utilities

This section describes the bug fixes for various graphics utility programs.

rasfilter8to1 (1)

1) In previous versions, the right edge of the output image from rasfilter8to1(1) appeared incorrectly. This has been fixed in Release 3.4.

2) In previous releases, rasfilter8to1(1) would generate an incorrect output file if the input file width was odd. In Release 3.4, this bug has been fixed.

rastrepl(1) and RT_BYTE_ENCODED Rasterfiles

In previous releases, rastrepl(1) could not read 8-bit deep RT_BYTE_ENCODED rasterfiles. This bug has been fixed in Release 3.4.

SunCGI Bug Fixes

Release 3.4 contains the following bug fixes for the *SunCGI* software.

Resizing Viewports

In previous releases, if a viewport was made smaller than a clip rectangle, and then made larger, a smaller viewport would remain. This has been fixed in Release 3.4.

NOCLIP

Warning: When clipping is disabled in *SunCGI*, pixwin clipping is also disabled. Therefore, a *SunCGI* application can write outside the display area and produce unpredictable results, including system errors.

Clear View Surface

When using `clear_view_surface` with `clear_control` set to `clip_rectangle(8000, 9000, 10000, 11000)`, *SunCGI* would draw a large rectangle instead of clearing the specified area. This has been fixed in Release 3.4.

Cvwsurf

1) In Release 3.0, it was possible to create a new View Surface Tool by setting the `flags` field of the `Cvwsurf` structure. A bug was introduced in Release 3.2 that prevented this. This has been fixed in Release 3.4.

2) In previous releases, *SunCGI* would not allow an application to set the `ptr` field of the `Cvwsurf` structure to one of the advertised strings. This has been fixed in Release 3.4.

Quitting a View Surface Tool

After opening a view surface tool with a system call, the view surface tool would not close without selecting the Quit item from the *Frame* menu. This has been fixed in Release 3.4.

Pixwin Output

Previously, *SunCGI* relied on an asynchronous event (a SIGWINCH signal), to gain access to the output pixwin, which sometimes caused a core dump. This has been fixed in Release 3.4.

CGIPW and SunView

1) A mismatch between *CGIPW* and *SunView* use of pixwin regions prevented the integration of *CGIPW* on a *SunView* canvas that had scrollbars. This has been fixed in Release 3.4.

2) Problems prevented the proper use of *CGIPW* on a *SunView* canvas with a retained backing larger than the visible screen window area. This has been fixed in Release 3.4.

Request Input

In previous releases, `request_input` and `await_event` did wait-polling while awaiting an input trigger or timeout. This would cause intense CPU resource use during these calls. In Release 3.4, `request_input` and `await_event` use the `select(2)` mechanism to block, thus freeing the CPU until either the input trigger fires, or the timeout expires.

`cfmksizespecmode`

The FORTRAN library was missing the `cfmkspecmode` function cited in Appendix G of the *SunCGI Reference Manual.* This function has been replaced by `cfmksizespecmode` in Release 3.4.

GP View Surfaces

When more than one *SunCGI* view surface was opened on a GP surface, a GP-specific attribute structure was freed, affecting subsequent view surfaces. This bug affected blanket windows and `view_surface` tools, but did not occur in *CGIPW* mode. This has been fixed in Release 3.4.

Rectangle Perimeters

In Release 3.2, perimeters drawn with a non-zero window offset in *x* or *y* by `rectangle` appeared at locations other than those specified. This has been fixed in Release 3.4.

Arc

Arcs having common *y*-value endpoints would sometimes invert the arc from the intended side of the points. Arcs are drawn correctly in Release 3.4.

Freed Memory with Await Event

The `await_event` function returned a pointer to freed memory, thus causing problems when making calls to input devices such as IC_STRING, IC_STROKE, IC_LOCATOR, and IC_PICK. This has been fixed in Release 3.4.

Multiple Character Keyboard Events in CGIPW

In *CGIPW* mode, multiple-character keyboard events were not immediately delivered when using Canvases. This has been fixed in Release 3.4.

Inquire Device Identification

The `inquire_device_identification` function returned a null string instead of a device type. This has been fixed in Release 3.4.

View Surface Table

CGIPW functions used in conjunction with output primitives and attribute settings in Release 3.4 more accurately and efficiently search the view surface table.

Obsolete Code Removed

In Release 3.4, obsolete *SunCGI* code from previous releases has been removed to improve maintenance.

Close View Surface

The `close_vws` function did not change from VSAC (at least one view surface active) or VSOP (at least one view surface open) to CGOP (CGI open, no view surfaces open) when the last view surface was freed.

Deactivate View Surface

In Release 3.4, the `deactivate_vws` function allows the view surface mode to change, so that a deactivated view surface will display output primitives if there is only one active workstation. Likewise, the `activate_vws` function now changes the view surface mode, so that an *open* view surface will display output primitives if it is the only active workstation.

CGIPW Validity Check

In Relase 3.4, *SunCGI* checks the validity of *CGIPW* function descriptors to avoid segmentation violations and inadvertent writes to memory.

CGIPW Set VDC Extent

The `cgipw_set_vdc_extent` function worked correctly with only one view surface in use. In Release 3.4, it works with more than one surface in use.

Scaling

Scaling failed to vary line width, perimeter width, and marker sizes when using VDC coordinates and multiple view surfaces. This has been fixed in Release 3.4.

Inquire Text Extent

The `inquire_text_extent` function attempted to display output in inactive surface areas. This has been fixed in Release 3.4.

Transparent Text

In Release 3.4, text transparency now behaves as it did in Release 3.0.

FORTRAN Input

In previous releases, use of any of *SunCGI* input function through a FORTRAN compatibility function would result in a segment violation. In Release 3.4, the FORTRAN input mechanisms work properly.

VALUATOR Input with Await Event

In previous releases, the receipt of the first input event when a VALUATOR logical input device is in EVENT mode disables the LID. The LID will be left in EVENT mode in Release 3.4.

Ignored Input with Await Event

In previous releases, `await_event` would return the initial value for CHOICE, VALUATOR, or PICK logical input devices. This has been fixed in Release 3.4.

Bad Data Structure Error Code with Await Event

In previous releases, when an application passed a `Cinrep` structure with an invalid `measure` field to `await_event`, *SunCGI* would return the error code EINQOVFL (Input Queue Overflow) instead of EBADDATA. This has been fixed in Release 3.4.

SunCore Bug Fixes

Release 3.4 contains the following bug fixes for the *SunCore* software.

Fat Vectors

1) In previous releases, the thickness of fat vectors drawn with *SunCore* would vary with the slope of the vector. While it is not possible to get exactly the same line width on raster displays, in Release 3.4 vectors have significantly more uniform widths regardless of the slope.

2) Fat vectors drawn on GP surfaces with *SunCore* would not be completely removed when erased. This has been fixed in Release 3.4.

`raster` Structure

The `raster` structure used in `put_raster` and `get_raster` is now defined in `<usercore.h>`. In Release 3.4, a *SunCore* application does not have to explicitly define this structure.

`put_raster`

In previous releases, `put_raster` in *SunCore* would invert the raster image on Sun-1 monochrome displays. This has been fixed in Release 3.4.

Clipped Vectors with GP View Surfaces

In *SunCore* on the 3.2 release, clipped vectors did not show up on GP view surfaces. This problem in the GP microcode has been fixed in Release 3.4.

Polygon Vertex Limit

In previous releases, polygons in *SunCore* were limited to 100 vertices. This limit has been changed to allow up to 200 vertices in Release 3.4.

Backspace Key

In previous releases, `await_keyboard` would use the CTRL-H as backspace, which would have differed from the previous terminal setting. In Release 3.4, *SunCore* uses the current terminal setting for the backspace character.

6.3. SunView Bug Fixes

Release 3.4 has several bug fixes made to SunView since Release 3.2. This section first describes bug fixes that are visible to users of the system, followed by bug fixes that affect SunView programmers.

Defaults, `.*rc` Files, and Filters

Various bug fixes have been made in these categories, so you may find that something that did not work as expected before now works, or possibly vice versa. It is worth checking your `~/.*rc` files.

`.textswrc` Processes Shell Metacharacters

In the `~/.textswrc` file itself, in the command line following the `FILTER` line, regular shell metacharacters such as `$`, ", and ' are now processed by the shell. This shell is the one defined by your `SHELL` environment variable (usually `/bin/csh`); if the environment variable does not exist the shell used is `/bin/sh`. For example,

```
FILTER  R13
/bin/echo $HOME
```

in previous releases would put the actual characters "`$HOME`" in the text subwindow when you press R13; now it inserts the path to your home directory as it would if you typed

```
/bin/echo $HOME to your shell.
```

CAUTION

This means shell metacharacters you were using before must be escaped following the normal "\" conventions for your default shell. The sample text subwindow filters file, `/usr/lib/.textswrc`, has been updated; you can use this as a guide.

Fixes to Input Filters

The input filters `capitalize` and `insert_brackets` (documented in the `textedit_filters(1)` manual page) have been improved. You usually use these filters by assigning them to function keys with the file `~/.textswrc`, then invoking them in text subwindows.

`capitalize` no longer blanks out numeric characters.

Both `capitalize` and `insert_brackets` can handle larger selections. In order to do this, they process standard input in a temporary file, called `/tmp/cap`*ProcessId* or `/tmp/ins`*ProcessId*. If the temporary file cannot be opened or deleted, each prints an error and copies standard input to standard output, so that if the selection is pending delete, it does not just disappear.

Previously, `insert_brackets` would dump core if it was not invoked with the right arguments. It now prints out

```
Usage: insert_brackets leftbracket rightbracket
```

on standard error (the console, if you invoke it in a `textedit` started from the root menu) and writes standard input to standard output unchanged. (That is, in a text subwindow, the pending delete selection is replaced by itself.)

Before Release 3.4, `insert_brackets` did not recognize all the escape character conventions in C. Now it supports the full rules (in *The C Programming Language* by Kernighan and Ritchie) for inserting control characters into strings; specifically, `\n`, `\t`, `\b`, `\r`, `\f`, `\\`, `\'`, and \octal_number are all valid escape sequences. *Check your* `~/.textswrc` *files to see if you need to modify them to work under this scheme.* For example, the old filter definition

```
R1      FILTER
insert_brackets \fI  \fP
```

becomes

```
R1      FILTER
insert_brackets "\\fI"  "\\fP"
```

The quotes cause the shell to leave the backslashes alone. `insert_brackets` will then change each pair of backslashes into a single backslash.

`.ttyswrc` String Changes

The similar file to map function keys in tty subwindows is `~/.ttyswrc`. Keys can send longer strings, now up to 1024 characters.

In previous releases, you had to preface the : character with a \ to output it; in 3.4, you can output colons without "quoting" them this way. You specify other special characters in strings in the `~/.ttyswrc` file in the same way you do for strings in `termcap`; see `termcap`(5) for details.

Blank lines in `~/.ttyswrc`

In Release 3.2, blank lines in a `.ttyswrc` file caused

```
unknown command
```

error messages to be printed when a tty subwindow was created. Now, blank lines are treated as comment lines, as they were in 3.0.

Mail Aliases in `defaultsedit`

In previous releases, if your `~/.mailrc` file included long aliases, `defaultsedit` would not read them in successfully. So, when you saved your defaults, the long aliases were lost. This bug has been fixed in 3.4. Note, however, that the problem can recur in client programs that use the SunView defaults database and are not relinked with the new SunView libraries.

.suntools Can Accept Blank Lines

In previous releases, blank lines in suntools would interfere with the automatic tool startup suntools provides. You can now have blank lines in your suntools file.

Other Bug Fixes Visible to the User

Cursor Correct on Sun-3/110

In Release 3.2, when opening an iconic window on the Sun-3/110LC display, the screen area under the cursor was sometimes erroneously set to display the color planes instead of the overlay plane.

lockscreen -e on the Sun-3/110LC Color Display

lockscreen(1) with its -e option no longer crashes the system.

'UnZoom' when Iconic

Selecting 'Resize ⇒ UnZoom' from the frame menu did not to work for iconic tools. This has been fixed in Release 3.4.

Characters Lost on Multi-character Input

In previous releases, if you held down keys mapped into multiple characters (such as the arrow keys[5]), some of those characters could sometimes be lost. Now, the kernel's input event queue expands to cope with the extra events, and shrinks down to a smaller size when the system has caught up.

Bug Fixes in cmdtool

In addition to the new functionality added to cmdtool (see *Improvements to* cmdtool above), there are several bug fixes to cmdtool in 3.4.

'Reset' in cmdtool

After selecting the 'Reset' menu item in cmdtool, you were unable to use the line editing keys (*Edit_back_char*, *Edit_back_word*, and *Edit_back_line* in the *Text* category of defaultsedit) until you typed a new line. This has been fixed.

Typing Ahead

Typing characters faster than the system was displaying them would create a false input buffer overflow condition. This has been fixed.

SIGIO Signals in cmdtool

The SIGIO signal is now delivered to the process running in cmdtool, even if that process has not emptied its input stream.

Bug Fixes in textedit and the Text Subwindow

In addition to the new features added to the text subwindow, (see *Text Subwindow Enhancements* above), there are several bug fixes to textedit, and the text subwindow in general, in Release 3.4.

Note: You may need to change your suntools or rootmenu files along with other aliases to reflect this bug fix.

[5] The arrow keys can be mapped to emit SunView function key codes instead of cursor motion codes by setting *Arrow_Keys* in the *Input* category of defaultsedit to *False*.

`textedit` Width Set Correctly

In prior releases, the `-Ww` *nnn* flag to `textedit` would not take the size of the scrollbar into account, so that to make, say, a 60-column window you would have to set the width slightly larger. Now the width you supply is the number of columns in the typable portion of the text subwindow.

User-Supplied Icon Label

`textedit` usually changes its icon's label to show the file being edited and its status. In 3.4, you can supply your own icon label using the standard `-WL`*label* frame command line option.

'Store to named file' and 'Get from file' with Null Selection

'Store to named file' in the text subwindow menu requires you to select the file name to store to first. In previous releases, if you chose 'Store to named file' (or its variants 'Store & Quit' and 'Close & Store') when there was no selection, it would sometimes save the file to a bogus filename. Now it prints a detailed message asking you to select the desired filename. The same message appears if you choose 'Get from file' with a null selection.

'Get from file' with Non-existent Filename

The text subwindow package no longer prints

```
    Stuff from file failed due to INTERNAL ERROR
```

when it cannot find the filename selected for 'Get from file.'

Scrollbar Bubble More Accurate

If the text subwindow's contents ended without a newline, the scrollbar bubble's position would indicate text remained below the window's contents, even if the window showed the end of the file.

Also, the scrollbar bubble's position is now updated after Delete, Undo, and Get operations.

Changing 'Line break' Mode

The caret now positions correctly after changing between 'Clip lines' and 'Wrap at character' modes.

Text Subwindow Loses Track of Selection

It was possible to confuse the Selection Service during a secondary Get by letting up on the Get key before releasing the mouse button; this has been fixed.

Highlighting Selections

In some cases, selecting Tab characters along with other characters would damage the screen image of characters; this is fixed in Release 3.4.

Programs Exiting in `shelltool` and `cmdtool`

The message printed out when the program running in a `shelltool` or `cmdtool` exited with non-zero status used to be the following:

```
child of tty exited with return code nn
```

or

```
child of cmdtool exited with return code nn
```

The message has been changed to be more informative, either

```
A tty window has exited because its child exited.
Its child's process id was pid and it exited with return code nn
```

or

```
A command window has exited because its child exited.
Its child's process id was pid and it exited with return code nn
```

Not Enough Swap Space in `shelltool` and `cmdtool`

Sometimes you can run out of swap space so that `shelltool` or `cmdtool` starts up, but it cannot start the program running in itself (usually this is a shell). In previous releases, the `shelltool` or `cmdtool` would just hang in this situation. Now the tool remains active and prints

```
 Out of swap space.  Cannot continue.
```

You can then 'Quit' the tool from its frame menu, and kill other processes until you have enough memory to try again.

Fixes to TTY Subwindows

These bug fixes apply to programs that use tty subwindows, such as `shelltool` and `cmdtool` in tty mode.

`clear` in a TTY Subwindow
: In previous releases, when a tty subwindow was cleared by running `clear(1)`, it did not reset its count of how many lines are currently displayed in itself, leading to problems when 'Page Mode' was enabled in `shelltool`.

Long Input Lines in TTY Subwindows
: An unavoidable kernel constraint makes tty subwindows lock up if you insert or type a line longer than 255 characters into them. In Release 3.2, a 'Flush' menu item was added to the tty subwindow menu, allowing you to flush the entire input line. In Release 3.4, in addition to the menu item, a warning message pops up as you type. You can choose to flush the input queue or ignore the error condition and work in another window.

Highlighting Word Selections
: Word selection in reverse video fields in `shelltool` highlights correctly in Release 3.4.

Security Hole in `lockscreen` Fixed

In previous releases, the password control in `lockscreen(1)` could be circumvented by using the Open (usually L5) function key to pop up a window above it briefly. The key is now disabled in `lockscreen`'s input mask.

Miscellaneous Fixes to Tools

Changes in `defaultsedit`

`defaultsedit` would sometimes lose changes made to the database when changing between categories. It now prompts you to save changes when you switch categories.

Invalid Options for `perfmeter`

In previous releases, if `perfmeter(1)` was invoked with an invalid option, it would never appear on the desktop but invisibly chew up CPU time. It now prints out a 'Usage' string when given an invalid option.

`clock` Test Mode

In Release 3.2, if you ran the `clock` in test mode (`-t`), it would print out

```
Notifier error: Unknown client
```

This has been fixed.

Large Fonts in `chesstool`

The `chesstool` game now works with large fonts.

User Interface Cleanup

The following bugs were fixed to make tools conform with Appendix B of the *SunView Programmer's Guide, Sun User Interface Conventions.*

- The cursor in `defaultsedit` and `mailtool` should change to an hourglass when either tool isn't responding to user input. It now does.
- In `iconedit`, the cursor would change to an hourglass, but the frame header would not change. The frame header now changes to show that `iconedit` is busy.

Bug Fixes Visible to the Programmer

The remainder of this section describes bug fixes visible to the programmer in Release 3.4. Some of the fixes are related to errata and addenda for the SunView programmer's manuals, given in Chapter 7.

Window Geometry Fixes

Several fixes have been made to window sizing and panel item layout. If you have programmed around bugs in window geometry, or if you are relying on anomalous behavior in Release 3.0 or 3.2, you should pay particular attention to this section. Also read the addenda relating to panel layout bugs, *Cautions for Default Panel Item Layout,* in Section 7.7, ''Errata and Addenda for the SunView Programmer's Guide.''

`WIN_ROWS` When There is No Frame Header

The value returned by the `WIN_ROWS` attribute for a frame used to be incorrect if you turned off the frame header (using the `-Wn` frame command line option or by setting `FRAME_SHOW_LABEL` to `FALSE`). This has been fixed.

Panel Choices Too High

In Release 3.2, if the `PANEL_DISPLAY` mode for a panel choice item was `PANEL_CURRENT`, the current choice would be positioned a few pixels too high relative to the label. Release 3.4 restores the correct behavior from 3.0.

''Next'' Panel Item Created Too Low

When creating items in a panel subwindow in Release 3.2, if the position of the next item was not specified when it was created, it would sometimes be located a few pixels lower than the preceding item. This is fixed in 3.4.

Choice Item Positioning

Choice Label baselines did not line up with choices by default. This has been fixed.

Other Fixes Visible to the Programmer

Miscellaneous Text Subwindow Fixes

Scrollbar Bubble Positioning

In previous releases, the TEXTSW_FIRSTand TEXTS_FIRST_LINE attributes did not set the scrollbar bubble. This is fixed in this release. Since TEXTSW_FIRST now sets the scrollbar mark, it makes textsw_normalize_view() unnecessary.

Read-Only Text Subwindow

You can make the file in a text subwindow read-only at the same time that you set the filename; that is, the following works:

```
window_set(textsw,
          TEXTSW_FILE,       filename,
          TEXTSW_READ_ONLY, 1,
          0)
```

Before, you had to do this in two separate operations.

Text Subwindows with Scrollbars

Text subwindows are created with a scrollbar by default. In 3.0, you could not explicitly set the scrollbar for a text subwindow. In 3.2, you could create the scrollbar, but not specify its thickness. Now you can do both, but you must specify the SCROLL_DIRECTION of the scrollbar when you create it — see *Creating Scrollbars* in Section 7.7, ''Errata and Addenda for the SunView Programmer's Guide.''

Also, Release 3.4 has fixed strange behavior that occurred in previous releases if you explicitly created a text subwindow with a horizontal scrollbar.

Destroying the Scrollbar

In previous releases, the text subwindow would not always destroy the old scrollbar and adjust the text painting rect when you specified your own scrollbar.

TEXTSW_RIGHT_MARGIN Works

The attribute TEXTSW_RIGHT_MARGIN now works properly in text subwindows.

Text Subwindow Storage

In previous releases, when a text subwindow was destroyed, it would not free the storage associated with [Again] logging. This has been fixed.

Timer Events in Text Subwindows

The text subwindow package uses timer events to blink the caret. When a text subwindow was destroyed in earlier releases, sometimes the destroy event would be delivered while a timer event was still pending. The blink caret routine would then fail and dump core because the text subwindow no longer existed. The text subwindow now removes the timer when the last text subwindow view is destroyed.

Pop Up Subframe Restrictions

You can use `window_loop()` to display a subframe containing a canvas. Prior to this release, you could only put a panel in subframes displayed in this manner.

These restrictions do not apply if you set `WIN_SHOW` to `TRUE` to display the subframe. `window_loop()` is usually used to prompt the user to acknowledge some event; `WIN_SHOW` is usually used to display secondary information or a *property sheet* while the user continues to work in the main frame.

Other restrictions on `window_loop()` (described in more detail in *Restrictions on Pop Up Windows* in Section 7.7, ''Errata and Addenda for the SunView Programmer's Guide'') remain:

- □ Pop up subframes displayed with `window_loop()` can only have one subwindow.
- □ You still cannot have a text subwindow in a pop up displayed with `window_loop()`.

`WIN_FONT` now Works for TTY Subwindows

The attribute `WIN_FONT` now correctly sets the font of a tty subwindow. `WIN_FONT` *still does not work for frames. You cannot create a text subwindow with a given font; you have to create the text subwindow, then use* `window_set()` *to set the font. Variable width fonts only work for frames, canvases, and text subwindows.*

`event_set_time()`

There should have been a macro to set the timestamp of an event, to complete the `event_set_*(event, *)` set. Now there is, `event_set_time(event, time)`.

`KEY_BOTTOMFIRST` and `KEY_BOTTOMLAST` Defined

In previous releases, the SunView event macro `event_is_key_bottom()` used the nonexistent constants `KEY_BOTTOMFIRST` and `KEY_BOTTOMLAST`. It now uses the existing *vuid* codes `KEY_BOTTOMLEFT` and `KEY_BOTTOMRIGHT`.

Diagonal `pw_copy()` Fix

Diagonal `pw_copy()` calls made two image in previous releases.

`pw_line()` in a Canvas

The routine `pw_line()` would draw in the coordinate space of the canvas subwindow. If the canvas was larger than the subwindow and scrolled, this would produce the wrong effect. `pw_line()` now draws in the coordinate space of the canvas itself.

Click_to_Type in Panels

In *Click_to_Type* mode (in the *SunView* category in `defaultsedit`) you could not set the keyboard focus to a panel with no text items, even though some of the items might have keyboard shortcuts. In Release 3.4, if `PANEL_ACCEPT_KEYSTROKE` is set, the panel will accept the keyboard focus.

Cursor Positioning Escape Sequence in TTY Subwindows

(Escape sequences for the Sun console and tty subwindow are documented in the `console`(4S) manual page.)

Note: The work-around for this bug was to modify the `cm` value in the Sun `termcap` entry. You do not need to apply the work-around any more.

The cursor positioning (`CUP`) escape sequence `ESC[`*#1*`;`*#2*`H` takes character position and line as parameters. If you omit the parameters, they are supposed to default to 1. In previous releases, the tty subwindow would sometimes ''remember'' the previous character position, so that if the character position was not specified, the cursor would not move to the beginning of the line.

Odd-Sized Icons

In previous releases, if you supplied a tool with an icon that was not the standard size (64 x 64 pixels) the icon would not display properly. This has now been fixed, so that you can use smaller or larger icons. However, icons still initially appear aligned on an invisible 64-pixel grid unless you specify their position with the `-WP`*x y* frame command line argument. `iconedit` produces only 64 x 64 or 16 x 16 patterns; you have to edit its output by hand for other sizes.

Internal Name Changes

The names of some routines used internally have been changed to reduce the likelihood of conflict with client names. Unless you have SunView source code, you should not use these internal names.

`draw_box` renamed

The routine `draw_box()` in `libsuntool` has been replaced by `_tool_draw_box()` to avoid possible name conflicts.

Changes to Window Adjustment Cursor Names

The names of some of the cursors used internally by the packages, such as `move_cursor`, could easily conflict with client programs. The internal names have been changed to `wmgr_*` to avoid conflict.

Example Program `filer`

The (optional) example program `/usr/src/sun/suntool/examples/filer.c` would not compile in Release 3.2 because the routine `confirm_yes` was undefined. `filer.c` was meant to be accompanied by a simple confirmer package, `confirm.c` that includes this routine. The confirmer code was printed at the end of `filer.c` in Appendix B of the 3.2 *SunView Programmer's Guide.*

The file `/usr/src/sun/suntool/examples/confirm.c` is included in the optional software for 3.4, along with a simple `Makefile` for the examples.

Other Fixes to Example Programs

The `seln_demo.c` and `tty_io.c` example programs (part of the optional software) from 3.2 did not work properly. They both called `window_fit()` to resize their frames, but did not explicitly give a size for their subwindows. This caused them to shrink to a very small size, as explained in the addenda *Fitting Frames Around Subwindows* in Section 7.7, ''Errata and Addenda for the SunView Programmer's Guide.'' Revised versions of the programs are included in 3.4.

The program `typein.c` has been revised to use the new tty subwindow attributes described in Section 5.2.6, *New TTY Subwindow Attributes*, above. The program is printed out in full in *Revised* `typein.c` *Example Program*, `in`

Section 7.7, ''Errata and Addenda for the SunView Programmer's Guide.''

6.4. SunOS Bug Fixes

This section contains bug fixes that were made to the kernel, utilities, and shells.

Installation and Configuration-related Bugs

The following bugs, related to booting, configuring the kernel, and installing the operating system, were fixed in this release.

Missing Sun-3/110 Configuration Files

The Sun-3/110 configuration files /usr/sys/conf/SDST110 and /usr/sys/conf/ND110 were not included in Release 3.2 These files are included in Release 3.4. (Refer to Chapter 3, *Reconfiguring Your Kernel*, for more information.)

/sys/sundev/mti_conf.c File Missing

The file /sys/sundev/mti_conf.c was not included in Release 3.2. It is included in Release 3.4.

Remote Install Problems

During remote installations of Release 3.3, the upgrade script redirected the dd command's stderr to standard output, which was then piped as standard input to tar. Thus, dd's status messages were getting mixed up with the file transfer.

Sun-3/50-4 Tape Cartridge Hangs

In Release 3.3, Sun 3/50-4s with shoe boxes containing Micropolis 1324 disks and Archive cartridge tape drives (serial 9050 B) experienced tape drive hangs. This occurred after upgrade and reconfiguration when users tried to access the tape drive with tar, and rewind.

This has been fixed in Release 3.4.

bzero Routine Could Not Handle Large MAXUSERS

The kernel's bzero routine did not handle more than 256 Kbytes correctly. As a result, systems with Xylogics disks could get the following kernel panic message:

```
panic:memall intrans|want
```

when MAXUSERS was configured above a certain amount. For example, a Sun-3/160 would fail when MAXUSERS was set to 12 or above. Sun-3/260s and Sun-3/180s would fail when MAXUSERS was set to 24 or above.

This condition was happening because setting MAXUSERS too large caused the kernel to call its own bzero routine with greater than 256 Kbytes. Previously, the kernel's bzero could not handle this size correctly.

In Release 3.4, the kernel's bzero routine works correctly with sizes greater than 256 Kbytes. You can now set MAXUSERS to any number required without problems resulting from bzero's incorrect handling of the larger request.

Sun-3 Kernel Could Not Be Built Without Sun-3/260 Defined

In Release 3.2, you could not build a Sun-3 kernel from source code unless SUN3_260 was defined in the configuration file. In this case, you would receive a message that vac_flushall() was undefined.

This problem has been fixed in Release 3.4.

Sun-3s with Early PROMs Had Rebooting Problems

If you added large amounts of memory to a Sun-3 with a pre-1.6 PROMs, it occasionally failed to reboot. As a result, you had to power cycle the system after it hung due to this problem.

This problem has been fixed in Release 3.4.

3.3UPGRADE Did Not Work on Systems with 3.2EXPORT

Systems running 3.2EXPORT could not upgrade to Release 3.3. Any attempts to do so resulted in an error message and termination of the upgrade. This was because the upgrade script inadvertently did not accept 3.2EXPORT as a legitimate release.

The installation script has been fixed. You can upgrade 3.2EXPORT to Release 3.4.

Standalone Programs Did Not Exit Correctly

The Release 3.2 boot program caused standalone programs, such as diag, to behave incorrectly when they exited. For example, if you entered

```
    diag> quit
```

this restarted diag, rather than quit the program, as expected.

This bug has been fixed in 3.4. Now standalone programs correctly return to the PROM monitor when exited.

Kernel-related Bugs

The following additional kernel-related bugs were fixed in Release 3.4

Cursor Motion Not Coordinated with Mouse Motion

In earlier releases, the cursor sometimes did not properly track mouse motion. For example, if you slowly moved the mouse in a horizontal direction, the cursor would move diagonally, from the lower left to the upper right of the screen. If you slowly moved the cursor in a vertical direction, the cursor once again would move diagonally.

This behavior would occasionally occur once the window system was initialized. Additionally, this resulted in the mouse buttons not being recognized unless you moved the mouse while pressing a button. Previously, you had to unplug and replug the mouse, or, if necessary, reboot the system to clear the problem. In Release 3.4, the cursor tracks the mouse correctly at all times.

Memory Device Driver Bug

In Release 3.2, a bug was introduced into the memory device driver. Though invisible to the users, this bug was fixed in Release 3.4 to improve memory device driver efficiency.

Bug in kadb

If you used a standalone sun-3 server, you had to manually invoke the kernel debugger kadb, because the PROM monitor did not run it automatically. Instructions for doing this are in the kadb(8S) manual pages in Appendix C.

Error Reporting Problems with SCSI Tape Drives

Sun 3/50s with MT02 SCSI tape drives were experiencing problems when you tried to append a file to a 1/4-inch tape. The application doing the write would seem to succeed, but actually, the tape would not move. Thus, the write would not happen. Furthermore, you would not receive a message to that affect. Instead, you occasionally received the following internal error message on the console:

```
st0: stintr: sense f0 0 5 0 0 0 6 3 33 0 0
```

This problem has been fixed in Release 3.4. The append now succeeds, and no error message is issued.

VPC-2200 Parallel Printer/Plotter Interface Problems

In prior releases, systems with the VPC-2200 parallel printer/plotter interface were experiencing compatibility problems with other drivers. The VPC-2200 was allocating an odd number of bytes of direct virtual memory access space. This caused problems for other drivers that expect their byte allocations on even boundaries. The SCSI driver was modified in Release 3.3 to handle this situation.

In Release 3.4, the VPC-2200 driver was modified so that it would allocate an even number of bytes. This ensures problem-free operation for drivers requiring even byte alignments that allocate memory after the VPC.

Daisy Chained Shoe Boxes on Sun-3/50 Sometimes Hung

In Release 3.3, Sun-3/50s with two daisy chained shoeboxes would not boot if the configuration had a 141 Mbyte drive (ESDI) designated as Unit 0 and a 71 Mbyte drive (ST506) as Unit 1. This problem has been fixed in Release 3.4.

setrlimit Sometimes Did Not Return Error Codes

In previous releases, the setrlimit system call sometimes had problems handling invalid arguments to it. In this release, setrlimit will return EINVAL errors when newly supplied resource limits conflict with each other.

Occasional Spurious User Bus Errors Around Page Boundaries

Previously, certain programs running on Sun-3s would occasionally dump core from a SIGSIGV signal. This was due to the error handling of some prefetch conditions. The problem was mostly corrected in the 3.2 release, and fully corrected in Release 3.4.

New Version of the ie Ethernet Driver

Release 3.4 contains a new version of the ie (Intel 82586) Ethernet Driver. This driver is faster and more reliable than the previous version of the driver.

Incompatible Routine in System V curses Library

In previous releases, the System V curses library contained select.o, a select system call emulation. This routine was removed from curses in Release 3.4 because it was incompatible with the existing select system call. The manual page select(3V) will not be deleted, however.

Daylight Savings Times Were Incorrectly Calculated

The ctime and localtime routines incorrectly calculated Daylight Savings Times for the United States, Canada, and Australia. This has been fixed in Release 3.4.

MAKEDEV Did Not Create a vpc Node

MAKEDEV vpc did not make a special file for a vpc device (that is, /dev/vpc0). Instead, it made a node for a vp device (/dev/vp0). This behavior was as documented in the vpc(4S) manual page.

Both the behavior and the manual page have been corrected in Release 3.4. MAKEDEV vpc now makes a node for a vpc device.

Kernel Did Not Recognize Pseudo-terminals

Pseudo-terminals /dev/ptyr0 (33) through /dev/ptyrf (48) were not recognized by the kernel. The actual number of pseudo-terminals has increased to 48, as documented.

Hard Links to Directories Caused System Panic

Making a hard link to a directory with ln -f caused a panic. Normally, the superuser can force a hard link to a directory. However, a bug in Sun Release 3.0 resulted in a system panic when this was attempted.

This has been fixed in 3.4. Using ln -f to force a hard link to a directory no longer causes a panic.

readlink Bug

In Sun Release 3.2, a readlink() call against a special device returned a garbage value. This has been corrected in Release 3.4.

Problems with kmem

Earlier releases contained a bug that caused the kernel to crash with a bus error when /dev/kmem was accessed using high virtual addresses of the form 0xffffffxx. This has been fixed.

indent-related Bug Fixes

The following fixes were made to indent.

4.3 BSD Bug Fixes

Bug fixes from 4.3 BSD were incorporated into the indent program.

indent Problems with typedefs

The indent program now correctly generates and formats typedefs specified in the .indent.pro file. Formerly, only the first 14 typedefs were generated and formatted correctly.

Comment Blocks Were Incorrectly Formatted

The indent program now correctly formats groups of block comments such as the one below:

```
/****************
 *  first comment
 ****************/

        /*****************
         *  second comment
         *****************/
```

Missing Semicolons Confused indent

In Release 3.4, C code with missing semicolons no longer confuses indent. For example, indent used to complain about this code:

```
struct x {
        struct x        *next;
        NAMEBLOCK       name
};
```

A semicolon is missing after name, but this is still legal C code.

indent Misformats Certain Combinations

The indent program used to misformat the first line of a C function when the function had an explicit return value specified, the function had no local variables declared, and the option -npsl was selected on the command line, or specified in .indent.pro. This following code:

```
long _cgi_shared_screen()
{
   if (errno == EBUSY)
           _cgi_shared_screen_state = EBUSY;
   return (0);
}
```

used to give the following results:

```
long       _cgi_shared_screen()
{
   if          (errno == EBUSY)
                   _cgi_shared_screen_state = EBUSY;
   return (0);
```

This problem has been fixed in Release 3.4.

Formatting Problems with Negative Floating-point Numbers

Running indent on a C program containing a floating-point number with a negative exponent resulted in incorrect results. In this expression, indent used to put spaces around the minus sign, changing the value of the result as follows:

```
x = 100e - 5;
```

This problem has been fixed in Release 3.4.

indent Put Spaces Around "$"

Even though the dollar sign is a valid character in an identifier, indent used to put spaces around it. For example:

```
abc$def          became           abc $ def
```

The dollar sign is treated correctly now.

Utilities-related Bug Fixes

The following utilities-related bugs were fixed in Release 3.4.

Processes Failed to Reset Terminal Modes

The system console was rendered inaccessible by processes that failed to reset terminal modes properly. Processes that set FIONBIO on their standard input, then exit without resetting the terminal mode, cause programs that do not handle EWOULDBLOCK errors to exit also. Programs that exit as a result of this error include `csh` and `login`, among others. A problem arose when a such process, having the console as its standard input, exited. This caused the shell to die and left FIONBIO set on the console. The new `login` process initiated on the console then received the EWOULDBLOCK error, causing it to die, and rendering the console inaccessible. The console would remain unavailable until the system was rebooted (or until a special program was executed over a remote login) to clear the console's terminal mode.

This problem has been averted. `login` now resets the terminal mode for the console, if appropriate.

Using `sccs` with a `setuid` Front End

When you used `sccs` with an `SCCS` subdirectory that has write access restricted to the owner, it disallowed creation of the lock (`p`*.file*) for editing for all but the owner. This is true even when the edit function is called from within a.L setuid front end. This has been corrected in 3.4beta. `sccs` now allows files to be checked out when called from within a `setuid` front end.

Bug in `dkinfo`

The program `/etc/dkinfo` failed to close file descriptors. It ran out on long lists of disks and partitions and exited prematurely.

This has been corrected in Release 3.4. `/etc/dkinfo` now closes file descriptors associated with each partition it opens.

`Reply` Sometimes Garbled Addresses

Certain mail-message header formats caused return addresses to be garbled by the `Reply` command in `Mail` (`/usr/ucb/mail`). This has been corrected.

`vgrind` Omitted "?" Identifier

The `vgrind` program was changed in Release 3.4 so that it considers `?` to be a valid identifier for `emacs` MLisp code that it formats.

Numeric Fields Failed in `tbl`

The `tbl` program was fixed in Release 3.3 to treat numeric data fields correctly, and the same fix applies to Release 3.4. As a side effect of a bug fix involving data overflow, the `tbl` in Release 3.2 formatted numeric data incorrectly when there were many columns of data.

`.TH` Macro Bug

The `.TH` macro in the `man` macro package did not process arguments consistently with other `man` macro implementations, or as documented in *Formatting Documents on the Sun Workstation.*

This has been fixed in this release. Argument four, when supplied, now sets a value for the page foot left (version); argument five now sets a value for the page head center (main).

Bug in maze Demonstration

The maze demonstration halted and dumped core when run outside SunView on a high-resolution, black-and-white monitor, due to fixed-sized arrays that were too small for the particular screen. This has been corrected.

Bug in chesstool

When chesstool was invoked with a large font size, not all buttons in the panel were visible. This has been corrected.

Complex Makefiles Caused Hash Table Overflow

In Releases 3.0 and 3.2, very large and complex makefiles caused make's internal hash table to overflow. In Release 3.4, the size of this table has been redefined to a suitably huge value.

indxbib Dropped Core

In Releases 3.0 and 3.2, indxbib dropped core due to a memory fault when it processed files that had been run through addbib. This has been corrected.

Shell-related Bug Fixes

The following bug fixes were made relative to the C and Bourne shells.

Bug in login Command

An interaction between csh, login, and sh resulted in a situation whereby a user running login from within csh (typically by using the csh built-in login command), to log in as a Bourne shell user, found that CTRL-C interrupts were disabled for the resulting Bourne shell. Apparently, login did not completely account for and eliminate the effects of csh's job control.

This has been corrected. Interrupts now work properly for Bourne shell sessions started by logging in from csh.

time Memory Statistics Printed in Pages

The documentation for the csh built-in time command stated that timed-process memory size statistics were to be printed in Kbytes. However, in Releases 3.0 and 3.2, these figures were printed in machine pages. This led to discrepancies in reports for identical programs running on different machines, and to unrealistic-appearing statistics.

This has been corrected in this release. Memory size statistics are now printed in Kbytes, as stated.

6.5. Networking

The following network-related bugs were fixed in Release 3.4.

Problems Using ifconfig for Diskless Clients

If ifconfig was used to change the Ethernet address of a diskless client, that workstation would not reboot unless the arp table of its file server was first updated by hand to reflect the new address.

In Release 3.4, this has been corrected. rarpd has been modified to supply the correct client's Ethernet address to the server's arp table when the client reboots.

Bugs in TCP/IP

The TCP and ICMP protocol support in the kernel was updated in Release 3.4 to include many bug fixes between 4.2BSD and 4.3BSD. This fixes many problems with congestion on slow-speed networks, such as initial retransmission timeouts that were too short and the handling of ICMP redirect messages.

Net Booting on Non-zero Subnets

In Release 3.3, booting workstations over the Ethernet could fail in some cases on non-zero IP subnets, giving the message

```
    nd: output error 51
```

Network booting on subnets should be more reliable in Release 3.4.

Telnet Server End-of-line Conventions

The Internet Telnet server (`/usr/etc/in.telnetd`) was changed in Release 3.2 to be compatible with 4.3BSD, which interprets the CRLF sequence as LF. Since this was incompatible with Release 3.0 and several other Telnet implementations, in 3.4 CRLF is again interpreted as CR. Release 3.4 should again allow other operating systems and terminal servers to send CRs.

Frozen Configuration Files Ignored Domain Information

Release 3.2 added the ability to use the Yellow Pages domain as the default domain for `/usr/lib/sendmail`. If the `-bz` option was used on sendmail to create a "frozen" configuration file, the Yellow Pages domain could not be overridden. In Release 3.4, `/usr/lib/sendmail` will override the domain name with an explicit name in the "DD" line.

Lance Ethernet Driver Bug

Earlier releases contained versions of the Lance Ethernet driver with a bug that, while mostly harmless, did cause etherfind to report that minimum packet sizes were four bytes larger than they actually were. This has been fixed.

`ping` Sent Bad Packets

Earlier releases contained versions of `ping` that sometimes sent bad packets, which were then ignored by the remote host. Only even-length packets were treated correctly. This has been fixed.

`rpcinfo` Did Not Work on Non-networked Standalones

Earlier releases contained versions of `rpcinfo`, which did not work on standalone, non-networked machines. This has been fixed in Release 3.4.

NFS Clients Sometimes Wrote Garbage

Earlier releases contained versions of NFS in which clients could write garbage in certain very heavily loaded states. The problem was that, under certain circumstances, the NFS client's output buffer was reused even though it was still busy. These circumstances were as follows:

□ The client received a reply for a transmission of a certain request while waiting for a reply for a subsequent transmission of the same request.

□ Another process acquired the client handle, which contains the busy output buffer, then used XDR to put data into that buffer prior to the freeing of its last segment.

The NFS bug that caused these problems has been fixed.

Software Loopback lo0 Address Could Not Be Set

Earlier releases contained kernel network code that could not correctly redefine the software loopback lo0. After an attempt to do so, routing to lo0 yielded the message

```
Network is unreachable
```

This has been fixed in Release 3.4. However, you may experience other problems with changing the loopback address, since Yellow Pages software always uses the standard loopback address.

FBIOGVIDEO Was Sometimes Unreliable

Earlier releases contained versions of the system software that sometimes returned incorrect values on the FBIOGVIDEO `ioctl`. This has been fixed in 3.4.

6.6. System Administration

The following system-administration-related bugs were fixed in Release 3.4.

Diskless Clients Can Now Reboot After Changing Ethernet Address with `ifconfig`

When changing the Ethernet address of a diskless client with the `ifconfig` command, you used to have to modify the `arp` table of its server by hand to get it to reboot properly. This bug has been fixed in Release 3.4, so that you no longer have to modify the `arp` table.

`/etc/dkinfo`

`/etc/dkinfo` no longer runs out on long sequences of disks that have many partitions, and no longer generates a

```
too many open files
```

error message.

NFS Server Port Checking

Running an `adb -w` command on `/usr/etc/rpc.mountd` no longer generates a `no symbol found` error message.

Wrong `termcap` Entry for TVI-925

In Release 3.2, the `termcap` entry for the TVI-925 was modified so that it would support the Wyse-50 in TVI-925 emulation mode. In fact, the Wyse-50 does not emulate the TVI-925 correctly. The TVI-925 has the newline glitch `:xn:`, meaning it ignores a newline after autowrap. The Wyse-50 in TVI-925 emulation mode does not have this characteristic. In Release 3.3 (and 3.4) the TVI-925 `termcap` entry is correct, and there is a new entry called `wyse-925` to describe the Wyse-50 emulating a TVI-925.

7

Errata and Addenda for 3.2 Manuals

7

Errata and Addenda for 3.2 Manuals

This chapter contains errata and addenda for the following manuals:

□ Various language manuals

□ *Pixrect Reference Manual*

□ *SunCore Reference Manual*

□ *SunCGI Reference Manual*

□ *SunView Programmer's Guide*

□ *SunView System Programmer's Guide*

□ *System Administration Manual for the Sun Workstation*

Language Manuals

The following languages-related errata and addenda will be included in this section:

7.1. Language-related Addenda

Below is an addendum to languages documentation.

`dbx setenv` Documentation Bug

Insert the following paragraph in *Debugging Tools for the Sun Workstation*, immediately before Section 3.12, Page 30:

`setenv`
Set the environment variable *name* to the value of *string*. (See `csh`(1)).

`as` Addressing Modes Documentation

Insert the following section in the *Assembly Language Reference Manual* at the bottom of Page 42 after the paragraph that starts "Note that the 3.0 release...":

The memory indirect and program counter memory indirect addressing modes listed in the following tables are usable only when assembling for the MC68020. In each of these addressing modes, four user-specified values are used to generate the final operand address:

- base register
- base displacement
- index register
- outer displacement

All four user-specified values are optional. Both base and outer displacements may be null, word, or long. When a displacement is null, or an element is suppressed, its value is taken as zero in the effective address calculation.

In the case of memory indirect addressing, an address register (`a`*n*) is used as a base register, and its value can be adjusted by an optional base displacement (*d'*). An index register (`r`*i*) specifies an index operand (`r`*i*`:L:s`) and finally, an outer displacement (*d*) can be added to the address operand, yielding the effective address.

Program counter memory indirect mode is exactly the same except that that the program counter is used as the base register.

Some examples of these addressing modes follow:

```
an@(d':L,ri:L:s)@(d:L)
an@(d:L)@(d':L,ri:L:s)
an@@
an@(d:L)@
an@(d':L,ri:L:s)@
pc@@
pc@(d:L)@
pc@(d':L,ri:L:s)@(d:L)
pc@(d:L)@(d':L,ri:L:s)
@(d:L)@
@(d':L,ri:L:s)@(d:L)
@(d:L)@(d':L,ri:L:s)
@(d':L,ri:L:s)@
```

Language-related Errata

The following language-related errata have been reported in this release.

as Documentation Bug

In the *Assembly Language Reference Manual*, Appendix B, Table B-1, Page 72, the group beginning with fmovex, remove the last two lines in Column 3. They read

```
fmovex fm,ea{dn}
fmovex fm,ea{#k}
```

Debugging Documentation Bug

Remove the description of the attach command described on Page 30 of the *Debugging Tools Manual*. The command does not exist in dbx.

lint Line Count Bug

If a source program contains lines that are continued over more than one line, lint counts the continued lines as one single line. The line numbers reported by lint from that point no longer correspond to line numbers according to a text editor.

adb Signal Bug

Sometimes, if you try to run adb on a core file where the stack includes a frame created by _sigtramp, adb will dump core. The problem does not occur with dbx.

File I/O Facilities for Pixrects

Sun Microsystems, Inc. has specified a file format for files containing raster images. The format is defined in the header file <rasterfile.h>. The pixrect library contains routines to perform I/O operations between pixrects and files in this raster file format. This I/O is done using the routines of the C Library Standard I/O package, requiring the caller to include the header file <stdio.h>.

The raster file format allows multiple types of raster images. Unencoded and run-length-encoded formats are supported directly by the pixrect library. Support for customer-defined formats is implemented by passing raster files with non-standard types through filter programs. Sun supplied filters are found in the directory /usr/lib/rasfilters. This directory also includes sample source code for a filter that corresponds to one of the standard raster file types to facilitate writing new filters.

7.2. Writing and Reading Raster Files

The sections that follow describe how to store and retrieve an image in a rasterfile.

Write Raster File

```
int pr_dump(input_pr, output, colormap, type, copy_flag)
Pixrect *input_pr;
FILE *output;
colormap_t *colormap;
int type, copy_flag;
```

The pr_dump procedure stores the image described by a pixrect onto a file. It normally returns 0, but if any error occurs it returns PIX_ERR. The caller can write a rectangular subregion of a pixrect by first creating an appropriate input_pr via a call to pr_region. The output file is specified via output. The desired output type should either be one of the following standard types or correspond to a customer provided filter.

```
#define RT_OLD 0
#define RT_STANDARD 1
#define RT_BYTE_ENCODED 2
```

The RT_STANDARD type is the common raster file format in the same sense that memory pixrects are the common pixrect format: every raster file filter is required to read and write this format. The RT_OLD type is very close to the RT_STANDARD type; it was the former standard generated by old versions of

Sun software. The RT_BYTE_ENCODED type implements a run-length encoding of bytes of the pixrect image. This usually results in shorter files, although pathological images may expand by 50 percent. Specifying any other output type causes pr_dump to pipe a raster file of RT_STANDARD type to the filter named convert.*type*, looking first in directories in the user's $PATH environment variable, and then in the directory /usr/lib/rasfilters. type is the ASCII corresponding to the specified type in decimal. The output of the filter is then copied to output.

It is strongly recommended that customer-defined formats use a type of 100 or more, to avoid conflicts with additions to the set of standard types. The RT_EXPERIMENTAL type is reserved for use in the development of experimental filters, although it is no longer treated specially.

```
#define RT_EXPERIMENTAL 65535
```

pr_dump and other functions that start filters wait until the filter process exits before returning, so caution is advisable when working with experimental filters.

For pixrects displayed on devices with colormaps, the values of the pixels are not sufficient to recreate the displayed image. Thus, the image's colormap can also be specified in the call to pr_dump. If the colormap is specified as NULL but input_pr is a non-monochrome display pixrect, pr_dump will attempt to write the colormap obtained from input_pr (via pr_getcolormap). The following structure is used to specify the colormap associated with input_pr:

```
typedef struct {
    int type;
    int length;
    unsigned char *map[3];
} colormap_t;
```

The colormap type should be one of the Sun supported types:

```
#define RMT_NONE 0
#define RMT_EQUAL_RGB 1
#define RMT_RAW 2
```

If the colormap type is RMT_NONE, then the colormap length must be 0. This case usually arises when dealing with monochrome displays and 1-bit deep memory pixrects. If the colormap type is RMT_EQUAL_RGB, then the map array should specify the red (map[0]), green (map[1]) and blue (map[2]) colormap values, with each vector in the map array being of the same specified colormap length. If the colormap type is RMT_RAW, the first map array (map[0]), should hold length bytes of colormap data, which will not be interpreted by the pixrect library.

Finally, copy_flag specifies whether or not input_pr should be copied to a temporary pixrect before the image is output. The copy_flag value should be non-zero if input_pr is a pixrect in a frame buffer that is likely to be asynchronously modified. Note that use of copy_flag still will not guarantee that the correct image will be output unless the pr_rop to copy from the frame buffer is made uninterruptible.

```
#include <stdio.h>
#include <sys/types.h>
#include <pixrect/pixrect.h>
#include <pixrect/pr_io.h>

main()
{
    Pixrect *screen, *icon;
    FILE *output = stdout;
    colormap_t *colormap = 0;
    int type = RT_STANDARD;
    int copy_flag = 1;

    if (!(screen = pr_open("/dev/fb")) ||
        !(icon = pr_region(screen, 1050, 10, 64, 64)))
        exit(1);

    pr_dump(icon, output, colormap, type, copy_flag);
    pr_close(screen);

    exit(0);
}
```

Figure 7-1 *Example Program with* pr_dump

Read Raster File

```
Pixrect *pr_load(input, colormap)
FILE *input;
colormap_t *colormap;
```

The pr_load function can be used to retrieve the image stored in a raster file into a pixrect. The raster file's header is read from input, a pixrect of the appropriate size is dynamically allocated, the colormap is read and placed in the location addressed by colormap, and finally the image is read into the pixrect and the pixrect returned. If any problems occurs, pr_load returns NULL.

As with pr_dump, if the specified raster file is not of standard type, pr_load first runs the file through the appropriate filter to convert it to RT_STANDARD type and then loads the output of the filter.

Additionally, if colormap is NULL, pr_load will simply discard any and all colormap information contained in the specified input raster file. If colormap is non-null, pr_load will load the colormap data even if the type and length specified do not match that of the file (see pr_load_colormap below).

```
#include <stdio.h>
#include <sys/types.h>
#include <pixrect/pixrect.h>
#include <pixrect/pr_io.h>

main()
{
    struct pixrect *screen, *icon;
    FILE *input = stdin;
    colormap_t colormap;

    colormap.type = RMT_NONE;

    if (!(screen = pr_open("/dev/fb")) ||
        !(icon = pr_load(input, &colormap)))
        exit(1);

    if (colormap.type == RMT_EQUAL_RGB)
        pr_putcolormap(screen, 0, colormap.length,
            colormap.map[0], colormap.map[1],
            colormap.map[2]);

    pr_rop(screen, 1050, 110, 64, 64, PIX_SET, icon, 0, 0);
    pr_close(screen);

    exit(0);
}
```

Figure 7-2 *Example Program with* `pr_load`

7.3. Details of the Raster File Format

A handful of additional routines are available in the pixrect library for manipulating pieces of raster files. In order to understand what they do, you need to understand the exact layout of the raster file format.

The raster file is in three parts: first, a small header containing eight 32-bit `int`'s; second, a (possibly empty) set of colormap values; third, the pixel image, stored a line at a time, in increasing `y` order.

The image is essentially laid out in the file the exact way that it would appear in a static memory pixrect. In particular, each line of the image is rounded out to a multiple of 16 bits, corresponding to the rounding convention used by static pixrects.

The header is defined by the following structure:

```
struct rasterfile {
    int ras_magic;
    int ras_width;
    int ras_height;
    int ras_depth;
    int ras_length;
    int ras_type;
    int ras_maptype;
    int ras_maplength;
};
```

The ras_magic field always contains the following constant:

```
#define RAS_MAGIC 0x59a66a95
```

The ras_width, ras_height and ras_depth fields contain the image's width and height in pixels, and its depth in bits-per-pixel, respectively. The depth is usually either 1 or 8, corresponding to the standard frame buffer depths.

The ras_length field contains the length in bytes of the image data. For an unencoded image, this number is computable from the ras_width, ras_height, and ras_depth fields, but for an encoded image it must be explicitly stored in order to be available without decoding the image itself. Note that the length of the header and of the possibly empty colormap values are not included in the value in the ras_length field; it is only the image data length. For historical reasons, files of type RT_OLD will usually have a 0 in the ras_length field, and software expecting to encounter such files should be prepared to compute the actual image data length if it is needed. The ras_maptype and ras_maplength fields contain the type and length in bytes of the colormap values, respectively.

If the ras_maptype is not RMT_NONE and the ras_maplength is not 0, then the colormap values are the ras_maplength bytes immediately after the header. These values are either uninterpreted bytes (usually with the ras_maptype set to RMT_RAW) or the equal length red, green, and blue vectors, in that order (when the ras_maptype is RMT_EQUAL_RGB). In the latter case, the ras_maplength must be three times the size in bytes of any one of the vectors.

7.4. Writing Parts of a Raster File

The following routines are available for writing the various parts of a raster file. Many of these routines are used to implement pr_dump. First, the raster file header and the colormap can be written by calling pr_dump_header.

Write Header to Raster File

```
int pr_dump_header(output, rh, colormap)
FILE *output;
struct rasterfile *rh;
colormap_t *colormap;
```

pr_dump_header returns PIX_ERR if there is a problem writing the header or the colormap, otherwise it returns 0. If the colormap is NULL, no colormap values are written.

Initialize Raster File Header

```
Pixrect *pr_dump_init(input_pr, rh, colormap,
    type, copy_flag)
Pixrect *input_pr;
struct rasterfile *rh;
colormap_t *colormap;
int type, copy_flag;
```

For clients that do not want to explicitly initialize the rasterfile struct the following routine can be used to set up the arguments for `pr_dump_header`. The arguments to `pr_dump_init` correspond to the arguments to `pr_dump`. However, `pr_dump_init` returns the pixrect to write, rather than actually writing it, and initializes the structure pointed to by `rh` rather than writing it. If colormap is NULL, the `ras_maptype` and `ras_maplength` fields of `rh` will be set to RMT_NONE and 0, respectively.

If any error is detected by `pr_dump_init`, the returned pixrect is NULL. If there is no error, the `copy_flag` is zero, and the input pixrect is suitable for direct dumping (a primary memory pixrect), the returned pixrect is simply `input_pr`. However, if `copy_flag` is non-zero, or the input pixrect cannot be dumped directly, the returned pixrect is dynamically allocated and the caller is responsible for deallocating it with `pr_destroy` when it is no longer needed.

Write Image Data to Raster File

```
int pr_dump_image(pr, output, rh)
Pixrect *pr;
FILE *output;
struct rasterfile *rh;
```

The actual image data can be output via a call to `pr_dump_image`. This routine returns 0 unless there is an error, in which case it is PIX_ERR. It cannot write the image in a non-standard (filtered) format, since by the time it is called the raster file header has already been written.

Since these routines sequentially advance the output file's write pointer, `pr_dump_image` must be called after `pr_dump_header`.

7.5. Reading Parts of a Raster File

The following routines are available for reading the various parts of a raster file. Many of these routines are used to implement `pr_load`. Since these routines sequentially advance the input file's read pointer, rather than doing random seeks in the input file, they should be called in the order presented below.

Read Header from Raster File

```
int pr_load_header(input, rh)
FILE *input;
struct rasterfile *rh;
```

The raster file header can be read by calling `pr_load_header`. This routine reads the header from the specified input, checks it for validity, and initializes the specified `rasterfile` structure from the header. The return value is 0 unless there is an error, in which case it is PIX_ERR.

Read Colormap from Raster File

```
int pr_load_colormap(input, rh, colormap)
FILE *input;
struct rasterfile *rh;
colormap_t *colormap;
```

If the header indicates that there is a non-empty set of colormap values, they can be read by calling `pr_load_colormap`. If the specified colormap is NULL, this routine will skip over the colormap values by reading and discarding them. If the type and length values in `colormap` do match the input file, `pr_load_colormap` will allocate space for the colormap with `malloc`, read the colormap, and modify `colormap` before returning. If this occurs, the space allocated can be released with a `free(colormap->map[0])`.

The return value is 0 unless there is an error, in which case it is PIX_ERR.

Read Image from Raster File

```
Pixrect *pr_load_image(input, rh, colormap)
FILE *input;
struct rasterfile *rh;
colormap_t *colormap;
```

An image can be read by calling `pr_load_image`. If the input is a standard raster file type, this routine reads in the image directly. Otherwise, it writes the header, colormap, and image into the appropriate filter and then reads the output of the filter. In this case, both the rasterfile and the colormap structures will be modified as a side-effect of calling this routine. In either case, a pixrect is dynamically allocated to contain the image, the image is read into the pixrect, and the pixrect is returned as the result of calling the routine. If there is an error, the return value is NULL.

Read Standard Raster File

```
Pixrect *pr_load_std_image(input, rh, colormap)
FILE *input;
struct rasterfile *rh;
colormap_t colormap;
```

If it is known that the image is from a standard raster file type, then it can be read in by calling `pr_load_std_image`. This routine is identical to `pr_load_image`, except that it will not invoke a filter on non-standard raster file types.

Windows and Window Based Tools: Beginner's Guide

7.6. Errata and Addenda for the Windows and Window Based Tools Beginner's Guide

The following are errata and addenda for the Sun Release 3.0 version of the *Windows and Window Based Tools: Beginner's Guide.* These are in addition to the errata and addenda printed on Pages 114-118 of the *Release 3.4 Manual.* They do not cover all the new features of SunView (and in particular, `cmdtool`) added since the beginner's guide was published.

`cmdtool` Improved

The new functionality added to `cmdtool` (see Section 5.2.1, *Improvements to* `cmdtool`, above), has made the cautions for using `cmdtool` in Chapter 10, *The Command Facility*, unnecessary. On Page 123, delete the entire section titled ``vi, more, and su: Cbreak or Raw Mode.''

``Quad-Clicking'' in Text Subwindows

Add the following to Page 103 in Chapter 9, *Advanced Editing*, at the end of the first paragraph:

To select the entire contents of the text subwindow, point anywhere in the window and click the mouse four times in rapid succession.

SunView Programmer's Guide

7.7. Errata and Addenda for the SunView Programmer's Guide

The following pages contain errata and addenda for the Sun Release 3.2 version of the *SunView Programmer's Guide* (Part Number: 800-1345-10).

Maximum Attribute-Value List Size

Add to Page 28 in Chapter 3, *Interface Outline*, under "Attribute-based Functions:"

The maximum length of attribute-value lists supported by the SunView packages (see `ATTR_STANDARD_SIZE` in `<sunwindow/attr.h>`) is 250. If the number of attributes in a list you pass to SunView exceeds this size, the attribute package prints

```
Number of attributes(nnn) in the attr list exceeds
the maximum number(nnn) specified.  Exit!
```

on standard output and exits with exit status `1`.

Fitting Frames Around Subwindows

Many 3.0 SunView programs do not explicitly size their subwindows — they rely on getting the default window size.

Add the following to Page 38 in Chapter 4, *Windows*, after the paragraphs on `window_fit()`:

Since Release 3.2, if you use `window_fit()` or its variants for sizing the width and height of a frame, you need to be careful that the subwindows have some specified size, or they will be shrunk very small by the `window_fit()` call. Usually subwindows have a fixed size in one or both dimensions, or they are sized to be a percentage of the frame's size. The default size of a frame is that it encloses an area 34-rows by 80-columns in its default font.

Restrictions on Pop Up Windows

Add the following note to Page 45 in Chapter 4, *Windows*, at the end of "Popup Confirmer."

There are some restrictions on popup frames displayed using `window_loop()`:

- □ You can only have one subwindow in the popup frame.
- □ The only subwindow types that work properly are canvases and panels.

These limitations do not apply to non-blocking popups.

Window Positioning

`WIN_EXTEND_TO_EDGE` Invalid for Frames

Add the following footnote to Page 46 in Chapter 4, *Windows*, after "Specifying Subwindow Sizes and Positions."

It is meaningless to set the width or height of a frame to `WIN_EXTEND_TO_EDGE`, and it will interfere with subwindow behavior.

Zero `WIN_X`

Add the following to Page 48 in Chapter 4, *Windows*, in "Specifying Subwindow Sizes and Positions:"

The origin of the window system coordinate space is always in the upper left corner, so change 'right' to 'left' in the following sentence:

> The computations for these attributes take the borders and header of the frame into account, so that specifying `WIN_X` and `WIN_Y` of 0 will result in the subwindow being placed correctly at the upper right corner of the frame.

Changing Subwindow Size

Add the following section to Page 48 in Chapter 4, *Windows*, after "Specifying Subwindow Sizes and Positions:"

If you programmatically change the size or position of subwindows after you create them, you must explicitly re-specify the origin of any subwindows that are below or to the right of the altered subwindows, even if you specified the positions of these other subwindows using relative position attributes, such as `WIN_BELOW`. This is necessary because subwindows are not automatically laid out again when the positions and sizes of other subwindows are changed. They are only laid out again if the frame changes size. When re-specifying the layout of the other subwindows, you *can* use relative position attributes such as `WIN_BELOW`.

Attribute Ordering

Make these changes to Page 51 in Chapter 4, *Windows*, under "Command-line Arguments:"

In the two examples, the order of the arguments to `FRAME_ARGS` should be reversed, so that `argc` comes *before* `argv`, instead of `argv, argc` as indicated.

File Descriptor Usage Table

Page 54 in Chapter 4, *Windows*, Table 4-2, *SunView File Descriptor Usage*, is missing some items. Add the following:

`PANEL`	1	1 for the window
`MENU`	0	0 fullscreen access uses the window's fd
`CURSOR`	0	Most cursors are managed by the kernel.

Canvas Resizing

Add the following to the end of Section 5.6 on Page 65 in Chapter 5, *Canvases*:

Note that, as described in Section 4.8, ''Attribute Ordering,'' the canvas attributes are evaluated before the generic window attributes. This means that if you want to set the window size, then disable automatic sizing of the canvas, you must *first* set the window size, and *then*, in a separate `window_set()` call, disable `CANVAS_AUTO_SHRINK` and/or `CANVAS_AUTO_EXPAND`. If you do both in the same call, the auto-sizing will be turned off before the window size is set, so that the canvas size will not match the window size you specify. Here is an example of how to do it correctly:

```
canvas = window_create(frame, CANVAS,
                       WIN_HEIGHT, 400,
                       WIN_WIDTH,  600,
                       0);

window_set(canvas,
           CANVAS_AUTO_SHRINK, FALSE,
           CANVAS_AUTO_EXPAND, FALSE,
           0);
```

`LOC_TRAJECTORY` Events

In Section 6.2, ''Events,'' of Chapter 6, *Handling Input*, the documentation is muddled on the distinction between `LOC_MOVE` and `LOC_TRAJECTORY` events. As stated on Page 77 in ''Locator Motion Events,'' `LOC_MOVE` and `LOC_DRAG` events are collapsed (so that clients receive the most recent mouse location) unless you enable `LOC_TRAJECTORY` events. However, the rest of the description on Page 78 is wrong. When you enable `LOC_TRAJECTORY` events, clients do not receive any `LOC_MOVE` events, they get `LOC_TRAJECTORY` events instead, together with `LOC_DRAG` events if you have enabled them..

Thus, Table 6-1, *Event Codes*, on Page 76 is wrong. In it, change the description of `LOC_TRAJECTORY` events from

> Causes consecutive `LOC_MOVE`s to be collapsed

to

> Inhibits the collapse of mouse motions; clients receive `LOC_TRAJECTORY` events for every locator motion the window system detects.

Also, on Page 78, change

> Note that when you ask for `LOC_TRAJECTORY` events, you still only get `LOC_MOVE's` or `LOC_DRAGS's` in the input stream — however, you get more of them.

to

> Note that when you ask for `LOC_TRAJECTORY` events, you get (many!) `LOC_TRAJECTORY` events in place of `LOC_MOVE's`, but you still get `LOC_DRAG` events if you have enabled them.

TEXTSW_INSERT_MAKES_VISIBLE Does Not Affect textsw_insert()

Add the following note to Page 118 in Chapter 8, *Text Subwindows*, in Section 8.2, and to Page 322 in Chapter 18, *SunView Interface Summary*, in Table 18-25, *Text Subwindow Attributes*:

TEXTSW_INSERT_MAKES_VISIBLE applies only to text inserted by the user typing from the keyboard; textsw_insert() never scrolls the window. You should use textsw_possibly_normalize() to ensure that the desired section is visible, for example:

```
textsw_possibly_normalize(mytextsw,
    (Textsw_index) window_get(mytextsw, TEXTSW_INSERTION_POINT);
```

Cautions for Default Panel Item Layout

Add the following comment and caution to Page 141 in Chapter 9, *Panels*, in Section 9.2, after "Default Item Positioning:"

The default position for the next item is computed after an item is created. But if a client calls panel_set() after creating an item in such a way that the enclosing rectangle of the item is altered, the default position for the next item will *not* be recomputed. So, for example,

```
item = panel_create(panel, PANEL_MESSAGE, 0);
panel_set(item, PANEL_LABEL_STRING, "Hi", 0);

item1 = panel_create(panel, PANEL_MESSAGE,
                     PANEL_LABEL_STRING,  "There",
                     0);
```

will result in "There" overlapping "Hi."

CAUTION

Choice items currently have problems with item "creep." Each time the label of a choice item is set, the position of the item will be evaluated. If the value's position has not been fixed (with VALUE_X/Y), the value is positioned after the label. The problem is that the label is baseline-adjusted for a choice item. If the item position is not given when the label is set, the choice item will creep down because of the baseline adjustment.

Menu Generate Proc Attributes

On Page 203 in Chapter 11, *Menus*, under "Generate Procedures"

There is no MENU_ITEM_GEN_PROC attribute; this should be MENU_GEN_PROC_ITEM instead.

On Page 207 in Chapter 11, *Menus*, under "Pullright Generate Procedure"

There is no MENU_GEN_PULLRIGHT_PROC attribute; this is the MENU_GEN_PULLRIGHT attribute referred to elsewhere. The sample call to specify a menu item's pullright generate procedure should be

```
Menu    my_pullright_gen();

menu_set(menu_item, MENU_GEN_PULLRIGHT, my_pullright_gen, 0);
```

Nonexistent WIN_ICON Attribute

On Page 227 in Chapter 13, *Icons*,

WIN_ICON should be FRAME_ICON.

Creating Scrollbars

On Page 235 in Chapter 14, *Scrollbars*, in Section 14.3, "Creating, Destroying and Modifying Scrollbars," add the following after the example of a non-default scrollbar:

If you set the SCROLL_THICKNESS attribute then you *must* also set the SCROLL_DIRECTION of the scrollbar, since the dimension of the scrollbar that is altered by SCROLL_THICKNESS depends on the orientation of the scrollbar.

Differences Between notify_set_signal_func() and signal(3)

On Page 253 in Chapter 16, *The Notifier*, add the following information to "A signal() *Replacement for Notifier Compatibility:*"

The Notifier routine notify_set_signal_func() does not fully emulate the signal(3) function. It should deal with error states in a similar manner as signal(3). Errors from signal(3) are indicated by a -1 return value, and the value of errno is set to EINVAL.

The errors for notify_set_signal_func() are not communicated back to the caller, but error messages are printed. For example, if the signal number is not valid, the Notifier prints

```
Bad signal number
```

but its return value indicates success; the signal(3) system call does not print a message, but returns -1 and sets errno to EINVAL. As another example, if SIGKILL or SIGSTOP are ignored or a handler supplied, the Notifier prints

```
Notifer assertion botched: Unexpected error: sigvec
```

but its return value indicates success, while signal(3) does not print a message, returns value of -1, and sets errno to EINVAL.

The workaround is to use the following replacement function for signal(3) instead:

```
#include <sunwindow/notify.h>
#include <errno.h>

int (*
signal(sig,func))()
int     sig,(*func)();
{
        if ( (sig < 1 || sig > NSIG) ||
             (sig == SIGKILL || sig == SIGSTOP) ) {
                errno = EINVAL;
                return(BADSIG);
        }
        if (sig == SIGCONT && func == SIG_IGN) {
                errno = EINVAL;
                return(BADSIG);
        }
        return ((int(*)())notify_set_signal_func(sig,func,
                                          sig,NOTIFY_ASYNC));
}
```

Correct PANEL_ITEM_Y_GAP Value

Change the following on Page 301 in Chapter 18, *SunView Interface Summary*, in Table 18-15, *Panel Attributes*.

The correct value for the default PANEL_ITEM_Y_GAP is 5 pixels, not 10. (The *Panels* chapter is correct, the table is wrong.)

Undocumented PANEL_MARK_IMAGE and PANEL_NOMARK_IMAGE Attributes

The panel choice attributes PANEL_MARK_IMAGE and PANEL_MARK_IMAGE are attributes corresponding to PANEL_MARK_IMAGES and PANEL_MARK_IMAGES. They let clients set the selected and unselected images, respectively, for an individual mark. Modify Table 18-17, *Choice and Toggle Item Attributes*, on Page 304 in Chapter 18, *SunView Interface Summary*, as follows:

PANEL_MARK_IMAGE	int, Pixrect *	Image to mark choice specified by the first argument when it is selected. Default is push-button image: <images/panel_choice_on.pr>.
PANEL_NOMARK_IMAGE	int, Pixrect *	Image to mark choice specified by the first argument when it is not selected. Default is push-button image: <images/panel_choice_off.pr>.

Correct `TEXTSW_BLINK_CARET` Default

Change the following on Page 321 in Chapter 18, *SunView Interface Summary*, in Table 18-25, *Text Subwindow Attributes*.

The text subwindow caret blinks by default, so the default for `TEXTSW_BLINK_CARET` is `TRUE`, not `FALSE`.

Revised `typein.c` Example Program

On Page 381 in Appendix A, *Example Programs*.

The new attributes for tty subwindows (described in Section 5.2.6, *New TTY Subwindow Attributes*, above) make the sample program *typein* supplied in 3.2 needlessly complex, and eliminate the requirement for a dummy program, *loopback*, running in its tty subwindow. Here is a revised version of *typein* that uses the new `TTY_TTY_FD` attribute and `TTY_ARGV_DO_NOT_FORK` value for `TTY_ARGV`. The new version is included in the optional software for 3.4.

```
/*********************************************************/
#ifndef lint
static char sccsid[] = "@(#)typein.c 1.5 87/01/07 Copyr 1986 Sun Micro";
#endif
/*********************************************************/

#include <stdio.h>
#include <suntool/sunview.h>
#include <suntool/canvas.h>
#include <suntool/tty.h>
#include <ctype.h>

static Frame     frame;
static Canvas    canvas;
static Tty       tty;
static Pixwin   *pw;

static Notify_client    my_client;

#define STDIN_FD        0
#define STDOUT_FD       1
#define BUFSIZE         1000

main(argc, argv)
int     argc;
char  **argv;
{
        static Notify_value read_input();
        int     tty_fd;

        frame = window_create(NULL, FRAME,
                FRAME_ARGS,     argc, argv,
                WIN_ERROR_MSG,  "Cannot create frame",
                FRAME_LABEL,    "typein",
                0);

        tty = window_create(frame, TTY,
                WIN_PERCENT_HEIGHT,     50,
                TTY_ARGV,               TTY_ARGV_DO_NOT_FORK,
                0);

        tty_fd = (int)window_get(tty, TTY_TTY_FD);
        dup2(tty_fd, STDOUT_FD);
        dup2(tty_fd, STDIN_FD);

        canvas = window_create(frame, CANVAS,
                        0);
        pw = canvas_pixwin(canvas);

        /*
         * Set up a notify proc so that whenever there is input to read on
         * stdin (fd 0), we are called to read it.
         * Notifier needs a unique handle: give it the address of tty.
         */
        my_client = (Notify_client) &tty;
        notify_set_input_func(my_client, read_input, STDIN_FD);

        printf("Enter first coordinate:\nx? ");

        window_main_loop(frame);
        exit(0);
}

/*
 * This section implements a simple application which writes prompts to
 * stdin and reads coordinates from stdout, drawing vectors with the
 * supplied coordinates.  It uses a state machine to keep track of what
 * number to read next.
 */
#define GET_X_1         0
#define GET_Y_1         1
#define GET_X_2         2
#define GET_Y_2         3
int state = GET_X_1;
int x1, y1, x2, y2;


/* ARGSUSED */
static Notify_value
read_input(client, in_fd)
Notify_client   client;         /* unused since this must be from ttysw */
int             in_fd;          /* unused since this is stdin */
{
        char    buf[BUFSIZE];
        char   *ptr, *gets();

        ptr = gets(buf);   /* read one line per call so that we
                                   don't ever block */
                                  /* ^^^^^ does this matter any more?? */
        /* handle end of file */
        if (ptr==NULL) {
                /* Note: could have been a read error */
                window_set(frame, FRAME_NO_CONFIRM, TRUE, 0);
                window_done(tty);
        } else {
                switch (state) {
                case GET_X_1:
                        if (sscanf(buf, "%d", &x1) != 1) {
                                printf("Illegal value!\nx? ");
                                fflush(stdout);
                        } else {
                                printf("y? ");
                                fflush(stdout);
                                state++;
                        }
                        break;
                case GET_Y_1:
                        if (sscanf(buf, "%d", &y1) != 1) {
                                printf("Illegal value!\ny? ");
                                fflush(stdout);
                        } else {
                                printf("Enter second coordinate:\nx? ");
                                fflush(stdout);
                                state++;
                        }
                        break;
                case GET_X_2:
                        if (sscanf(buf, "%d", &x2) != 1) {
                                printf("Illegal value!\nx? ");
                                fflush(stdout);
                        } else {
                                printf("y? ");
                                fflush(stdout);
                                state++;
                        }
                        break;
                case GET_Y_2:
                        if (sscanf(buf, "%d", &y2) != 1) {
                                printf("Illegal value!\ny? ");
                                fflush(stdout);
                        } else {
                                printf("Vector from (%d, %d) to (%d, %d)\n",
                                        x1, y1, x2, y2);
                                pw_vector(pw, x1, y1, x2, y2, PIX_SET, 1);
                                printf("\nEnter first coordinate:\nx? ");
                                fflush(stdout);
                                state = GET_X_1;
                        }
                        break;
                }
        }
        return(NOTIFY_DONE);
}
```

SunView System Programmer's Guide

7.8. Errata and Addenda for the SunView System Programmer's Guide

The following pages contain errata and addenda for the Sun Release 3.2 version of the *SunView System Programmer's Guide* (Part Number: 800-1342-10).

Scrollbar Chapter

Page 187 in Chapter 15, *Scrollbars*, the third sentence from the top,

> For canvases and text subwindows...

should read

> For canvases and panels...

In the same chapter, on Page 191

> ... defaults to 4.

should read:

> ... defaults to 4 pixels.

System Administration Manual

7.9. System Administration Manual Addenda

The following pages are Errata and Addenda for Sun Release 3.0 version of the *System Administration for the Sun Workstation* manual (Part Number: 800-1323).

/etc/ttys File

Pages 146-147

The sample /etc/ttys file on Page 146 is incorrect and should read as follows:

```
12console
12ttya
02ttyb
12ttys0
1fttys1
   .
   .
   .
```

The last line of the example should read 1fttys1 instead of 14ttys1. The change of 4 to f matches the changes to the /etc/gettytab entries below; that is, the 1fttys1 line indicates ttys1 should be enabled for logins at 1200-baud.

Change the sample /etc/gettytab file on Page 147 to read as follows:

```
c|std.300|300-baud: \
    :nd#1:cd#1:sp#300:
f|std.1200|1200-baud: \
    :fd#1:sp#1200:
2|std.9600|9600-baud: \
    :sp#9600:
```

The corresponding sentence in the paragraph following the /etc/gettytab file excerpt on Page 147 should read as follows:

> Therefore, a 'c' as the second character on the line would set the rate to 300, an 'f' would set the rate to 1200, and a '2' would set the rate to 9600.

Run `catman` to Access Online Manual

After installing your system, you need to run `catman -w` while in the `/usr/man` directory. This creates the `whatis` database. The `whatis` database must exist for you to be able to use either the `man -k` command (which prints out one-line summaries from the `whatis` database) or the `whatis` command (which describes what a command is).

Note that it usually takes about two hours for the `whatis` database to be generated.

Hooking Up a Serial ASCII Printer

Page 160

In the *Editing the* `printcap` *File* section, add the following sentence to the end of the bulleted description of `lp`:

> This file should be owned by `daemon`, belong to the `daemon` group, and have permissions 660.

Page 161

In the *Other File System Modifications* section under the first bulleted item, the first sentence should read as follows:

> Check to make sure the proper permissions and ownerships exist on the files `/usr/lib/lpd`, `/usr/ucb/lpr`, `/dev/ttya`, and on the directory `/usr/spool/lpd`.

Add the following line to the example:

```
# ls -lg /dev/ttya
crw-rw----  1 daemon   daemon  12,     0 Oct 21 11:57 /dev/ttya
```

Hooking Up a Printer to a VPC-2200 Multibus Board

Page 165

In the section *Editing the* `/etc/printcap` *File*, add the following sentence to the end of the bulleted item on `lp`:

> This file should be owned by `daemon`, belong to the `daemon` group, and have permissions 660.

In the *Other File System Modifications* section under the first bulleted item, the first sentence should read as follows:

> Check to make sure the proper permissions and ownerships exist on the files `/usr/lib/lpd`, `/usr/ucb/lpr`, `/dev/vpc0`, `/dev/lp0`, and on the directory `/usr/spool/lpd`.

Add the following lines to the example:

```
# ls -lg /dev/vpc0 /dev/lp0
crw-rw----  1 daemon   daemon  28,     1 Oct 21 11:57 /dev/lp0
crw-rw----  1 daemon   daemon  28,     0 Oct 21 11:57 /dev/vpc0
```

Page 164

Add the following sentences to the section *Using* MAKEDEV *to Create Special Files*:

> `/dev/vpc0` is for the Versatec printer/plotter interface of the Systech VPC-2200 board. `/dev/lp0` is for the Centronics/Dataproducts interface of the same board. Once you have created the `/dev` entries, change the permissions with the following commands:

```
  # chown daemon /dev/vp0 /dev/vpc0 /dev/lp0
  # chgrp daemon /dev/vp0 /dev/vpc0 /dev/lp0
  # chmod 660 /dev/vp0 /dev/vpc0 /dev/lp0
```

Major Changes in Disk and Directory Layout in Release 3.0 and Later Releases

The changes in disk and directory layout in Release 3.0 are not described in the *System Administration for the Sun Workstation* manual for Release 3.0. There is no `/usr2` partition in Release 3.0; instead there is `/usr/`*server_name/hostname*. This is important because if you put user directories in `/usr2`, you will end up with full root file systems almost immediately.

The changes in disk and directory layout are described on Page 4 of the *Release 3.0 Change Notes.*

Preparing a Previously-Used Client Partition

Page 64

Under the section *Preparing a Previously-Used Client Partition*, users are referred to *Installing UNIX on the Sun Workstation* manual for instructions on how to prepare a client partition "by hand." There are no such instructions in the *Installing UNIX on the Sun Workstation* manual.

To prepare a client partition "by hand," refer to Appendix G of the *Installing UNIX on the Sun Workstation* manual for a listing of the contents of the distribution tape. On the first distribution tape is a raw client image in `tar` format, so it is no longer necessary to dump a raw partition to tape when installing your system; instead you can just `tar` off the client image and use that for your client partition.

Then refer to Appendix A in the *Installing UNIX on the Sun Workstation* manual for a list of files affected on a client. These are the files that the *Setup* program would have created for you, so make sure that these files exist and alter them accordingly for the new client. Then install any site-specific files. Then finish setting up the new client partition as you have other clients on your network.

dump Cannot Be Used to Dump an NFS-mounted File System

Note that you cannot use dump to dump an NFS-mounted file system. You can only dump hard partitions (/dev/rxy0a through /rxy0h) or soft partitions (/dev/rndl0, /dev/rndl1, and so forth.) For example, if you wanted to dump the home directories in the /usr partition, you could type the following:

```
# /etc/dump 9uf /dev/nrmt0 /dev/rxy0d
```

to do a level 9 dump of the home directories in /usr. If you wanted to dump the clients' root partitions you might type:

```
# /etc/dump 9uf /dev/nrmt0 /dev/rndl0
# /etc/dump 9uf /dev/nrmt0 /dev/rndl1
# /etc/dump 9uf /dev/nrmt0 /dev/rndl2
```

This would dump the clients' root (soft) partitions, which are located on the hard partition /dev/xy0c. You could also dump a file system onto a remote system's tape drive from a machine where you are logged in as superuser by using the command:

```
# /etc/rdump 9uf remote_machine_name:/dev/rmt0 /dev/xy0d
```

This would dump the d hard partition where the home directories are located. Use the rdump command when doing remote dumps over the Ethernet. You cannot, however, dump an NFS-mounted file system.

How to Use *yppasswd* Versus *passwd*

Use the passwd command to change (or create) your local /etc/passwd file. Use yppasswd to change your password in the yp database. The yppasswd command does not affect your local password file, or any of the password files on any remote machines on which you have accounts; it only affects the password in the yp database. Thus the password in the yp database may be different from the one on your own machine.

Note that only the owner of the name or the superuser can change a password.

yppasswdd, the yellow pages password daemon, must be running on your yp server for the new password to take effect.

For more information, refer to the passwd(8), yppasswd(1), and passwd(5) manual pages.

Print Filter Hook

Page 167

In the *Adding Hardware to your System* chapter, the subsection entitled *Output Filter Specifications* describes the interface for print filters. The line printer daemon (lpd) uses a hook for switching from one filter to another, and introduces special characters that the filter can detect and thus switch to the other filter. If you are using more than one print filter, you need to use the following print filter hook:

```
/*
 *  This example filter demonstrates how to handle the ^Y^A sequence lpd
 *  sends for switching from this 'of' filter to another.  This action is
 *  required when using 'of' filters only for filtering or printing banner
 *  pages and this 'of' filter is stopped to use another filter, e.g., 'if'.
 *
 *  This filter highlights the lpd-generated banner page by overstriking and
 *  is intended to work with an 'if' filter entry in the printcap file.
 *
 *  The banner text can be processed in other ways, e.g., parsing the short
 *  banner line (see 'sb' in printcap(5)) and incorporating this information
 *  in your own banner page.
 */

#include <stdio.h>
#include <signal.h>

main(argc, argv)
int argc;
char *argv[];
{
    register char *cp;
    register int ch;

    /* 'of' filters are only passed width and length arguments from lpd */
    while (--argc) {
        if (*(cp = *++argv) == '-') {
            switch (cp[1]) {
            case 'w':
                /* this filter does not use width argument */
                break;

            case 'l':
                /* this filter does not use length argument */
                break;
            }
        }
    }

    while ((ch = getchar()) != EOF) {
        switch (ch) {
        case ' 31':
            /*
             * lpd needs to use a different filter to
             * print data so stop what we are doing and
             * wait for lpd to restart us.
             */
            if ((ch = getchar()) == ' 01') {
                (void) fflush(stdout);
                (void) kill(getpid(), SIGSTOP);
                break;
            } else {
                (void) ungetc(ch, stdin);
        ch=' 31';
      }

    default:
      putchar(ch);
      putchar('\
)     putchar(ch);
;     break;
    }
  }
  exit(0);
}
```

`fstab` Entry for a Sun-2/3Com Diskless Client Mounting from a Sun-3 Server

If a diskless Sun-2/3-Com client NFS-mounts a file system from a Sun-3 server, extra entries need to be added to the `/etc/fstab` and `/etc/nd.local` files to make it work successfully. Otherwise, the Sun-3 server pumps out packets so fast that the Sun-2 cannot keep up. The NFS buffer size and `nd` maxpack parameter have to be limited, or else you will get too many re-transmissions causing *server not responding* messages.

If the Sun-2/3-Com mounts an NFS file system from a Sun-3, you need to have the Sun-2/3-Com specify "rsize=2048" in `/etc/fstab` for this mount. This makes sure NFS reads by the Sun-2/3-Com NFS client from the Sun-3 server are reduced to a size that the 3-Com can handle.

If a Sun-3 mounts an NFS file system from a Sun-2/3-Com, you need to have the Sun-3 specify "wsize=2048" for this mount. This makes sure that NFS writes by the Sun-3 client to the Sun-2/3-Com NFS server are reduced to a size that the 3-Com can handle.

If you have a Sun-2/3-Com diskless client being served by a Sun-3, you must similarly adjust the `nd` software to prevent this overrun of the 3-Com. This is done by adding a "2" at the end of each Sun-2/3-Com client line in `/etc/nd.local` file on the Sun-3 server.

The reverse situation, that is, having a Sun-3 diskless client being served by a Sun-2/3Com server, cannot be compensated for by changing `/etc/nd.local` and is not recommended.

How to Add a New `yp` Slave Server Not in the Original Set

Page 45

The following section replaces the section currently on Page 45 of the *System Administration for the Sun Workstation* manual:

To add a new `yp` slave server, start by modifying some maps on the `yp` master server. If the new server is a host that has not been a `yp` server before, you must add the host's name to the *ypservers* map in the default domain. To add a server named *ypslave* to `domain_name`, do the following:

```
ypmaster# cd /usr/etc/yp/domain_name
ypmaster# ../makedbm -u ypservers > /tmp/temp_file
ypmaster# vi /tmp/temp_file
        <add the new hostname to the temporary file>
ypmaster# ../makedbm /tmp/temp_file ypservers
```

Running the makedbm command with the -u option undoes *ypservers*; that is, it converts it from *dbm* format into text file format temporarily so that you can add the new hostname to the temporary file *temp_file*. Then running the makedbm command with *temp_file* as the input file and *ypservers* as the output file converts *ypservers* back into *dbm* format.

You can then set up the new yp slave server's databases by copying the databases from the yp master server *ypmaster*. To do this, remote log in to the new yp slave server, and run the ypinit(8) command as follows:

```
ypslave# cd /usr/etc/yp
ypslave# ypinit -s ypmaster
```

To verify that the *ypservers* file is correct (because there is no ASCII file for the *ypservers* map), do the following:

```
ypslave# cd /usr/etc/yp/domain_name
ypslave# ../makedbm -u ypservers
```

Note: If a host name is not in the *ypservers* map it will not be notified when updates are made to the yp database.

Then complete the steps described in the section *How To Set Up A Slave yp Server* above.

uucp *L.cmds* File

Each line of the L.cmds file contains the name of a command. Avoid leaving any white space that is not part of the name of a command, either at the beginning or end of the line. If you leave white space in the L.cmds file, uucp incorrectly parses the file and sends out the error message

```
PERMISSION DENIED to LOGFILE
```

A

Installing SunPro

A

Installing SunPro

This appendix explains how to finish installing the SunPro optional software. Before following the instructions in this section, you should have upgraded your system to 3.4 and selected the SunPro option during the installation process.

A.1. Installation Instructions

Follow these instructions to install SunPro:

1. Enter

```
% cd /usr/sunpro
```

When you select SunPro during the 3.4 upgrade process, the operating system makes the directory `/usr/sunpro`, which contains the following new SunPro software:

```
compile      default.mk     filemerge.1       ld     make
cpp          filemerge      install_sunpro    make.1
```

`install_sunpro` is the shell script that you run to complete SunPro installation.

2. Enter

```
% install_sunpro
```

During the installation process, `install_sunpro` does the following:

- creates the directory `/usr/sunpro/3.2`, and copies pre-SunPro versions of `compile`, `/lib/cpp` , `/bin/ld`, `/usr/bin/m4`, `/bin/make`, and `/usr/man/man1/make1` into it.

Note: If you have a heterogeneous server, running this script installs SunPro on all of the server's architecture.

- moves the files in `/usr/sunpro` to their appropriate places within the file system, overwriting older versions of the software with the new SunPro files.

After you complete these steps, the system will automatically run SunPro versions of such files as `make` and `ld` when you invoke them. If you need more file space, you may want to delete the older files in `/usr/sunpro/3.2`.

For information about filemerge, refer to the `filemerge` manual page in the manual page package included with this release. The `make` *Manual*, also included in this release, explains the `make` tool available with SunPro.

B

Optional Software

B

Optional Software

The following optional software is available with Release 3.4:

Networking

```
usr/ucb/rdist
usr/etc/ping
usr/etc/rarpd
usr/etc/rpcinfo
```

Debug

```
usr/ucb/dbx
usr/bin/dbxtool
```

SunView User

```
usr/bin/suntools
usr/bin/othertools
usr/lib/.rootmenu
usr/lib/.suntools
usr/lib/.textswrc
usr/lib/fonts/tekfonts
usr/lib/defaults
usr/lib/fonts/fixedwidthfonts
```

SunView Prog

```
usr/include/images
usr/include/suntool
usr/include/sunwindow
usr/lib/libtoolmerge.a
usr/lib/libsuntool.a
usr/lib/lint/llib-lsuntool
usr/lib/lint/llib-lsuntool.ln
usr/lib/libsunwindow.a
usr/lib/lint/llib-lsunwindow
usr/lib/lint/llib-lsunwindow.ln
```

SunView/Demo Source

```
usr/src
```

Text Processing Tools

```
usr/ucb/vgrind
usr/bin/refer
usr/bin/tbl
```

Fortran

```
usr/lib/f77pass1
usr/lib/libF77.a
usr/lib/libI77.a
usr/lib/f1
usr/lib/cg
```

Graphics

```
usr/lib/libcgi.a
usr/include/cgiconstants.h
usr/include/cgidefs.h
usr/include/cgicbind.h
usr/include/cgipw.h
usr/include/cgi_gpl_pwpr.h
usr/lib/libcgi77.a
usr/include/f77/cgidefs77.h
usr/lib/libcore.a
usr/lib/libcore77.a
usr/lib/libcorepas.a
usr/lib/libcore68881.a
usr/lib/libcorefpa.a
usr/lib/libcoresky.a
usr/lib/libpixrect.a
usr/include/pixrect/cg2reg.h
usr/include/pixrect/chain.h
usr/include/pixrect/gp1cmds.h
usr/include/pixrect/pixfont.h
usr/include/pixrect/pixrect_hs.h
usr/include/pixrect/pr_dblbuf.h
usr/include/pixrect/pr_io.h
usr/include/pixrect/pr_util.h
```

Pascal

```
usr/lib/f1
usr/lib/pc0
usr/lib/libpc.a
```

Profiled Libraries

```
usr/lib/libc_p.a
```

```
usr/lib/libpc_p.a
usr/lib/libtoolmerge_p.a
usr/lib/libsuntool_p.a
usr/lib/libsunwindow_p.a
usr/lib/libF77_p.a
usr/lib/libI77_p.a
usr/lib/libm_p.a
```

System V

```
usr/5include/sys/ioctl.h
usr/5lib/libcurses.a
usr/5lib/libcurses_p.a
```

Man Pages

```
usr/man/man1/indent.1
usr/man/man2/getrlimmit.2
usr/man/man8/ifconfig.8c
```

Demo

```
usr/demo/GP1/VIEWPORT/matrix.c.C
usr/demo/GP1/VIEWPORT/gpbuf.c.C
usr/demo/SRC/maze.c
usr/demo/SRC/gomaster.c
usr/demo/SRC/goban.c
usr/demo/SRC/goboard.c
usr/demo/SRC/gopanel.c
usr/demo/SRC/goservice.c
usr/demo/SRC/gocapture.c
usr/demo/SRC/gocircle.c
usr/demo/SRC/goprint.c
usr/demo/SRC/goban.h
usr/demo/SRC/Makefile
```

Games

```
usr/games/chesstool
```

SunPro

```
usr/sunpro/commpile
usr/sunpro/cpp
usr/sunpro/filemerge.1
usr/sunpro/filemerge
usr/sunpro/install_sunpro
usr/sunpro/ld
usr/sunpro/m4
usr/sunpro/make
```

C

Manual Pages

C

Manual Pages

A number of manual pages have been reprinted to include corrections of both reported and unreported technical errors, as well as enhancements for Sun Release 3.4. The following manual pages have been reprinted, and are included as attachments to this release manual.

Reprinted Manual Pages with Reported Errors Corrected

Manual Page	*Description*
`aliases(5)`	The `.forward` file must be readable by all, home directory searchable by all, to forward mail.
`ascii(7)`	The octal chart was mislabeled as decimal.
`fingerd(8C)`	Reference Page missing from Release 3.2.
`getrlimit(2)`	Now returns EINVAL for invalid parameters.
`gettimeofday(2)`	Declared `timeval` structure incorrectly, was unclear that `tz_dsttime` indicates that daylight savings time *could* apply (not whether it is currently in effect), omitted mention that nonzero return values are defined in the header file `/usr/include/sys/time.h`.
`kadb(8s)`	Additions have been made to text.
`login(1)`	Manual page was missing from printed manual, omitted mention of `/etc/securetty` root-login terminal access file.
`mount(8)`	Incorrectly indicated that the default `retry` value for `nfs` file systems was `1`, when it is `10000`.
`on(1C)`	Added a BUGS entry: a CTRL-Z in an NFS-mounted directory hangs the window from which the signal originates.
`rdist(1)`	Dollar-sign characters (`$`) were misprinted as percent signs (`%`).
`sh(1)`	Dollar-sign characters (`$`) were misprinted as percent signs (`%`).
`shmop(2)`	Spurious "EINVAL" appears in wrong paragraph.
`socket(2)`	Omitted EPROTOTYPE error return.
`vlimit(3C)`	Declaration of arguments missing.

Reprinted Manual Pages with other Corrections and Enhancements	
Manual Page	*Description*
bwone(4S)	Now documents enhancements to the Sun Release 3.4 device driver.
bwtwo(4S)	Now documents enhancements to the Sun Release 3.4 device driver.
catman(8)	Now documents harmless but disturbing warning messages.
cgfour(4S)	Now documents enhancements to the Sun Release 3.4.
chmod(2)	Now documents sticky bit for directories.
clear_colormap(1)	Now documents enhancements to Sun Release 3.4.
cp(1)	Now warns that the -r option does not preserve symbolic links.
defaultsedit(1)	Now documents new environment variable, SUNTOOLS_DEFAULTS.
dump(8)	Now notes that multi-file tapes are not supported.
ex(1)	Now documents EXINIT environment variable and .exrc files.
edquota(8)	Now documents requirement for a /etc/quotas file in order to activate quotas.
fbio(4S)	Documents enhancements for Sun Release 3.4.
fopen(3S)	Notes NULL returns, other corrections.
frexp(3)	Now documents 4.3 BSD enhancements made in Sun Release 3.2.
goban(6)	Now documents enhancements including labeled points, branching play and running-commentary.
grep(1V)	Now notes that -n behavior is position-dependent when concatenated with other options.
id(1V)	Now classified correctly as System V optional software.
indent(1)	Now documents enhancements for Sun Release 3.4.
lpr(1)	Now documents PRINTER environment variable and the default-printer printcap entry lpr supplies
overview(1)	Corrected erroneous view-surface names.
pack(1)	Corrected vertical page alignment.
ps(1)	Now documents the fact that the reported size of process segments depends on options selected.
rasfilter8to1(1G)	Now documents enhancements for Sun Release 3.4.
rastrepl(1)	Now documents enhancements for Sun Release 3.4.
rcp(1)	Now documents -p (preserve access and mod. times) option.
read(2V)	Now notes that you must use readdir or getdirentries over NFS.
scanf(3S)	Changed to reflect corrected handling of errors.
scanf(3V)	Changed to better reflect the System V behavior.
screendump(1)	Now documents enhancements for Sun Release 3.4.
screenload(1)	Now documents enhancements for Sun Release 3.4.
setup(8)	Formatting error, index entry was embedded in text.
sticky(8)	Documents Sun Release 3.4 enhancement, sticky directories. When sticky bit set for a directory, files can only be deleted by their respective owners.
suntools(1)	Documents new environment variable, SUNTOOLS_DEFAULTS.
symoder(1)	Error in synopsis.

The online sources to the following manual pages have been updated to correct typographical and other minor errors.

Online Manual Pages with Minor Corrections

Manual Page	*Description*
ar(1)	Undocumented member-name length restriction, 15 characters; longer names are truncated in the library entry.
biff(1)	Must own terminal to use, but superuser owns terminal windows; workaround noted.
binmail(1)	-i option is invalid.
bstring(3)	ffs returns 0 for passed value of 0.
creat(2)	Missing reference to open(2), which supercedes creat.
csh_builtins(1)	Bad ".so" reference.
cuserid(3)	Typographical error.
etherfind(8C)	Spurious reference to { and } removed.
find(1)	Typographical error.
getgrent(3)	Formatting error.
getrusage(2)	Formatting error
indxbib(1)	Does not allow pathnames in *database* argument, only files in current directory.
mntent(5)	Spelling error.
monitor(8s)	Minor change to register names in the R command.
more(1)	Typographical error.
putc(3S)	Minor clarifications, returns integer written.
printf(3)	Formatting error.
readdir(3)	Formatting error.
rm(1)	Fails to note that rm checks real UID before removing a directory.
select(2)	The structure timeval should be declared as type *timeout.
swapon(8)	Typographical error.
textedit(1)	FILES entry /tmp/Text* should be /tmp/EtHost*.
vi(1)	Did not document EXINIT environment variable. (See attached reprint of ex manual page.)
vacation(1)	Must use pathname /usr/ucb/vacation in the .forward file
vpc(4S)	FILES entry /dev/vp0 should be /dev/vpc0.
wall(1)	*file* argument not documented, names a file containing text of broadcast message.
ypfiles(5)	Minor clarification to description of YP database.

Revision History

Revision	*Date*	*Comments*
01α	1/30/87	Alpha draft.
α	2/2/87	Interim Alpha draft.
50β	2/16/87	Beta draft.
β	3/2/87	Interim Beta draft.
β	3/24/87	Interim Beta draft.

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 287815

For U.S. Sales Office locations, call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Sun House
31-41 Pembroke Broadway
Camberley
Surrey GU15 3XD
England
0276 62111
TLX 859017

Australia: 61-2-436-4699
Canada: 416 477-6745
France: (1) 46 30 23 24
Germany: (089) 95094-0
Japan: (03) 221-7021
The Netherlands: 02155 24888
UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters:
0276 62111

Elsewhere in the world, call Corporate Headquarters:
415 960-1300
Intercontinental Sales