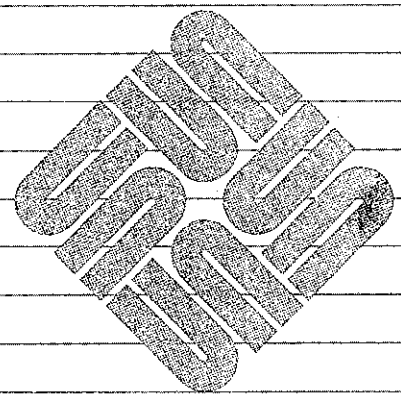
sun microsystems

Release 2.2 Manual for the Sun Workstation



Sun Microsystems, Inc. • 2550 Garcia Avenue • Mountain View, CA 94043 • 415–960–1300

Part Number 800-1318-06
Revision A of 3 October, 1985

Part Number 800-1318-06
Revision: A of 3 October, 1985

Release 2.2 Manual

for the

Sun Workstation

Sun Microsystems, Inc.,
2550 Garcia Avenue,
Mountain View,
California 94043
(415) 960-1300

Credits and Trademarks

Multibus is a trademark of Intel Corporation.

Sun Microsystems and **Sun Workstation** are registered trademarks of Sun Microsystems, Incorporated. **Sun-2, Sun2, Sun-2/xxx, Deskside, SunStation, SunCore, SunWindows,** and **DVMA** are trademarks of Sun Microsystems, Incorporated.

UNIX is a trademark of AT&T Bell Laboratories.

DEC is a trademark of Digital Equipment Corporation.

Contents

Contents

1

Introduction

Introduction

This document describes Release 2.2 for the Sun Workstation. This is an upgrade release. You must be running either Release 2.0 or 2.1 in order to load this release. Release 2.2 incorporates Release 2.1 with general bug fixes for Release 2.0 and 2.1. The procedures for installing and testing Release 2.2 from a distribution tape are included in this document.

Chapter 2 describes the contents of Release 2.1. This release was restricted to the Sun-2/160 Color Workstation with the new Graphics Processor and Grpahics Buffer options. If you are not using this type of workstation proceed to Chapter 3-Installing the 2.2 Release. This chapter combines installation instructions for workstations now running Release 2.0 as well as those running 2.1. Chapter 4-Software Bug Fixes is new to Release 2.2. Chapter 5-Errata Pages and Chapter 6-Insert Pages contain information pertinent to both Release 2.1 and Release 2.2. Chapter 7 contains Appendix H:SunWindows Examples for the *Programmer's Reference Manual for SunWindows.*

1.1. Supporting Documentation

The following manuals are useful for the installation procedures for Release 2.2.

[1] *System Administration for the Sun-2 Workstation*
(Part Number: 800-1150).

[2] *Commands Reference Manual for the Sun Workstation*
(Part Number: 800-1172).

[3] *Installing UNIX on the Sun Workstation*
(Part Number: 800-1158).

NOTE *If you have the Sun-2/160 Color SunStation with a Graphics Processor, you may want to refer to the two manuals listed below:*

[4] *Hardware Installation Manual for the Sun-2/160 Color SunStation and Sun-2/130 SunStation*
(Part Number: 800-1144).

[5] *Sun-2/160 Diagnostic Manual*
(Part Number: 800-1236). :

2

New Graphics Processor and Graphics Buffer Options for the Sun-2/160

New Graphics Processor and Graphics Buffer Options for the Sun-2/160

This chapter describes the contents of the Release 2.1 which exclusively supported the Sun-2/160 Color Workstation with the new Graphics Processor and Graphics Buffer options. Since these options are VME bus compatible boards that require a color system, Release 2.1 was restricted to the Sun-2/160. IF YOU DO NOT HAVE Sun-2/160 Color Workstation with the new Graphics Processor and Graphics Buffer PROCEED TO CHAPTER 3.

The procedures for installing and testing this special release are incorporated in Release 2.2. A Release 2.0 installation is required before a Release 2.2 upgrade.

2.1. New Hardware Products

The Graphics Processor (GP) and Graphics Buffer (GB) boards are options designed to enhance graphics performance for the Sun-2/160 Color Workstation. These are Eurocard format printed circuit boards that fit into VME slots of the Sun-2/160 card cage.

2.2. New Software Products

There are no new software products.

2.3. Changes to Operating System Software

The operating system software changes are included in a device driver for the Graphics Processor, microcode that is downloaded into the Graphics Processor, and a configuration utility that binds frame buffers to a GP board.

The lowest level command interface for the Graphics Processor is not included. The Graphics Processor can only be used through the graphics library routines that have been expanded to communicate with the GP.

Device Drivers

The file *gpone.o* is a binary copy of the device driver for the Graphics Processor board. It is installed in the kernel configuration directory during the software installation procedure.

The file *cgtwo.o* is a new binary copy of a device driver for the color frame buffer. It has additional *ioctl*'s for communicating with the Graphics Processor. This device driver is upward compatible with previous versions.

The file *consfb.o* is a new binary copy of the device driver for the console frame buffer. Some applications like *suntools* (1) use */dev/fb* as the default frame buffer. This new version allows redirection of the GP frame buffer driver to */dev/fb*.

Downloadable GP Microcode

The files */etc/gp1cg2.1152.ucode* and */etc/gp1cg2.1024.ucode* contain microcode that is downloaded into the Graphics Processor board when the Sun-2/160 is booted.

***gpconfig* Command**

The *gpconfig* command binds specific frame buffers to the Graphics Processor board. Since an individual Graphics Processor board can drive up to four frame buffers, the GP driver must be given certain frame buffer specific information. Chapter 4 includes a manual page, *gpconfig* (8), that describes the *gpconfig* command and its arguments. If the Graphics Processor is to be used regularly, the *gpconfig* command should be added to the file */etc/rc.local*.

2.4. Changes to Application Software

The application software that changed is limited to the graphics libraries listed below. New versions of some of the utilities from Release 2.0 must be recompiled to run with these new libraries. Release 2.2 will not contain any new utilities.

Graphics Libraries

Four libraries in */usr/lib* have been revised to use the Graphics Processor board. Any application programs that depend on these libraries should be relinked with the newer versions.

SunCGI — The Sun Microsystems implementation of the Computer Graphics Interface (CGI) standard reflects Sun Microsystems interpretation of the March 1984 working draft of CGI. The file *libcgi.a* is a new version of this graphics library for use with the GP. See *Programmer's Reference Manual for SunCGI* (Part Number: 800-1166). Section 4 has an errata page reflecting the new GP device that should be included in this manual.

SunCore — SunCore is an implementation of the ACM Core graphics standard with extensions. The file *libcore.a* is a new version of this graphics library for use with the GP. The file *libcoresky.a* is a version of this library for use with the GP and a SKY floating point board. See *Programmer's Reference Manual for SunCore* (Part Number: 800-1165). Section 4 has an errata page reflecting the new GP device that should be included in this manual.

Pixrects — Pixrects is a low-level graphics library for manipulating rectangles of pixels with RasterOps. The file *libpixrect.a* is a version of this graphics library for use with the GP. See *Programmer's Reference Manual for SunWindows — the Sun Window System* (Part Number: 800-1167) and *Programmer's Tutorial to SunWindows* (Part Number: 800-1182).

SunWindow Tools

There are no new SunWindow tools with this release. Tools from Release 2.0 must be reinstalled with the new graphics libraries described above. This is performed automatically installation utility.

NOTE *Locally written tools must be relinked with the new graphics libraries.*

2.5. Performance Issues

This release represent the first stage of an ongoing effort to improve graphics performance through the use of the GP. The Pixrects, SunCGI, and SunCore graphics libraries will all provide increased performance. The sections that follow outline the graphics performance improvements resulting from the GP.

Pixrects

Vector pixel rates have increased by about a factor of 5. Solid-filled rectangle rates have increased by more than a factor of 3. RasterOps from one area to another on the frame buffer (e.g. scrolling) have increased by more than a factor of 3. Polygon scan conversion is nearly 10 times faster for certain polygons Pixrects cannot currently exploit the full performance of the GP due to the lack of a batching mechanism.

SunCGI

Sun CGI takes advantage of the Pixrects level performance improvements and also used the GP to perform coordinate transformations from VDC to device coordinates. SunCGI also sends the vectors of a polyline to the GP as a batch of vectors which greatly improves polyline speed.

SunCore

Since SunCore has floating point world coordinates and a display list, several further performance improvements are achieved with the GP. Vectors can be read out of a retained segment in the display list, 3D images transformed, clipped and drawn at over 20,000 vectors per second. The transformations from floating point world coordinates to NDC space are performed by the GP as well as various matrix multiply operations. These floating point operations are improved by as much as 200 times. 2D and 3D solid color or Gouraud shaded polygons are transformed, clipped, scaled, or rendered by the GP. If a GB board is present, hidden surface elimination for 3D solid or Gouraud shaded polygons is also performed by the GP.

3

Installing the 2.2 Release

Installing the 2.2 Release

In this chapter, we give directions for installing the Sun 2.2 release software on a Sun system running the 2.0 Release.

NOTE *We recommend that you install the 2.2 Release in single user mode. If you do not, you will be unable to upgrade the following four programs:* `/etc/inetd`, `/etc/ypbind`, `/usr/etc/rpc.mountd`, and `/usr/lib/lpd`. `These are multi-user mode processes. You can not overwrite existing active processes.`

The directions below assume you are working with the incremental release software on either one 1/4'' or 1/2'' tape, and support installation on

- Standalone machines with local tape and disk, which can read in the distribution tape via the local tape drive;
- Machines with a local disk, but no local tape drive, that are on a network. Such machines will use the tape drive on another machine (called *remote_host* or *server_name* in the procedures) to read the tape;
- Server machines with local tape and disk and with diskless clients, which will use the local drive to install both the server and its clients.

Before beginning installation, there are several important things you should be aware of:

- If you are already running release 2.0, you must have *at least 508KBytes of disk space available on your root partition, and at least 26KBytes available on your /usr partition to do this installation. If you wish to load the optional software included on the tape (manual pages and demonstration executables and source), allow at least another 4.0 MBytes on /usr (22 KBytes for the manual pages, 3.5 MBytes for the demos and 485 Kbytes for the games).*

  The *df*(1) command displays information about space available in each file system. You should use this command before installing the 2.2 Release software to make sure that there is enough disk space available for it. For example:

```
gaia% df
Filesystem    kbytes    used   avail capacity  Mounted on
/dev/xy0a       7437    5470    1223    82%    /
/dev/xy0h      148455  128709    4900    96%    /usr
/dev/xy0g      117327   66896   38698    63%    /usr/misc
[and so on]
```

In this example, the */usr* file system has 4.9 MBytes of disk space available.

- If you are installing a new disk, you *must* follow the directions in *"Installing Unix on the Sun Workstation"* (Part Number: 800-1158) for formatting and labeling it.
- This upgrade release is for installation only on systems running Release 2.0 or 2.1 software. This release is *incremental* in the sense that *you need not re-install the complete operating system.*
- This release is intended for installation as a package; you must install the entire release. Sun will not provide direct support for users who wish to install selected portions of the release software.
- You can 'un-install' this release if you have to: a facility has been provided with the release for backing out changes. In most circumstances, you should not find this necessary; however, if you do, please inform Sun Microsystems Technical Support — we'd like to know what went wrong.
- You must build and install a new operating system kernel to complete the installation of the 2.2 release.
- *We strongly advise a thorough back-up before beginning installation.* See the manual entry *dump* (8) in the *Commands Reference Manual for the Sun Workstation* (Part Number: 800-1166).
- In order for the GP and/or the GB to run, the command */etc/gpconfig* must be run when the workstation is booted. This command loads the GP and GB boards with their respective microcode. This can be done automatically if the file */etc/rc.local* is edited to include the *gpconfig* command line. See the manual page for *gpconfig* (8) in Chapter 6 of this document.

3.1. What is on the Distribution Tape?

Distribution of the 2.2 Release binaries is either on a 1/4'' magnetic tape cartridge or a 1/2'' nine-track tape. The tapes contain eight files, as follows:

File 1: Boot block.
A general-purpose boot program which knows how to boot from the various devices that can be attached to the Sun Workstation. The PROM monitor boots this general-purpose boot program.

File 2: Bootable *diag* program.
diag is the disk formatting and labeling program.

File 3: Copyright file.

File 4: *tar* file of the backup utility.
This *tar* file contains two files: the backup utility (*2.2_backup*) which

saves the current versions of the files replaced in this upgrade and the listing of files which changed between 2.2 and 2.0 (*get_arch_f*).

File 5: *tar* file of the 2.2 upgrade utility (*2.2_upgrade*).

File 6: *tar* file of new 2.2 object
A *tar* format file of the 2.2 object files, executable files, and libraries.

File 7: *tar* file of optional software.
With Release 2.2, this includes several new/revised manual pages, gp demos, and gp diagnostics.

File 8: Copyright file.

3.2. Overview of the Installation Procedure

The object of this procedure is to load the Release 2.2 binaries from the magnetic tape onto your local or network disk subsystem. You will need to have a blank tape for Step 5.

The basic steps in installation are:

1. Boot single user mode.
2. Load the release tape.
3. Extract the 2.2 backup utility and 2.2 upgrade utility.
4. If you are installing a server, halt any diskless clients.
5. Run the 2.2 backup utility.
6. Run the 2.2 upgrade utility.
7. Optionally, use the *tar*(1) command to extract the manual pages and/or color demos.
8. If you have a Sun-2/160 with a Graphics Processor, edit the file */etc/rc.local* to contain a command line for */etc/gpconfig* (8).
9. Reconfigure your system kernel and reboot.

If you have performed an installation of a Sun system before, you will recall that we use several conventions in the procedures and examples to try and clarify things:

- What the system types at you is printed in `typewriter font like this`.
- What you type at the system is shown in **`boldface typewriter font like this`**. Everything shown in boldface should be typed exactly as it appears.
- Where parts of a command are shown in *italic text like this*, they refer to a variable which you have to substitute from a selection; it is up to you to make the proper substitution.

The *tape* variable is important. The values for *tape* are listed in Table 3-1.

For example, a common configuration would load from a quarter-inch magnetic tape cartridge via an SCSI tape controller. In this case, *tape* would be replaced by *st* (*S*CSI *T*ape) everywhere.

Now, you are ready to begin the actual installation.

3.3. Boot single user mode

To boot single user mode, do the following:

```
gaia#/etc/shutdown -h
>b vmunix -s
# /etc/mount /usr
```

3.4. Load the Release Tape

NOTE *If you are installing 2.2 on a network disk server, you must have a tape drive on the server machine.*

NOTE *We do not guarantee full compatibility between the 4-track 20 MByte and the 9-track 45 MByte drives.*

Your chances of successfully reading tapes produced by a different type of drive are increased if you follow the manufacturer's instructions for drive maintenance: clean the drive heads after every use of a new tape and after every eight hours of use.

Load the release tape.

3.5. Extract the 2.2 Backup and Upgrade Utilities from Tape

When you have loaded the tape, use the *tar*(1) command to extract the *2.2_upgrade*, the 2.2 installation utility.

- *If you are using a local tape drive*, do the following. Remember to replace *tape* with the appropriate device abbreviation for your tape (*ar* for the Archive drive, *st* for an SCSI tape drive, or *mt* for the nine-track tape):

```
# cd /usr/etc
# mt -f /dev/nrtape0 ret
      [Note: a 1/2" tape drive cannot be retensioned]
# mt -f /dev/nrtape0 rew
# mt -f /dev/nrtape0 fsf 3
# tar xvpf /dev/nrtape0
x 2.2_backup  3488 Bytes, 7 tape blocks
x get_arch_f  3924 Bytes, 8 tape blocks
# mt -f /dev/nrtape0 rew
# mt -f /dev/nrtape0 fsf 4
# tar xvpf /dev/nrtape0
x 2.2_upgrade  4494 Bytes, 9 tape blocks
#
```

The *ret* option of the *mt*(1) command will cause the tape drive to wind the tape forward to the end and back to the beginning to get even tension on the tape.

- *If you are using a remote tape drive*, do the following. Note that, since you are performing a remote process as super-user, the hostname of the local machine (which you are typing commands on) must be in the remote machine's */.rhosts* file to avoid permission problems. In addition, each machine must have an entry for the other (name and Internet address) in its */etc/hosts* file. Remember to replace *tape* with the appropriate device abbreviation for the remote tape drive you are using, to replace *remote_host* with the hostname of the machine this tape drive is attached to, and to replace *block_size* with *20b* for a 1/2" tape or *126b* for a 1/4" tape:

```
# cd /usr/etc
# rsh remote_host mt -f /dev/nrtape0 ret
      [Note: a 1/2" tape drive cannot be retensioned]
# rsh remote_host mt -f /dev/nrtape0 rew
# rsh remote_host mt -f /dev/nrtape0 fsf 3
# rsh remote_host dd if=/dev/nrtape0 \
      bs=block_size | tar xvpBf -
x 2.2_backup 3488 Bytes, 7 tape blocks
x get_arch_f 3924 Bytes, 8 tape blocks
# rsh remote_host mt -f /dev/nrtape0 rew
# rsh remote_host mt -f /dev/nrtape0 fsf 4
# rsh remote_host dd if=/dev/nrtape0 \
      bs=block_size | tar xvpBf -
x 2.2_upgrade.  4494 Bytes, 9 tape blocks
#
```

If you get a "`Broken pipe`" message, you can ignore it.

3.6. Halt Diskless Clients

If you are installing a server, halt all diskless clients.

3.7. Run the 2.2 Backup Utility

Next, you run the 2.2 backup utility to save the 2.0 or 2.1 files on your disk that are replaced in this upgrade. This process takes about 20 minutes.

You will now need a blank tape.

The backup utility will not support backup for ND clients. If you have changed any of the following files in the clients' root file systems, you will manually have to dump each client. The files affected are:

/etc/gp1cg2.1024.ucode
/etc/gp1cg2.1152.ucode
/etc/gpconfig
/etc/inetd
/etc/mount
/etc/nfsd
/etc/mount
/etc/shutdown -h
/etc/umount
/etc/ypbind
/etc/ypserv
/etc/yp/makedbm

/dev/MAKEDEV

You can *tar* these files to blank tape. Whenever you need to uninstall a client, backup your server first then *tar* these files back to the client.

If you are not running 2.1 release, when you do 2.2 backup you will see the following message: `tar: filename: no such file or directory` for any files that pertain to the 2.1 release. The files affected are:

/etc/gp1cg2.1024.ucode
/etc/gp1cg2.1152.ucode
/etc/gpconfig
/usr/sys/OBJ/gp1_colormap.o
/usr/sys/OBJ/gp1_kern_sync.o
/usr/sys/OBJ/gp1_rop.o
/usr/sys/OBJ/gpone.h
/usr/sys/OBJ/gpone.o
/usr/sys/conf/SDST160GP
/usr/sys/sun/gpio.h
/usr/include/sun/gpio.h
/usr/include/gp1_pwpr.h

If you are not using the yellow pages and have moved ypbind to ypbind- you will want to reverse this with the following command before you proceed:

mv /etc/ypbind- /etc/ypbind

There are two parameters with the *2.2_backup* utility. (the command is printed on two lines for formatting purposes only; type it as a single line):

```
# /usr/etc/2.2_backup {ar|mt|st}
      {server|tapefull|tapeless server_name}
```

The backup command and system response for a server machine will look like this:

```
# f(LB/usr/etc/2.2_backup mt server
Beginning backup
backup: load blank tape to mt and press return.
Extracting object files for backup.
```

NOTE *Make sure your tape is write protected after finishing this step.*

3.8. Run the 2.2 Upgrade Utility

Load the release tape now.

NOTE *Do not run SunWindows during the upgrade procedure.*

If you are doing the upgrade on a server, note that the upgrade utility takes care of 2.2 upgrade on your diskless clients, but that it assumes a standard form of the

nd configuration file */etc/nd.local* in order to do so. In particular, if there are lines in the server's */etc/nd.local* for client partitions which are commented out (lines with a leading '#'), these clients will *not* have 2.2 installed on them. If you wish to have the 2.2 Release installed on these clients, you must remove the comment symbol from their lines in the file before performing installation on the server. If you do not do this during the initial install, you will have to run through the entire installation procedure again (on the server and all the clients) in order to bring the commented-out client partitions up to date.

Also, if your */etc/nd.local* file has `user...` lines and/or `ether ...` lines that refer to non-existent clients, comment out these lines before running the installation utility. Again, the utility will not run to completion if this is not done.

There are two parameters with the *2.2_upgrade* utility. (the command is printed on two lines for formatting purposes only; type it as a single line):

```
# /usr/etc/2.2_upgrade {ar|mt|st}
      {server|tapefull|tapeless server_name}
```

- The first set specifies your tape device:

Table 3-1 *Tape Devices*

Devices	
ar	Archive quarter-inch tape
mt	Nine-track magnetic tape
st	SCSI tape controller

- The last designates your machine as a server, standalone with tape drive, or workstation without a tape drive using a another machine's (*server_name*) tape drive:

```
server
tapefull
tapeless server_name
```

For example, the installation command and system response for a server machine with a half-inch tape drive would look like this:

```
# /usr/etc/2.2_upgrade mt server
Beginning 2.2 install.
Extracting 2.2 object files.
      . . .
[ and so on . . . extraction takes about 20 minutes . . . ]
 . . .
Installing new bootable code on server.
Beginning 2.2 install on diskless clients.
Beginning 2.2 install on client client_1.
Completed 2.2 install on client client_1.
Beginning 2.2 install on client client_2.
Completed 2.2 install on client client_2.
Beginning 2.2 install on client client_3.
Completed 2.2 install on client client_3.
Completed 2.2 install on diskless clients.
Running ranlib on new libraries.
        . . . . .
2.2 install completed.
You should now reconfigure and rebuild your kernel.
```

NOTE *If you are using an SCSI tape for the install, you may get a message like* "/dev/nrst0 rewind 1 failed: I/O error" *at the beginning or end of the install utility. You can ignore it.*

To install the release on a tapeless workstation using the 1/4" SCSI tape drive on a machine named *hal*, the command line would be:

```
# /usr/etc/2.2_upgrade st tapeless hal
        ...
```

3.9. Loading Optional Software from the Release Tape

The seventh file on the upgrade tape contains new/revised manual pages, several new color demos and games, and GP diagnostics. You may optionally load this software.

Manual pages take approximately 22 KBytes of space. They are:

gpone (4S)
gpconfig (8)
nfsmount (2)
mount (8)
toolplaces (1)
st (4S)

Demos take approximately 3.5 MBytes of space. They are:

cframedemo
framedemo
jumpdemo
rotobj
shaded
showmap
stringart
DATA/.vecs*
READ_ME
show
flight
draw
suncube
molecule

Games take approximately 485 KBytes of space. They are:

gammontool
chesstool

The GP diagnostic program is the file *gp1.2.diag*. It takes 400 KBytes.

To extract the optional software use the directions that follow.

- First forward position the tape with *mt*(1) to file 7.

```
# cd /usr
# mt -f /dev/nrtape0 rew
# mt -f /dev/nrtape0 fsf 6
```

- Use the appropriate command line arguments to *tar* to select which directories (*demo*, *man*, and/or *games*) you wish load. If you do not specify a directory, *tar* loads the manual pages, demos and games. The complete load takes about 14 minutes, regardless of which options are chosen.

  For a machine with a local tape drive:

```
# tar xvpf /dev/nrtape0 ./man
      loads manual pages
# tar xvpf /dev/nrtape0 ./demo
      loads demonstration programs
# tar xvpf /dev/nrtape0 ./games
      loads games
# tar xvpf /dev/nrtape0 ./games ./demos
      loads games and demos
# tar xvpf /dev/nrtape0
      loads manual pages, demonstration programs, and games
# tar xvpf /dev/nrtape0 /stand
      loads GP diagnostic
# mt -f /dev/nrtape0 rew
```

For a machine using a remote tape drive, type the following. Remember to replace *tape* with the appropriate device abbreviation for the remote tape drive you are using, to replace *remote_host* with the hostname of the machine this tape drive is attached to, and to replace *block_size* with **20b** for a 1/2" tape or **126b** for a 1/4" tape:

```
# cd /usr
# rsh remote_host mt -f /dev/nrtape0 rew
# rsh remote_host mt -f /dev/nrtape0 fsf 6
# rsh remote_host dd if=/dev/nrtape0 \
      bs=block_size | tar xvpBf - ./man
# rsh remote_host mt -f /dev/nrtape0 rew
```

See the above discussion of loading with a local tape drive for an explanation of how to optionally load the manual pages, demonstration programs and/or games with *tar*.

The *dd* command above will print the number of records read in and written out, for example:

```
6 + 0 records in
6 + 0 records out
```

This message may be interspersed with the standard error output from *tar* and, in any case, may be ignored. Also, you can ignore the "`Broken pipe`" message delivered by *tar*.

□ If you load the pages, and normally use *catman*(8) to create pre-formatted copies of your online manual pages, then don't forget to re-issue the *catman* command for the new 2.2 pages.

3.10. Reconfigure your UNIX System Kernel

NOTE *Changes have been made to the device description lines in the kernel configuration file to allow for the Sun-2/160 GP option.*

Finally, to complete the 2.2 Release installation, you must reconfigure your system kernel.

NOTE *Changes have been made to the device description lines in the kernel configuration file to allow for the Sun-2/160 GP option.*

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

Installing UNIX on the Sun Workstation
(Part Number: 800-1158).

If you have previously configured a kernel, you can use the following sections to guide you through reconfiguration. The first subsection is an annotated copy of the new *GENERIC* kernel configuration file; read it carefully to make sure you are including the correct device description lines for your system. The second

subsection gives reconfiguration procedures for standalone machines, and the third subsection addresses servers.

Kernel Reconfiguration for Standalone Systems

For standalone machines, proceed as follows.

1. Change the current directory to */sys/conf*:

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

2. Create a kernel configuration file. There are two ways to produce the kernel configuration file.

- Copy the file *GENERIC* and comment out the lines that don't apply to your system. We'll call the new file *SYS_NAME* (the name of the system). For example,

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

- Alternatively, copy the file *SDST160GP* onto a file called *SYS_NAME* (the name of the system). This file is a basic Model 160 kernel configuration file. If you have any additional devices on your system, you should add lines to this file as appropriate for your system.

```
# cp SDST160GP SYS_NAME
# chmod +w SYS_NAME
```

3. Edit */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 you are including the proper device description lines for your system.

4. Edit the file */etc/rc.local* to contain a line for the *gpconfig* (8) command.

```
/etc/gpconfig gpone0 -b -f cgtwo0
```
for initializing the GP and GB

```
/etc/gpconfig gpone0 -f cgtwo0
```
for initializing the GP only

If you don't want to have the GP and/or GB boards active all the time, you can run the *gpconfig* (8) command interactively.

5. Create the directory *../SYS_NAME* (if you haven't already) to contain the kernel image. Remember: since the system build utility */etc/config* places its output files there, this directory *must* have the same name as your system configuration file:

```
# mkdir ../SYS_NAME
```

6. Still in the */sys/conf* directory, run */etc/config*. Then change directory 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 depend
[ lots of output ]
# make
[ lots of output ]
```

7. Now you can save your old kernel and install your new one:

```
# mv /vmunix /vmunix.old
# cp vmunix /vmunix
# /etc/shutdown -h
        The system goes through the halt sequence, then
    the monitor displays its prompt, at which point you
    can boot the system:
> b
        The system boots up multi-user, and then
    you can try things out.
gaia#
```

8. If the system appears to work, this completes the upgrade procedure. If performance is slow, check that *gpconfig* has been run properly. If the new kernel doesn't seem to be functioning properly, boot */vmunix.old*, copy it back to */vmunix*, and go about fixing your new kernel:

```
# /etc/shutdown -h
> b vmunix.old -s
# mv /vmunix /vmunix.oops
# cp /vmunix.old /vmunix
# ^D      [ Brings the system up multi-user ]
gaia#
```

Kernel Reconfiguration for Servers

For server machines, proceed as follows.

1. Change the current directory to */sys/conf*:

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

2. Create a kernel configuration file. There are two ways to produce the kernel configuration file.

- Copy the file *GENERIC* and comment out the lines that don't apply to your system. We'll call the new file *SYS_NAME* (the name of the system). For example,

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

- Alternatively, copy the file *SDST160GP* onto a file called *SYS_NAME* (the name of the system). This file is a basic Model 160 kernel configuration file. If you have any additional devices on your system, you should add lines to this file as appropriate for your system.

```
# cp SDST160GP SYS_NAME
# chmod +w SYS_NAME
```

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

4. Edit the file */etc/rc.local* to contain a line for the *gpconfig* (8) command.

```
/etc/gpconfig gpone0 -b -f cgtwo0
```
for initializing the GP and GB

```
/etc/gpconfig gpone0 -f cgtwo0
```
for initializing the GP only

If you don't want to have the GP and/or GB boards active all the time, you can run the *gpconfig* 8 command interactively.

5. Create the directory *../SYS_NAME (if you haven't already) to contain the kernel image. Remember: since the system build utility /etc/config places its output files there, this directory must have the same name as your system configuration file:*

```
# mkdir ../SYS_NAME
```

6. Still in the */sys/conf* directory, run */etc/config*. Then change directory 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 depend
[ lots of output ]
# make
[ lots of output ]
```

7. If you have a specially configured client kernel, it can be reconfigured now as well:

```
# cd /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, given above. ]
# mkdir ../CLIENT_KERNEL_NAME
# /etc/config CLIENT_KERNEL_NAME
# cd ../CLIENT_KERNEL_NAME
# make depend
[ lots of output ]
# make
[ lots of output ]
```

8. Now you can position yourself in the directory which has the server's kernel in it, save your server's old kernel, install your new one, and try everything out:

```
# cd ../SYS_NAME
# mv /vmunix /vmunix.old
# cp vmunix /vmunix
# /etc/shutdown -h
      The system goes through the halt sequence, then
   the monitor displays its prompt, at which point you
   can boot the system:
> b
      The system boots up multi-user, and
   then you can try things out.
gaia#
```

9. Next, install the appropriate client kernel in */pub*.

- If you reconfigured a special client kernel (in Step 5 above), copy it into */pub*:

```
# cd /sys/CLIENT_KERNEL_NAME
      [or wherever your client kernel is]
# cp vmunix /pub/vmunix
```

- Otherwise place a copy of your server's kernel (if appropriate) in */pub*:

```
# cp /vmunix /pub/vmunix
```

9. If everything appears to work, you can finish by rebooting each of your clients. See the final step in the standalone instructions above if you have problems with your kernel.

NOTE *If you want to run the yellow pages, be sure to move* `ypbind-` *back to* `ypbind`.

If performance is slow, check that *gpconfig* has been run properly.

Kernel Reconfiguration — an Annotated Copy of *GENERIC*

The following pages provide an annotated copy of the *GENERIC* file shipped with this distribution. You can use the explanations of each line in the file to determine which lines should be included in your own system configuration file.

```
#
# GENERIC SUN
#
machine   sun
```

[**mandatory.**]

```
cpu       "SUN2"
```

[**mandatory.**]

```
ident     GENERIC
```

[**mandatory.** If you use `GENERIC` as your system identifier, you may use the `swap generic` clause in the `config` line below. If you customize the identifier to *SYS_NAME*, you must either include an `options GENERIC` line, or specify at least the device where your root file system lives in place of `swap generic`. For example, the `config` line for a standard Sun-2 might read: `config vmunix root on xy`. See *General* and *Specific System Description Lines*, in *Installing Unix on the SunWorkstation* above, for information. Finally, if *SYS_NAME* contains both alpha and numeric characters (as in, 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 timezone. Adjust value accordingly.]

```
maxusers  4
```

[**mandatory.** Number may vary. For most systems, "2" is the proper value for maxusers.

```
options   INET
```

[**mandatory.** Controls inclusion of Internet code. See *inet*(4). You must also include the `pseudo-device inet` and `pseudo-device loop` lines below.]

```
options   SYSACCT
```

[Controls inclusion of code to do process accounting. See *acct*(2) and *acct*(5). If you include this line, you must also include the `pseudo-device sysacct` line below.]

```
options   RPC
```

[Necessary for the network file system.]

```
options   NFS
```

[Necessary for the network file system.]

```
config    vmunix  swap generic
```

[**mandatory**. Specify kernel name and configuration clauses. See *Specific System Description Lines*, above, for information.]

```
pseudo-device    rpc
```

[Necessary for the network file system.]

```
pseudo-device    nfs
```

[Necessary for the network file system.]

```
pseudo-device    pty
```

[Pseudo-tty's. Needed for network or window system.]

```
pseudo-device    bk
```

[Berknet line discipline for high speed tty input. See *bk*(4).]

```
pseudo-device    sysacct
```

[See `options SYSACCT` line above.]

```
pseudo-device    inet
```

[**mandatory**. See `options INET` line above.]

```
pseudo-device    ether
```
mandatory.

[ARP code. See *arp*(4).]

```
pseudo-device    loop
```

[**mandatory**. Software loop back network device driver. See *lo*(4). Must include with 'options INET'.]

```
pseudo-device    nd
```

[Network disk. Necessary for servers and diskless clients, and for machines serving as remote hosts for remote installation. See *nd*(4).]

```
pseudo-device    win128
```

[Window system. Number indicates maximum windows. If you include this line, you must also include the `pseudo-device dtop`, `ms`, and `kb` lines just below.]

```
pseudo-device    dtop4
```

[Maximum number of screens (`desktops`). Required for window system.]

```
pseudo-device    ms3
```

[Maximum number of mice. Required for window system. See *ms* (4).]

```
pseudo-device    kb3
```

[Maximum number of Sun keyboards. Required if using any Sun keyboard, and for the window system.]

```
pseudo-device    ingres
```

[Sun MicroINGRES lock device.]

```
controller       mb0 at nexus ?
```

[**mandatory**. Main bus code.]

```
controller       ipc0 at mb0 csr all virt 0xeb0040 priority 2
```

[1st Interphase SMD disk controller. See *ip* (4).]

```
controller       ipc1 at mb0 csr all virt 0xeb0044 priority 2
```

[2nd Interphase controller.]

```
disk      ip0 at ipc0 drive 0
```

[1st disk on 1st Interphase controller.]

```
disk      ip1 at ipc0 drive 1
```

[2nd disk on 1st Interphase controller.]

```
disk      ip2 at ipc1 drive 0
```

[1st disk on 2nd Interphase controller.]

```
disk      ip3 at ipc1 drive 1
```

[2nd disk on 2nd Interphase controller.]

```
controller xyc0 at mb0 csr all virt 0xebee40 priority 2 vector xyintr 72
```

[1st Xylogics SMD disk controller. See *xy* (4).]

```
controller xyc1 at mb0 csr all virt 0xebee48 priority 2 vector xyintr 73
```

[2nd Xylogics controller.]

```
disk      xy0 at xyc0 drive 0
```
[1st disk on 1st Xylogics controller.]

```
disk      xy1 at xyc0 drive 1
```
[2nd disk on 1st Xylogics controller.]

```
disk      xy2 at xyc1 drive 0
```
[1st disk on 2nd Xylogics controller.]

```
disk      xy3 at xyc1 drive 1
```
[2nd disk on 2nd Xylogics controller.]

```
controller sc0 at mb0 csr 0x80000 priority 2
```
[1st SCSI controller on a Sun-2/120 or Sun-2/170.]

```
controller sc0 at mb0 csr vme busmem 0x200000 priority 2 vector scintr 64
```
[1st SCSI controller on a Sun-2/160.]

```
disk      sd0 at sc0 drive 0 flags 0
```
[1st disk on 1st SCSI controller.]

```
disk      sd1 at sc0 drive 1 flags 0
```
[2nd disk on 1st SCSI controller.]

```
tape      st0 at sc0 drive 32 flags 1
```
[1st SCSI tape.]

```
controller      sc1 at mb0 csr 0x84000 priority 2
```
[2nd SCSI controller.]

```
disk      sd2 at sc1 drive 0 flags 0
```
[1st disk on 2nd SCSI controller.]

```
disk      sd3 at sc1 drive 1 flags 0
```
[2nd disk on 2nd SCSI controller.]

```
tape      st1 at sc1 drive 32 flags 1
```

[2nd SCSI tape.]

```
device    ropc0 at mb0 csr 0xee0800
```

[**mandatory**. RasterOp chip. See *ropc* (4).]

```
device    sky0 at mb0 csr 0x2000 priority 2
```

[Sky Floating Point board in any Sun-1, Sun-2/120, or Sun-2/170.]

```
device    sky0 at mb0 csr vme busio 0x8000 priority 2 vector skyintr 176
```

[Sky Floating Point board in a Sun-2/50 or Sun-2/160.]

```
device    zs0 at mb0 csr all virt 0xeec800 flags 3 priority 3 # cpu
```

[CPU serial I/O ports. See *zs* (4).]

```
device    zs1 at mb0 csr all virt 0xeec000 flags 0x103 priority 3 # video
```

[Sun-2 Video Board ports. Required for Sun-2 keyboard and mouse.]

```
device    zs2 at mb0 csr 0x80800 flags 3 priority 3
```

[1st two serial I/O ports on 1st SCSI Board.]

```
device    zs3 at mb0 csr 0x81000 flags 3 priority 3
```

[2nd two serial I/O ports on 1st SCSI Board.]

```
device    zs4 at mb0 csr 0x84800 flags 3 priority 3
```

[1st two serial I/O ports on 2nd SCSI Board.]

```
device    zs5 at mb0 csr 0x85000 flags 3 priority 3
```

[2nd two serial I/O ports on 2nd SCSI Board.]

```
device    mti0 at mb0 csr all virt 0xeb0620 flags 0xffff priority 4 vector mtiintr 136
```

[Systech terminal MUX. See *mti* (4).]

```
device    ie0 at mb0 csr 0x88000 priority 3
```

[1st Sun-2 Ethernet Controller on a Sun-2/120 or Sun-2/170.]

```
device    ie0 at mb0 csr vme virt 0x0ee3000 priority 3
```

[1st Sun-2 Ethernet Controller on a Sun-2/50 or Sun-2/160.]

```
device    ie1 at mb0 csr 0x8c000 flags 2 priority 3
```

[2nd Sun-2 Ethernet Controller on a Sun-2/120 or Sun-2/170.]

```
device    ec0 at mb0 csr 0xe0000 priority 3
```

[1st 3COM Ethernet Controller. See *ec* (4).]

```
device  ie1 at mb0 csr vme busmem 0xe88000 priority 3
device    ec1 at mb0 csr 0xe2000 priority 3
```

[2nd 3COM Ethernet Controller. See *ec* (4).]

```
controller tm0 at mb0 csr all virt 0xeb00a0 priority 3 vector tmintr 96
```

[1st TAPEMASTER tape controller. See *tm* (4).]

```
controller tm1 at mb0 csr all virt 0xeb00a2 priority 3 vector tmintr 97
```

[2nd TAPEMASTER tape controller. See *tm* (4).]

```
tape      mt0 at tm0 drive 0 flags 1
```

[1st 1/2" tape drive on 1st TAPEMASTER controller.]

```
tape      mt1 at tm1 drive 0 flags 1
```

[1st 1/2" tape drive on 2nd TAPEMASTER controller.]

```
controller      xtc0 at mb0 csr all virt 0xebee60 priority 3 vector xtintr 100
controller      xtc1 at mb0 csr all virt 0xebee68 priority 3 vector xtintr 101
tape            xt0 at xtc0 drive 0 flags 1
tape            xt1 at xtc1 drive 0 flags 2
device    ar0 at mb0 csr 0x200 priority 3
```

[1st 1/4" tape drive. See *ar* (4).]

```
device    ar1 at mb0 csr 0x208 priority 3
```

[2nd 1/4" tape drive.]

```
device    gpone0 at mb0 csr vme busmem 0x210000 priority 3
```

[Sun Graphics Processor board.]

```
device    cgtwo0 at mb0 csr vme busmem 0x400000 priority 3
```

[Sun-2 color graphics interface. Required if gpone config line is present. See *cgtwo* (4s).]

```
device    cgone0 at mb0 csr 0xec000 priority 3
device  cgone0 at mb0 csr vme busmem 0x1ec00 priority 3
```

[Sun-1 Color Board. See *cgone* (4s).]

```
device    bwtwo0 at mb0 csr 0x700000 priority 4
```

[1st monochrome monitor on a Sun-2/120 or Sun-2/170. See *bwtwo* (4s).]

```
device    bwtwo0 at mb0 csr vme obio 0x0 priority 4
```

[1st monochrome monitor on a Sun-2/50 or Sun-2/160.]

```
device    bwone0 at mb0 csr 0xc0000 priority 3
```

[1st monochrome Sun-1 monitor. See *bwone* (4s).]

```
device    vp0 at mb0 csr 0x400 priority 2
```

[Ikon Versatec Board. See *vp* (4).]

```
device    vpc0 at mb0 csr 0x480 priority 2
```

[1st Systech Centronics/Versatec Board. See *vpc* (4s).]

```
device    vpc1 at mb0 csr 0x500 priority 2
```

[2nd Systech Centronics/Versatec Board.]

```
device    pi0 at mb0 csr 0xee2000
```

[Parallel input. Only used on Sun Models 100U and 150U, for keyboard and mouse.]

```
device    des0 at mb0 csr all virt 0xee1800
```

[Interface to the AMD8068 Data Ciphering Processor, a hardware implementation of the NBS Data Encryption Standard.]

```
device    tod0 at mb0 csr 0xee1000
```

[Time of day clock on the Sun-2/120 or Sun-2/170.]

```
device    tod0 at mb0 csr vme busmem 0x200800
```

[Time of day clock on the Sun-2/160 or Sun-2/50.]

3.11. Uninstalling the 2.2 Release

If you run into problems while running 2.2, you can back out the changes by using the tape which you got after running 2.2_backup. You then reconfigure your kernel, and can run as you were before. Proceed as follows.

NOTE *Optional software is not backed out during the uninstall.*

NOTE *If you are 'uninstalling' a network disk server, you must halt all diskless clients before proceeding.*

1. Load the backup tape, as described in the normal installation procedure above.
2. If you have edited */etc/rc.local* to contain a line for the *gpconfig* (8) command, that file should be re-edited to remove or comment out the line.

```
# /etc/shutdown -h
>b vmunix -s
#/etc/mount /dev/nrtape0 /usr
#mt -f /dev/nrtape0 rew
# tar xvpf /dev/nrtape0
```

   This takes about 30 minutes.

4. Reconfigure your kernel.

4

Fixed Software Bugs

Fixed Software Bugs

This chapter briefly presents the bugs which are fixed in Release 2.2. The bugs are sorted under general headings: Compiler, Graphics, Kernel, Network, Utilities, Windows, Documentation, and Miscellaneous. As noted in the "Introduction", Release 2.2 is primarily concerned with general system bug fixes, To take advantage of the bug fixes in the libraries, programs should be recompiled.

4.1. Language Processors

Assembler

The assembler has been corrected to accept compiler lines in the `symbolname =.` form.

C Compiler

The C compiler now compiles the assignments of address register variables to the bit fields. Previously, the 2.0 compiler failed to do this correctly.

Compiler expressions in the form `<expr>%1 == 0` no longer generate an illegal instruction `tstl #0`.

The C compiler no longer produces a fatal error message when compiling a structure declaration with too many initial values.

The C compiler has been fixed so that array subscripts no longer make it loop. In Release 2.0, array subscript fragments could make the C compiler loop and the message `compiler error: expression causes compiler loop: try simplifying` would appear.

FORTRAN

FORTRAN DO-loops with floating point index variables have been corrected. Previously, these caused compile time errors when the `-fsky` option was used.

FORTRAN arithmetic expressions containing array references now compile correctly when using the `-fsky` option. Previously, these compiled incorrectly resulting in a segmentation fault at run time.

The `f77` compiler generates large offsets in base displacement addressing of common blocks resulting in assembly time errors. This bug has been fixed.

The bad read of list-directed floats has been fixed. `list` directories that read statements with floating point arguments use to produce incorrect results.

A memory allocation error in the Fortran `optomizer` has been fixed.

A syntax error in `DATA` statement that the compiler uses caused an abort. This has now been fixed.

The `-Nx` option must come before the file name(s), otherwise the flag is used by the loader. This is not a bug in `f77` but rather a clarification.

4.2. Graphics

CGI

If a viewport is made smaller than a clip rectangle and then enlarged, the smaller viewport is also enlarged. Previously, when a viewport was made smaller than a clip rectangle, then made larger, the smaller viewport remained.

`circular_arc.3pt` returns an error message when given collinear or coincident points. Previously, there was no error message when this occurred.

When five view surfaces are opened, and then one is closed, the others stay open. In Release 2.0, the other view surfaces could not be reopened.

The CGI package did not free up the memory which it had allocated once it was done with it.

Now when the expression within the function `inquire_text_extent` is set to the value `string` (which indicates raster text), the function no longer returns a message indicating that there is no text. Previously, the function returned a message indicating that there was no text in this file when in fact text did exist.

Bundled attributes now automatically rescale. They did not in the previous release.

Pixrects

`pr_rop` is now working for 1-bit and n-bit memory pixrects. The bug used to trash the text on a retained pixwin (on a Sun-2/160 color window) when the window redisplayed.

In order to gain full screen access, a program running in a window called `fullscreen_init` calls to `pw_replrop`. This program will now expand to full screen access. Previously, it would clip to the boundries of the original pixwin.

Graphics Processor

Circular arcs of width one now display on the Graphics Processor.

The Graphics Processor no longer hangs if the number of polygon vertices is higher than 25.

The hidden surface removal now works using the Graphics Processor without the Graphics Buffer. Previously, there appeared to be a hardware buffer even when it was not present.

A Graphics Processor device driver declaration has been added to `/usr/include/pascal/devincpas.h`.

`3-D polygons` in temporary segments drawn on a `gpl` view surface are now shaded correctly. Previously, due to a bug in `/usr/lib/libcore.a`, the view surface would sometimes shade unevenly.

4.3. Kernel

Driver

A terminal on `/dev/ttya` that is being used as a console will no longer hang after the `telnet` gives the login message.

The old and new `tty` line disciplines now control the `tty` line flow in `raw` mode, `cbreak` mode, and `cooked` mode. In Release 2.0, old and new `tty` line disciplines would control the `tty` line flow only in `raw` mode or `cbreak` mode, not in `cooked` mode.

Sun machines with Sun Ethernet boards not plugged into a network generated `ie0: ethernet jammed` messages every ninety seconds if `ifconfig` had been run. Running `ifconfig ie0 down` will now solve the problem.

`KIOCSETKEY ioctl` no longer interacts with the keyboard translation tables in such a manner that random locations in the kernel are trashed if the user changes the interpretation of keys.

Heavily loaded servers sometimes locked up with processes hung waiting for disk I/O on a Xylogics controller. This was a hardware problem with the Xylogics 450 disk controller. The driver was modified to compensate for it.

NSF

NFS client programs no longer cause the NFS server to panic with an address error. The server and client have been modified to check for negative file offsets.

A bad block in a directory caused a server to panic when using the system call `readdir`. This bug has been fixed.

Miscellaneous

When trying to run very large programs with very large text sizes and small data sizes, the system used to crash. Now, the process fails to execute.

The bogus `trace/BPT` trap has been corrected. Previously, processes were intermittently dying during `forks`.

The kernel configuration file `/usr/sys/conf/ND120` has been corrected. It had an entry for the pseudo-device `dtop` which was incorrect.

A line for `ie1` for VME-based systems has been added to the generic kernel making it possible to use the generic kernel on a VME-based gateway machine.

The kernel now recognizes bad addresses sent by user's programs such as games. Previously, the kernel did not check the PTEs carefully and the system would panic.

A line has been added to the GENERIC configuration file so a medium resolution color monitor will work with a Sun-2/160.

The Systech mux board driver has been fixed. A bug in the driver caused the system to crash under certain error conditions, most often occurring on heavily loaded systems.

The system no longer panics when it allocates a new `inode` in a cylinder group that is very full.

4.4. Network

Protocol

The bug in `UDP RPC` programs that call to `recvfrom` after a `fork` has been fixed.

`TCP RPC` programs no longer hang if the server crashs during a remote procedure call.

Gateway system's TCP code decremented an internet packet time-to-live counter by too much, limiting the maximum number of hops of a `TCP` packet to four. This has been fixed.

Yellow Pages

Erroneous comparisons of usernames in the password file has been corrected. In Release 2.0, it caused `yppasswd` to think that names which were only similar were in fact the same.

There was a memory leak in `ypbind` and `ypserv`; the program allocated memory but did not free it. The bug was in the C library.

`yp_next` no longer fails if the input key is zero length (null).

`/etc/yp/ypinit` has been fixed. Previously, it was unable to detect an unset host name or domain name.

The `yp_bind` function has been corrected to allow the code to work correctly when compiled on a DEC VAX.

Miscellaneous

The bug in the `tftp` server which caused file transfers to timeout has been fixed.

Running out of process table space made a machine unable to accept new network connections making programs such as `rsh` and `rlogin` timeout. Under these circumstances, the internet daemon would eat up all the rest of the CPU cycles.

4.5. Utilities

The `vi` and `ex` command `:so` (source) has been fixed. Using the `source` command to read and execute a file of `vi` commands caused `vi` to loop.

The `vtroff` formatter has been fixed. Previously, it used to generate floating exception messages when asked to draw boxes via `-ms .BX` commands.

Miscellaneous

The error response of `sendmail` has been corrected. The error response of `sendmail` treated a temporary error like a permanent error. For example, the Release 2.0 responded `Deferred: Bad file number` on any temporary error involving `SMTP`.

`dcheck` no longer produces bogus errors after encountering a real one.

The `man` command will now give the user a usage string citing the valid options instead of crashing.

The `/bin/test` command, used from the Bourne shell, now recognizes the command-line arguments `-b`, `-c`, `-g`, `-k`, and `-x`.

`lpd` will now parse `+` entries in `/etc/hosts.equiv`.

The `ftp` bug which used to cause files to overwrite has been corrected.

The date computation in `makedbm` has been fixed.

A spurious `not on export list` error could result from a remote mount from a gateway.

4.6. Windows

Suntools

A long line of input typed to a Suntools shell window no longer locks up that window.

The double `pr_destroy` in `libsuntool/gfxsw.c` has been corrected.

In the `panel_public.c` package in `libsuntool`, in a routine named `shrink_to_fit()`, two variables have to be initialized to zero. In the function `panel_get()`, a third argument has to be defined.

4.7. Miscellaneous Software Fixes

In Release 2.0, `sysdiag`'s account name appears in `/etc/passwd` with a blank password field. This allows over-the-wire diagnostics but also allows a user to log in as `sysdiag` and become `root`. If you wish to prevent this, put a "*" in the password slot of `passwd` file.

5

Errata Pages for 2.0 Manuals

Errata Pages for 2.0 Manuals

The following pages list errata from the 2.0 *Programmer's Reference Manual for SunCore* (Part Number: 800-1165), *Programmer's Reference Manual for SunCGI* (Part Number: 800-1166), *System Administration for the Sun-Workstation* (Part Number: 800-1150) and *Installing Unix on the SunWorkstation* (Part Number: 800-1158).

Errata in the 2.0 Release

of the

Programmer's Reference Manual for SunCore

Page(s)	Comments
B-3	In Section B-2, two additional device specifications for the GP board are necessary. *gp1dd* A Sun-2/160 graphics display with a Graphics Processor option. *gp1pixwindd* A color graphics window within the Suntools window environment running on a Sun-2/160 color graphics display with a Graphics Processor option. In addition, the sentence following these specifications that lists devices capable of hidden surface removal should include the new GP devices.

Errata in the 2.0 Release

of the

Programmer's Reference Manual for SunCGI

Table 5-1 **SunCGI** *Errata*

Page	*Comments*
2-3	Section 2.1.2 should state that many failures during view surface initialization produce error 11, ENOWSTYP. For example, opening a device surface type `PIXWINDD` instead of `CGPIXWINDD` on a color pixwin, or using `CG2DD` when the */dev/cgtwo** surface is being used by `suntools`.
2-5	In Table 2-2, an additional device specification for the GP board is necessary. *GP1DD* for Sun-2/160 graphics display with optional Graphics Processor
2-16	Section 2.3.2.1 should contain the following explanation: The default state is VDC_EXTENT. If clipping is not NOCLIP, output primitives are clipped to either the clip rectangle (if `cflag` equals CLIP_RECTANGLE) or the full extent of VDC space (if `cflag` equals VDC_EXTENT). The extent of VDC may be set with the `vdc_extent` function.

2-19	In Section 2.3.3.4, the *extent* argument is of the enumerated type `Cexttype`. The function specification is correct, but the text makes reference to an argument `clear_extent` incorrectly of type `Cclip`. The definition of type `Cexttype` is:
	`typedef enum {`
	`CLIP_RECT,`
	`VIEWPORT,`
	`VIEWSURFACE`
	`} Cexttype;`

2-19	In Section 2.3.3.5, the discussion of `set_error_warning_mask` should contain the following explanation: SunCGI defines no errors as FATAL. (POLL and INTERRUPT actions are therefore the same). The error number is always returned. A message is printed unless the action is NO_ACTION.

Table 5-1 **SunCGI** *Errata— Continued*

Page	*Comments*

2-20	In Section 2.4.3 The `sig_function` argument to `set_up_sigwinch` is a pointer to a function, The complete specification should be:
	
	`Cerror set_up_sigwinch(name, sig_function)`
	`Cint name;`
	`Cint (*sig_function)(); /* signal handling function */`
	
	The `sig_function` argument is called with a single argument: the name of the view surface with which it is associated by the call to `set_up_sigwinch`. This allows more than one view surface to share the same `sig_function`, and differentiate which view surface needs redisplay.

3-13	Section 3.2.7 the discussion of `bitblt_source_array` should contain the following explanation: `pixsource` and `pixtarget` are pointers to pixrects which must already be created by the user with `mem_create` (see the *Programmer's Reference Manual for SunWindows*). These pixrects must be the same depth as the view surface: 1-bit deep on a monochrome device, 8-bit on a color device. The source area of the view surface associated with name is saved into `pixsource` (at 0,0), possibly NOT-ed, depending on the drawing mode. The target area, after `pixsource` is applied to it, is read into `pixtarget` pixrect (at 0,0).
3-14	In Section 3.2.8, the second paragraph should say: `pixpat` is a pointer to a pixrect which must be created and initialized with the pattern by the application program. `pixtarget` is a pointer to a pixrect (with same depth as device) which must already be created by the user, using `mem_create`. The target area, after `pixpat` is applied to it, is read into the `pixtarget` pixrect (at 0,0).
3-14	In Section 3.2.9, the first paragraph should say: `bitblt_patterned_source_array` replicates (using the current drawing mode) the pattern stored in pixpat to fill the area of the view surface determined by `ox, oy` and `dx, dy`. The source area of the view surface is read into the pixrect pointed to by `pixsource` (which must already be created by the user with same depth as device) at 0,0. The replicated pattern array is AND-ed into the pixsource, and possibly NOT-ed, depending on the drawing mode. The resulting pixrect is copied to the view surface at `ox, oy`, using the current drawing mode. The target area, after the copy, is read into the `pixtarget` pixrect. If the replicated pattern array overlaps with the source array on the screen, the visual result depends on the current drawing mode.
4-11	The sentence in Section 4.4.2.3 which says: All nonzero entries in `colorind` are set to 1. should say: For monochrome view surfaces, all nonzero entries in `colorind` are treated as 1 when used.

Table 5-1 **SunCGI** *Errata—Continued*

Page	*Comments*

4-12	New Section 4.4.2.6:
	
	`Cerror pattern_with_fill_color(flag)`
	`Cflag flag; /* ON to use nonzero pattern elements as fill color */`
	
	Binary patterns are a **SunCGI** extension that allow the same pattern to be applied in different colors, without redefining the pattern array. `pattern_with_fill_color` sets a non-standard CGI state *pattern with fill color*. The default for `flag` is OFF and each color value in a pattern table entry is used verbatim, as in standard CGI. When a pattern is used while `pattern_with_fill_color` is ON, the pattern is considered to be a 2D array of flags: where the pattern element is nonzero, the current fill color is used, instead of the actual value of the pattern element. (Where `flag` is zero, a zero color index is used, just as when the flag is OFF.)

4-20	Section 4.6.1 should say: The default color lookup table size for a color device has 8 entries. The minimum and maximum color table entries are treated specially by pixwins and hence by **SunCGI**. If they are set to be the same value, the user's values for these two entries are *both* ignored: they revert to the "inverse" of the normal values: entry 0 becomes white, the maximum entry becomes black.
5-1	The following paragraphs are supplemental to the introduction of logical input devices. Each logical input device can be used in one of three ways, distinguished by the three EVENT input device states. To use input synchronously, the application program should call request_input. The program blocks until the operator fires a trigger associated with this device (or a timeout occurs). If timeout is −1, the request will wait forever. To receive input asynchronously, the `initiate_request` function should be used. This function initializes the device so that the measure from the next trigger activation will be recorded in the request register. The program does not block in REQUEST mode. Until the trigger fires, `sample_input` may be called to get the current measure of the input device. After the trigger fires, `get_last_request_input` will then return the request register (value) and status. To use queued input the `enable_events` function should be called. This function initializes the device so that trigger activations will result in input events being enqueued onto the event queue. Calls to `await_event` will return the first event in the event queue. If the event queue is EMPTY, it will wait for a trigger (or until timeout). A call to `disable_events` will terminate the queuing of input events. When initializing the locator or keyboard device, the the `xypt` field of the `Cinrep` structure must point to a `Ccoor` allocated by the application program.

Table 5-1 **SunCGI** *Errata—Continued*

Page	*Comments*
5-3	The program example in Section 5-1 is wrong. It should be replace by the following example:

```
#include <cgidefs.h>
#define TEN_SECONDS (10 * 1000 * 1000) /* timeout is in microseconds */

main()
{
        Cawresult stat;
        Ccoor point;
        Cinrep ivalue;
        Cint name;
        Cint trig;
        Cvwsurf device;

        device.dd = PIXWINDD;
        point.x = 16384; /* put cursor in the middle of the view surface */
        point.y = 16384;
        ivalue.xypt = &point;

        open_cgi();
        open_vws(&name, &device);

        initialize_lid(IC_LOCATOR, 1, &ivalue);
        /* associate locator with mouse button 1 */
        associate(2, IC_LOCATOR, 1);
        /* track with printer's fist: move cursor (fist) to initial point */
        track_on(IC_LOCATOR, 1, 1, (Ccoorpair *)0, &ivalue);
        /* wait up to ten seconds for input */
        request_input(IC_LOCATOR, 1, TEN_SECONDS, &stat, &ivalue, &trig);
        if (stat == VALID_DATA)
                printf(" trigger activated at %d %d 0,
                        ivalue.xypt->x, ivalue.xypt->y);
        else
                printf(" trigger not activated 0);
        /* shut device off */
        dissociate(2, IC_LOCATOR, 1);
        release_input_device(IC_LOCATOR, 1);

        sleep(10);
        close_vws(name);
        close_cgi();
}
```

5-5	Trigger 6, LOC_STILL, has been added. This trigger fires when the mouse doesn't move for about 1/5th of a second (or more).

Table 5-1 **SunCGI** *Errata— Continued*

Page	*Comments*

5-11	The device specification arguments for `get_last_requested_input` are wrong. The complete specification should be consistent with the other input routines.
	`Cerror get_last_requested_input(devclass, devnum, valid, sample)`
	`Cdevoff devclass; /* device class and number */`
	`Cint devnum;`
	`Clogical *valid; /* device status */`
	`Cinrep *sample; /* device value */`

F-2	The `name` and `pw` arguments for `open_cgi_pw` are wrong. The complete specification should be:
	`Cerror open_cgi_pw(pw, desc, name)`
	`struct pixwin *pw; /* pixwin */`
	`Ccgiwin *desc; /* CGI pixwin descriptor */`
	`Cint *name;`

F-2	All cgipw input and output primitives use screen (pixel) coordinates, for compatibility with pixwins. As described in *Programmer's Reference Manual for SunWindows*, pixel coordinates have the upper left corner as the origin for compatibility with Pixwins. Table F-1 should not contain reference to `track_on` or `track_off`; the functions `cgipw_track_on` and `cgipw_track_off` do not exist. The bottom of page F-2 should say `desc` is a pointer to the pixwin descriptor filled in by the `open_cgi_pw` function.

Table 5-1 SunCGI *Errata— Continued*

Page	*Comments*
G-2	The FORTRAN example program is wrong. Here is a version that works.

```
      program test

      parameter (ibignum=256)

      integer name
      character screenname* (ibignum)
      integer screenlen
      character windowname* (ibignum)
      integer windowlen
      integer windowfd
      integer retained
      integer dd
      integer cmapsize
      character cmapname* (ibignum)
      integer cmaplen
      integer flags
      character ptr* (ibignum)
      integer noargs
      integer xc(10),yc(10),n
      integer xc2(2),yc2(2)
      data xc /0,-10,-1,-1,-15,15,1,1,10,0 /
      data yc /0,0,1,20,35,35,20,1,0,0 /
      data xc2 /-12,12/
      data yc2 /33,33/

      call cfopencgi()
      dd = 4
      call cfopenvws(name,screenname,screenlen,windowname,windowlen,
     +  windowfd,retained,dd,cmapsize,
     +  cmapname,cmaplen,flags,ptr,noargs)
      call cfvdcext(-50,-10,50,80)
      n = 10
      call cfpolyline(xc,yc,n)
      n = 2
      call cfpolyline(xc2,yc2,n)
      call sleep(10)
      call cfclosecgi()
      call exit()
      end
```

G-3 The calling sequence for `bitblt_patterned_source_array` should say

```
bitblt_patterned_source_array(pixpat, px, py, pixsource,
      sx, sy, pixtarget, rx, ry, ox, oy, dx, dy, name)
```

Table 5-2 *Functions not compatible with CGIPW mode*

Function	*Discussion*
`vdc_extent`	cgipw's VDC space is identical to screen space
`device_viewport`	use `pw_region` prior to `open_cgi_pw`
`clip_indicator`	when *cflag* is CLIP_RECTANGLE
`clip_rectangle`	Instead, use `pw_region` prior to `open_cgi_pw`
`clear_control`	All clear extents are identical
`open_vws`	Use `open_cgi_pw`
`close_vws`	Use `close_cgi_pw`
`close_cgi`	Use `close_pw_cgi`

Errata in the 2.0 Release

of the

System Administration for the SunWorkstation

Page(s)	Comments
2	In section 1.1 of the tutorial on*FSCK-The Unix File System Check Program*, the default block size for the 2.0 Release should be 4K with a hardcoded default block size of 4096.
4-24	In Section 4.3 when mail is sent, `sendmail` checks the files `/etc/usr/hosts.equiv` and `/usr/lib/mailhosts` for the name of the host to whom the mail is being sent. If neither of these files recognize the name, sendmail forwards the mail to the mailhost machine.

Errata in the 2.0 Release

of the

Installing Unix on the Sun Workstation

Page(s)	Comments
3-2	If you have a 1/4" tape and the > prompt returns instead of **Boot**, the tape may need retensioning. Try entering: **>b** *tape* `(0,0,100)` Boot: *tape* `(0,0,0)` Boot:
6-10	If your standalone system is not attached to the net, or you are not using the yellow pages server, you will need to bypass the yellow pages in order to continue your installation of UNIX. Move `/etc/ypbind` to `/etc/ypbind-` by using the following command: **mv /etc/ypbind /etc/ypbind-**

Errata in the 2.0 Release

of the

Installing Unix on the Sun Workstation

Insert Sheets for Chapter 10

5.1. Converting Diskful Workstations to use NFS

This section describes how to convert a diskful workstation to use NFS.

The NFS enables diskful workstations to mount directories from other machines, thus reducing local disk storage needs and allowing them to share common resources.

The conversion process in this section is designed specifically for diskful workstations. To convert servers and diskless clients, use the instructions provided earlier in this chapter.

This procedure uses examples from an upgrade where the diskful workstation named *topnotch* was upgraded to use NFS, and the system *vfree* is its NFS server.

Use the following steps:

(1) Record Vital Information

Before you can proceed, you need to know:

Your local machine name and internet address, your NFS server's machine name and internet address, your user id number (uid) from */etc/passwd*, and your domain name if you are going to use the yellow pages . See Chapter 1 for descriptions of these items (except uid). For example:

```
topnotch      192.9.4.53          [These are examples only
vfree         192.9.4.54          do not use this information]
121
```

(2) Dump

Dump your entire disk(s).

(3) Follow Install Manual Instructions

Follow the instructions in Chapters 1 through 5 up until the point it tells you to run *setup*. Then, instead of running *setup* go to the next step. Be sure to boot the system in single user mode first:

```
>b vmunix -s
```

(4) Edit *fstab*

Edit your */etc/fstab* file. The entire file should look like the one below. Be sure to substitute the proper value for *disk*: Note that you may have to use the "ed" editor, as "vi" may not be available (See *ed*(1)).

```
/dev/disk0a /      4.2 rw 1 1
/dev/disk0g /private 4.2 rw 1 2
nfs_server_name:/usr /usr nfs rw,hard 0 0
```

Be sure to replace *nfs_server_name* with the name of your NFS server.

(5) Edit *rc.local*

Edit the file *rc.local*. Replace *noname* with your machine name, and *nodomainname* with your domain name. If you do not use domains, leave the *nodomainname* alone. For example:

```
/bin/hostname topnotch                  [Example - do not use this]
/bin/domainname nodomainname            [Note no domainname]
```

(6) Edit *hosts*

Add two lines with the following to your */etc/hosts* file:

internet_address machinename
NFS_server_internet_address NFS_server_machinename

for example:

```
192.9.4.53      topnotch        [This is an example only:
192.9.4.54      vfree           do not enter this data]
```

(7) Add to Server's /etc/hosts

Make sure your internet address and machine name are in the NFS server's */etc/hosts* file.

(8) Run *newfs*

Do the following */etc/newfs* (remember to substitute the proper value for *disk*):

```
newfs /dev/rdisk0g
```

(9) Mount *diskg*

Enter the following *mount* command:

```
/etc/mount /dev/disk0g /private
```

(10) Make new directories

Use *mkdir* to create the following directories:

```
/private
/private/usr
/private/usr/adm
/private/usr/crash
/private/usr/preserve
/private/usr/spool
/private/usr/tmp
/private/usr/lib
/private/usr/spool/mqueue
```

(11) Move ypbind

Move */etc/ypbind* to */etc/ypbind-*

(12) Reboot your machine

Now, prepare your machine for halting, by syncing the disks several times take it down, and boot multiuser. Note that the default boot command (>b) should work.

Try out a few things to make sure they work, before you begin the next steps and start restoring your files.

(13) Create files

Use *touch* to create the following new files:

```
/private/usr/adm/lastlog
/private/usr/adm/messages
/private/usr/adm/msgbuf
/private/usr/adm/shutdownlog
/private/usr/adm/usracct
/private/usr/adm/wtmp
```

(14) Restore files

From the dump tape you made earlier, restore your home directory files to */usr2*, which is mounted on the 'g' partition. Remember, files dumped as */usr/john* must be restored as */usr2/john*.

Next, restore the following files individually:

Restore */usr/lib/crontab* to */private/usr/lib/crontab*.

Restore */usr/lib/aliases* to *private/usr/lib/aliases*.

Restore all of */usr/spool* to */private/usr/spool*.

(15) Edit *crontab*

Edit the */private/usr/lib/crontab* file you restored and make three changes: 1) remove or comment out any *find* command that starts at "/" — the root. 2) remove or comment out any 'calendar' entry. 3) Change a line in *crontab* as follows:

Change:

```
15 4 * * * find /usr/preserve -mtime +7 -a -exec rm -f {} \;
```

Change it by adding a '/' (slash) at the end of preserve:

```
15 4 * * * find /usr/preserve/ -mtime +7 -a -exec rm -f {} \;
```

(16) Install *sendmail*

Now, you must install *sendmail* as described in the "System Administration Manual". Note that the files */usr/lib/sendmail.main.cf* and */usr/lib/sendmail.subsidiary.cf* should already be in /usr/lib.

(17) Edit *passwd*

Next, edit your */etc/passwd* to include the uid that you recorded above, and make sure that the home directory field for each user is */usr2* instead of */usr*. Note that eventually you will want to assign passwords to both root and yourself; you can do this with *passwd*.

For example, make sure the field looks like this:

```
zippy:OVceErnaqI:1492:10:Zippy the Hacker:/usr2/zippy:/bin/csh
```

(18) Restore more files

Now, restore the following files. Note that, unlike the files restored above, these return to their original locations. The files are: */etc/printcap*, */etc/ttys*, */etc/ttytype*, */etc/remote* (you may have others).

If you do not plan to use the yellow pages, then also restore the following files: */etc/group*, */etc/services*, */etc/protocols*, */etc/networks*, and */etc/hosts*.

Now, restore other files peculiar to your disk. For example, */usr/lib/emacs* should be restored to */usr*. Note however, that almost all programs must be recompiled; 1.x binaries do not normally run on a 2.0 system.

6

Insert Pages for 2.0 Reference Manuals

Insert Pages for 2.0 Reference Manuals

The following pages are reference manual pages for *Commands Reference Manual for the Sun Workstation* (Part Number: 800-1172) and the *System Interface Manual for the Sun Workstation* (Part Number: 800-1173).

How you handle the pages is up to you. We recommend inserting them directly into your 2.0 manuals (copying them if necessary); they are numbered accordingly.

New Pages

gpone(4S)	—	Sun-2/160 color graphics processor interface
gpconfig(8)	—	initialize the Graphics Processor
nfsmount(2)	—	revised
mount(8)	—	revised
st(4S)	—	revised
toolplaces(1)	—	revised

NAME

nfsmount – mount an NFS file system

SYNOPSIS

```
#include <netinet/in.h>
#include <nfs/nfs.h>

nfsmount(addr, fh, dir, flags, rsize, wsize)
struct sockaddr_in *addr;
fhandle_t *fh;
char *freq;
int flags;
int rsize;
int wsize;
```

DESCRIPTION

Nfsmount mounts an *NFS*(4) file system on the directory *dir*. *Addr* is the *UDP*(4) address of the server that owns the file system to mount. *Fh* is a file handle, obtained from the server, to identify the root directory on the server that is being mounted.

The *flags* argument contains mount flag bits. The *NFSMNT_RDONLY* flag tells whether the file system can be written on; if it is 0 writing is allowed, if non-zero no writing is done.

The *NFSMNT_SOFT* flag determines whether the remote file system is mounted hard or soft. A soft mount causes an error to be returned when a remote access times out. Hard mounts cause the access to retry until the server responds. A value of 1 indicates a soft mount.

The *NFSMNT_RSIZE* and *NFSMNT_WSIZE* flags tell whether the *rsize* and *wsize* parameters are valid. If a flag is set the coresponding parameter is used to set the number of bytes sent in a read or write operation.

RETURN VALUE

Nfsmount returns 0 if the action occurred, –1 if some error occurred.

ERRORS

Nfsmount will fail when one of the following occurs:

[EPERM] The caller is not the super-user or the path name given for *dir* contains characters with the high bit set.

[ENAMETOOLONG] The path name for *dir* is too long.

[ELOOP] *Dir* contains a symbolic link loop.

[ETIMEDOUT] The server at *addr* is not accessable. This can only happen if the *hard* flag is set.

[ENOTDIR] A component of the path prefix in *dir* is not a directory.

[EBUSY] Another process currently holds a reference to *fh*.

SEE ALSO

mount(2), unmount(2), mount(8)

NAME

gpconfig – initialize the Graphics Processor

SYOPNSIS

/etc/gpconfig gpunit [[-b] [-f] *fbunit* ...]

DESCRIPTION

gpconfig binds *cgtwo* frame buffers to the Graphics Processor (GP), and loads and starts the appropriate microcode in the GP. For example, the command line:

/etc/gpconfig gpone0 cgtwo0 cgtwo1

will bind the frame buffer boards *cgtwo0* and *cgtwo1* to the Graphics Processor *gpone0* . The devices */dev/gpone0a* and */dev/gpone0b* will then refer to the combination of *gpone* and *cgtwo0* or *cgtwo1* respectively.

The same *cgtwo* frame buffer cannot be bound to more than one GP.

All *cgtwo* frame buffer boards bound to a GP must be configured to the same width and height.

Note: The *gpconfig* command should be placed in the file */etc/rc.local* if the GP is used regularly.

Note: It is inadvisable to run the *gpconfig* command while the GP is being used. Unpredictable results may occur. If it is neccessary to change the frame buffer bindings to the GP (or to stop using the GP altogether), bring the system down gently, boot single user, and edit the *gpconfig* line in the */etc/rc.local* file and bring the system back up multiuser.

OPTIONS

–b Configure the GP to use the Graphics Buffer as well. **Note:** Currently only one GP / frame buffer binding is allowed to use the graphics buffer at a time.

–f Indicates that the next frame buffer specified is to be used for */dev/fb* as well.

FILES

/dev/cgtwo[0-9]
/dev/fb
/dev/gpone[0-3][abcd]
/etc/gpicg2.1024.ucode
/etc/gpicg2.1152.ucode
/etc/rc.local

SEE ALSO

cgtwo(4S), gpone(4S)

NAME

mount, umount – mount and dismount filesystems

SYNOPSIS

/etc/mount
/etc/mount –p
/etc/mount –a[fv][t type]
/etc/mount [**–frv**][**to** type options] [fsname] [dir]

/etc/umount [**–av**] [fsname | dir] ...

DESCRIPTION

Mount announces to the system that a filesystem *fsname* is to be attached to the file tree at the directory *dir*. The directory *dir* must already exist. It becomes the name of the newly mounted root. The contents of *dir* are hidden until the filesystem is unmounted. If *fsname* is of the form host:path the filesystem type is assumed to be *nfs*(4).

Umount announces to the system that the filesystem *fsname* previously mounted on directory *dir* should be removed. Either the filesystem name or the mounted-on directory may be used.

Mount and *umount* maintain a table of mounted filesystems in */etc/mtab*, described in *mtab*(5). If invoked without an argument, *mount* displays the table. If invoked with only one of *fsname* or *dir* mount searches */etc/fstab* for an entry whose *dir* or *fsname* field matches the given argument. For example,

mount /usr

and

mount /dev/xy0g

are shorthand for

mount /dev/xy0g /usr

if this line is in */etc/fstab*

/dev/xy0g /usr 4.2 rw 1 1

MOUNT OPTIONS

–a Attempt to mount all the filesystems described in */etc/fstab*. In this case, *fsname* and *dir* are taken from */etc/fstab*. If a type is specified all of the filesystems in */etc/fstab* with that type will be mounted.

–o The next argument is a string that specifies mount options. Valid options are: ro, rw, quota, noquota, hard, soft, rsize=n, and wsize=n. Hard, soft, rsize=n and wsize=n only make sense on *nfs*(4) filesystems. Options are separated by commas. The options *ro* and *rw* stand for read-only and read-write; *rw* is the default. Since quotas are not implemented, *noquota* is the default. With a hard remote mount, *mount* tries forever if the *mountd*(8c) server does not respond. Once the filesystem is mounted, access requests will retry forever if the *nfsd*(8) server does not respond. Hard is the default. With a soft remote mount, if the *mountd*(8c) server does not respond, *mount* forks a background copy to retry forever. Once the soft mount completes, access requests will fail with [ETIMEDOUT] if the *nfsd*(8) server does not respond. The *rsize=n* and *wsize=n* options can be used to set the number of bytes in a read or write operation on *nfs*(4) filesystems.

–r Mount the specified filesystem read-only. This is a shorthand for:

mount –o ro fsname dir

Physically write-protected and magnetic tape filesystems must be mounted read-only, or errors will occur when access times are updated, whether or not any explicit write is attempted.

–t The next argument is the filesystem type. The accepted types are: 4.2, nfs, and pc; see *fstab*(5) for a description of the legal filesystem types.

–f Fake a new */etc/mtab* entry, but do not actually mount any filesystems.

–p Print the list of mounted filesystems in a format suitable for use in */etc/fstab*.

–v Verbose — *mount* displays a message indicating the filesystem being mounted.

UMOUNT OPTIONS

-a Attempt to unmount all the filesystems currently mounted. In this case, *fsname* is taken from */etc/mtab*.

-v Verbose — *umount* displays a message indicating the filesystem being unmounted.

EXAMPLES

mount /dev/xy0g /usr	mount a local disk
mount -ft 4.2 /dev/nd0 /	fake an entry for nd root
mount -at 4.2	mount all 4.2 filesystems
mount -t nfs serv:/usr/src /usr/src	mount remote filesystem
mount serv:/usr/src /usr/src	same as above
mount -o hard serv:/usr/src /usr/src	same as above but hard mount
mount -p > /etc/fstab	save current mount state

FILES

/etc/mtab	mount table
/etc/fstab	filesystem table

SEE ALSO

mount(2), nfsmount(2), unmount(2), fstab(5), mountd(8c), nfsd(8c)

BUGS

Mounting filesystems full of garbage will crash the system.

No more than one user should mount a disk partition "read-write" or the file system may become corrupted.

NAME

gpone – Sun-2/160 color graphics processor interface

SYNOPSIS

gpone0 at mb0 csr vme busmem 0x210000 priority 3

DESCRIPTION

The *gpone* interface provides access to the optional GP graphics processor board.

gpone supports the *FBIOGTYPE* ioctl which a program can use to inquire as to the characteristics of the display device; see *fbio* (4s).

gpone supports the *FBIOGPIXRECT* ioctl which allows SunWindows to run on it; see *fbio* (4s).

The hardware consumes 64 kilobytes of VME bus address space. The GP board starts at standard address 0x210000 and must be configured for interrupt level 3.

FILES

/dev/gpone[0-3][abcd]

SEE ALSO

fbio (4s), *mmap* (2)

A

Appendix H: for the *Programmer's Reference Manual for SunWindows*

Appendix H: for the *Programmer's Reference Manual for SunWindows*

The following pages are to be inserted as an appendix to the *Programmer's Reference Manual for SunWindows.*

The **SunWindows** software provides a rich basis for designing custom user-interfaces. With the window system, the programmer can share the screen's "real estate" amongst several virtual screens called windows. Each of these windows is essentially its own state machine, and so windows can be sensitive to programmer specified input events and take any number of actions in response to the events. Most of the time an input eventually results in some change in the window itself — sometimes drawing or writing something new in the window, and other times changing the window's state with respect to the entire window system.

With software this sophisticated, sometimes it is hard at first to get a handle on the data structures and library routines that make up the windows system. We feel that programming examples are the best way to show how to access specific data structures and call specific routines. Also, programming examples often give a good sense of what **SunWindows** is intended to do. The *Programmer's Tutorial to SunWindows* gives several good examples for beginners.

This document is meant as a supplement to the *Tutorial* with more advanced examples. None of the programs are useful except as examples of a specific facet of **SunWindows.** They are designed to be short and to the point. The program listings included cover the following subjects:

- How to use timers in a `tool_select()` call `(timertool.c)`
- How to change icons dynamically `(changing_icon.c)`
- How to find newly exposed area in a window after window damage occurs `(rectlists.c)`
- How to customize subwindow layout in a window `(layout.c)`
- How to manipulate the colormap and color bit-planes in a color window `(onewaytomakeacolorbar.c, anotherwaytomakeacolorbar.c, animation.c)`
- How to use the `tool_parse_all()` call `(parse_all.c)`
- How to find the root window file descriptor from any window `(findroot.c)`

Each program has approximately fifty lines of code (not counting comment lines, of course), many of which are copied from one program to the next. We suggest typing in the programs to help familiarize yourself with the data structures and calls, and so you can see the programs work. Further, we hope you use these programs as building blocks for your own experimentation with the window system. Each program contains comments explaining how the program works in general. For more information on data structures and routines used, please refer to the *SunWindows Reference Manual*.

This program illustrates how to use subwindow timers. These are briefly described in Section 6.3.1 of the *SunWindows Reference Manual.*

Each subwindow is described by a `toolsw` structure. One field in that structure is a pointer to a `toolio` structure; one of the fields in THAT struct is a pointer to a *timeval* structure. A *timeval* has the following definition (see `/usr/include/sys/time.h`):

```
struct timeval {
        long    tv_sec;
        long    tv_usec;
};
```

`tv_sec` and `tv_usec` specify a number of seconds and microseconds.

In a *toolio*, the timeval pointer field is called `tio_timer` . Ordinarily its value is NULL. When its value is non-null, then the *timeval* gives an amount of time for *tool_select* to wait for input events in the relevant subwindow. If the timer expires before an input event occurs, then control passes to the subwindow's selected routine, just as if there had been an input event. The way to tell whether it was an input event or a timeout that activated the selected routine is to check the fields in `timeval`. They are dynamically decremented, so if they are both zero you know that a timeout has happened.

This tool has a single subwindow. When you click the left mouse button, it activates a half-second timer in the subwindow. When the timer times out, the cursor image is changed and the timer is restarted. This goes on until you click the left mouse button again, at which point the tool goes into its original state, waiting for a click but not timing anything.

To compile:

```
cc timertool.c -o timertool -lsuntool -lsunwindow -lpixrect
```

```
#include <stdio.h>
#include <suntool/tool_hs.h>
#include <suntool/msgsw.h>

#define TIMING          1               /* Values for "state" */
#define NOT_TIMING      2

struct tool *tool;                      /* The tool */
struct toolsw *subwin;                  /* The subwindow */
struct msgsubwindow *msw;               /* Ditto */
struct pixfont *font;
struct inputmask im;
struct timeval timeval;
int sigwinchcatcher(), selected();
int state;
struct cursor *cursor_array[8];         /* Pointers to 8 possible cursors */
int cursor_index;                       /* Index of current cursor (0-7) */


/* The cursors.  See section 4.8.1. of the SunWindows Manual. */
DEFINE_CURSOR(cur0, 7, 7, PIX_SRC,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF);
DEFINE_CURSOR(cur1, 7, 7, PIX_SRC,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFE7F,
        0xFE7F,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF);
DEFINE_CURSOR(cur2, 7, 7, PIX_SRC,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFC3F,0xFC3F,
        0xFC3F,0xFC3F,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF);
DEFINE_CURSOR(cur3, 7, 7, PIX_SRC,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xF81F,0xF81F,0xF81F,
        0xF81F,0xF81F,0xF81F,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF);
DEFINE_CURSOR(cur4, 7, 7, PIX_SRC,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xF00F,0xF00F,0xF00F,0xF00F,
        0xF00F,0xF00F,0xF00F,0xF00F,0xFFFF,0xFFFF,0xFFFF,0xFFFF);
DEFINE_CURSOR(cur5, 7, 7, PIX_SRC,
        0xFFFF,0xFFFF,0xFFFF,0xE007,0xE007,0xE007,0xE007,0xE007,
        0xE007,0xE007,0xE007,0xE007,0xE007,0xFFFF,0xFFFF,0xFFFF);
DEFINE_CURSOR(cur6, 7, 7, PIX_SRC,
        0xFFFF,0xFFFF,0xC003,0xC003,0xC003,0xC003,0xC003,0xC003,
        0xC003,0xC003,0xC003,0xC003,0xC003,0xC003,0xFFFF,0xFFFF);
DEFINE_CURSOR(cur7, 7, 7, PIX_SRC,
        0xFFFF,0x8001,0x8001,0x8001,0x8001,0x8001,0x8001,0x8001,
        0x8001,0x8001,0x8001,0x8001,0x8001,0x8001,0x8001,0xFFFF);

main()
{
        state = NOT_TIMING;
        font = pw_pfsysopen();

        /* Create the tool. */
```

```
        tool = tool_make(WIN_LABEL, "Timer Example", 0);
        if (tool == NULL) {
                fputs("Can't make the tool.0, stderr);
                exit(1);
        }

        /* Create and init the subwindow. */
        subwin = msgsw_createtoolsubwindow(tool,"",
                 TOOL_SWEXTENDTOEDGE, TOOL_SWEXTENDTOEDGE,
                 "Click left button to start timing operation.", font);
        if (subwin == NULL) {
                fputs("Can't make the subwindow.0, stderr);
                exit(2);
        }
        msw = (struct msgsubwindow *)subwin->ts_data;
        subwin->ts_io.tio_selected = selected;
        /* Set up input mask to respond to left-button clicks. */
        input_imnull(&im);
        win_setinputcodebit(&im,MS_LEFT);
        win_setinputmask(subwin->ts_windowfd, &im, NULL, WIN_NULLLINK);

        /* Init the array of cursor-pointers. */
        cursor_index = 0;
        cursor_array[0] = &cur0;
        cursor_array[1] = &cur1;
        cursor_array[2] = &cur2;
        cursor_array[3] = &cur3;
        cursor_array[4] = &cur4;
        cursor_array[5] = &cur5;
        cursor_array[6] = &cur6;
        cursor_array[7] = &cur7;

        /* Install the tool */
        signal(SIGWINCH, sigwinchcatcher);
        tool_install(tool);

        /* Main loop. */
        tool_select(tool,0);

        /* Clean up */
        tool_destroy(tool);
        exit(0);
}
```

In this routine, there is a left-button click, or a timeout.

If you are in NOT_TIMING state, then it can only have been a click. In this case go to TIMING state and set up the timer for 1/2 second (500,000 microseconds).

If you are in TIMING state, you will have to figure out if you got a timeout or a click. As mentioned above, you do this by seeing if both fields in the `timeval` are zero. If so, restart the timer and change the cursor. If not, de-activate the

timer and switch to NOT_TIMING state.

```
selected() {
        struct inputevent ie;

        if (state == NOT_TIMING) {
                input_readevent(subwin->ts_windowfd, &ie);
                state = TIMING;                          /* Change state */
                timeval.tv_sec = 0;
                timeval.tv_usec = 500000;
                subwin->ts_io.tio_timer = &timeval;      /* Start timer */
                msgsw_setstring(msw,
                  "Click left button to halt timing operation.");
        }
        else {                                           /* TIMING state */
                if (timeval.tv_sec == 0 &&               /* If got timeout */
                    timeval.tv_usec == 0) {
                        timeval.tv_usec = 500000;        /* Restart timer */
                        cursor_index = (cursor_index+1) % 8;    /* Update */
                        win_setcursor(subwin->ts_windowfd,      /* the    */
                          cursor_array[cursor_index]);          /* cursor */
                }
                else {          /* Not timeout, must be click */
                        input_readevent(subwin->ts_windowfd, &ie);
                        subwin->ts_io.tio_timer = NULL; /* Deactivate timer */
                        state = NOT_TIMING;             /* Change state */
                        msgsw_setstring(msw,
                          "Click left button to restart timing operation.");
                }
        }
}


/* Standard SIGWINCH handler. */
sigwinchcatcher()
{
        tool_sigwinch(tool);
}
```

This program illustrates how to create an icon without the `icontool` facility, and how to change it dynamically.

The icon's image is a 64x64 pixrect. After we create it and draw in it (the image is a simple checkerboard pattern), we set up an `icon` structure. When we create the tool, we specify a pointer to that `icon` structure.

The tool consists of a single message subwindow. We set up a half-second timer in the subwindow. Please see *timertool.c* elsewhere in this package for details on using timers. When the timer goes off, we check to see if the tool is in iconic form. If so, we toggle the icon's image by XORing its pixrect with an all-1's pattern.

pattern.

To compile:

```
:cc changing_icon.c -o changing_icon -lsuntool -lsunwindow -lpixrect
```

```
#include <stdio.h>
#include <suntool/tool_hs.h>
#include <suntool/msgsw.h>


struct tool *tool;                          /* The tool itself */
struct toolsw *msw;                         /* The message subwindow */
struct pixfont *font;                       /* Font for writing in msg subwindow */
struct icon icon;                           /* The icon. */
struct pixrect *icon_mpr;                   /* Memory pixrect for the icon */
struct timeval timeval;                     /* The timer */
int sigwinchcatcher(), selected();

main(argc,argv)
int argc;
char **argv;
{
        /* Create and init the icon's memory pixrect. */
        icon_mpr = mem_create(64, 64, 1);
        if (icon_mpr == NULL) {
                printf("Aborting: cannot create icon's memory pixrect.0);
                exit(1);
        };
        pr_rop(icon_mpr, 0,0,32,32, PIX_SET, NULL,0,0);
        pr_rop(icon_mpr, 32,32,32,32, PIX_SET, NULL,0,0);


        /* Init the icon. */
        icon.ic_width = icon.ic_height = 64;
        icon.ic_background = NULL;
        icon.ic_gfxrect.r_left = 0;
        icon.ic_gfxrect.r_top = 0;
        icon.ic_gfxrect.r_width = 64;
        icon.ic_gfxrect.r_height = 64;
        icon.ic_mpr = icon_mpr;
        icon.ic_textrect.r_left = 0;
        icon.ic_textrect.r_top = 0;
        icon.ic_textrect.r_width = 0;
        icon.ic_textrect.r_height = 0;
        icon.ic_text = NULL;
        icon.ic_font = NULL;
        icon.ic_flags = ICON_BKGRDGRY;


        /* Create the tool struct. */
        tool = tool_create("Changing-Icon Tool", TOOL_NAMESTRIPE, NULL, &icon);
```

```
        if (tool == NULL) {
                printf("Aborting: cannot create tool.0);
                exit(1);
        }


        /* Create and init the subwindow. */
        font = pw_pfsysopen();
        msw = msgsw_createtoolsubwindow(tool, "", TOOL_SWEXTENDTOEDGE,
          TOOL_SWEXTENDTOEDGE, "Please make me iconic.", font);
        if (msw == NULL) {
                printf("Aborting: cannot create subwindow.0);
                exit(1);
        }
        msw->ts_io.tio_selected = selected;
        timeval.tv_sec = 0;                          /* Set up the timer */
        timeval.tv_usec = 500000;
        msw->ts_io.tio_timer = &timeval;


        /* install the tool */
        signal(SIGWINCH, sigwinchcatcher);
        tool_install(tool);



        /* Main loop. */
        tool_select(tool,0);



        /* clean up */
        tool_destroy(tool);
        exit(0);

}



/* Routine to handle SIGWINCH.    */
sigwinchcatcher()
{
        tool_sigwinch(tool);
}
```

This is the selected routine. Since there is no input mask to select any input events, the only way to get here is if the timer times out. We restart the timer and, if the tool is in iconic state, we invert the icon. In order to update the icon's

image as it appears on the screen, we have to call `tool_display`. (See p. 6-8 of the *SunTools Manual* for an explanation of the `tl_flags` field in the `tool` struct; we check this field when we see whether or not the tool is iconic. See p. 6-22 for an explanation of `tool_display`.)

```
selected() {
    timeval.tv_usec = 500000;       if ((tool->tl_flags & TOOL_ICONIC) == 0)          return(0);
    timeval.tv_usec = 500000;
    pr_rop(icon_mpr, 0,0,64,64,PIX_NOT(PIX_DST),NULL,0,0);
    tool_display(tool); }
```

This program demonstrates how to find out where a window has been damaged. "Damage" occurs either because the window size has changed or because windows above the window this program makes go away. For more information, see Chpt. 3 and Appendix A of the SunWindows Reference Manual (Revision G -- 2.0 release).

Each time this program recieves a SIGWINCH, it clears the entire window, loads in the clip list which tells what window "real estate" has been exposed, and draws boxes around each new piece of real estate.

This is the technique used by a program like the shelltool to repaint after window damage occurs. In a tty subwindow, a buffer of all the characters on the screen is maintained. When the rectlist indicates there is new exposed area in the window, the tty subwindow translates the corners of the damage in pixel coordinates to the corners in row and column coordinates. Then it simply re-writes the rows and columns that have been exposed. You can observe this by covering and uncovering a shelltool and watching the way it re-paints.

```
  To compile:
  machine% cc rectlists.c -o rectlists -lsuntool -lsunwindow -lpixrect -lm

#include <stdio.h>
#include <suntool/tool_hs.h>
#include <suntool/gfxsw.h>

main(argc, argv)
        int argc;
        char **argv;

{
        int height, width, vertical_line = 0;
        int x0, x1, y0, y1;
        struct rect rect;
        struct rectnode *rectnode;
        struct rectlist *rectlist;

        /* set up a graphics subwindow */
        struct gfxsubwindow *gfx = gfxsw_init(0, argv);
        struct pixwin *win = gfx->gfx_pixwin;

        for (;;) {
                if (gfx->gfx_flags & GFX_DAMAGED) {

                        /* check if window size changed */
                        height = win_getheight(gfx->gfx_windowfd);
                        width = win_getwidth(gfx->gfx_windowfd);

                        /* clear the window */
                        pw_writebackground(win, 0, 0, width, height, PIX_CLR);

                        /* set up new clip list & lock window during fix */
                        pw_damaged(win);
                        rectlist = &win->pw_clipdata->pwcd_clipping;
                        rl_rectoffset(rectlist, &rectlist->rl_bound, &rect);
                        pw_lock(win, &rect);

                        /* check if a window has been removed */
                        for (rectnode = rectlist->rl_head;
                          rectnode; rectnode = rectnode->rn_next) {
                                rl_rectoffset(rectlist, &rectnode->rn_rect,
                                  &rect);
                                /* draw a square around this rect */
                                x0 = rect.r_left;
                                x1 = rect_right(&rect);
                                y0 = rect.r_top;
                                y1 = rect_bottom(&rect);
                                pw_vector(win, x0, y0, x0, y1, PIX_SET, 1);
```

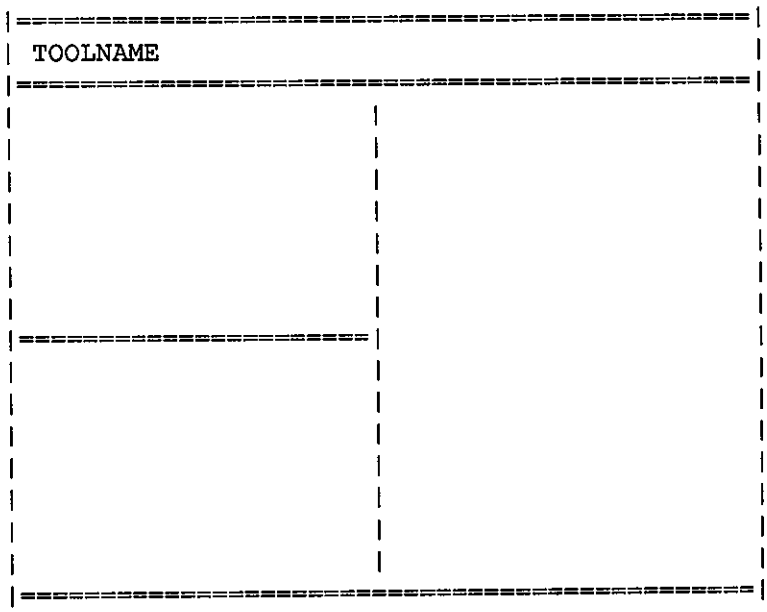
```
                        pw_vector(win, x0, y1, x1, y1, PIX_SET, 1);
                        pw_vector(win, x1, y1, x1, y0, PIX_SET, 1);
                        pw_vector(win, x1, y0, x0, y0, PIX_SET, 1);
                    }

                    /* done, so unlock, clear flag, and undo rectlist */
                    pw_unlock(win);
                    gfx->gfx_flags &= ~GFX_DAMAGED;
                    pw_donedamaged(win);

                    /* give the user some idea what to do */
                    vertical_line += 20;
                    if (vertical_line > height) vertical_line = 20;
                    pw_text(win, 20, vertical_line, PIX_SRC, NULL,
                      "Put windows on this window, and then take them off!")
                }
        }
}


#ifndef lint
static  char sccid[] = "@(#)layout.c 2.0 85/05/20 Copyr 1985 Sun Micro";
#endif
```

This program illustrates how to defeat the default tiling algorithm, which as of 2.0 leaves much to be desired. A major problem is that you can't lay out subwindows in a tool like this:



Section 6.2.5 of the SunWindows Reference Manual says that what you need to do is write your own routine named tool_layoutsubwindows(tool). At link time, this routine will supersede the routine provided in the suntool.a library. The user-supplied version should set up rects to describe the desired layout (one rect per subwindow), and enforce the rects with calls to win_setrect, which is documented in section 4.3.

The only tricky part is figuring out what the coordinates and sizes (which you put in the rects) refer to: should you or should you not figure in the borders? Here's the low-down. The Width and height of the tool (specified in tool_make) include the top namestripe (16 pixels) and the top, left, bottom, and right borders (5 pixels each: 2 black, then 1 white, then 2 more black). Moreover, in order for things to look right, there should be 5 pixels in between all subwindows. All coordinates are relative to the top-left corner of the tool (i.e. the first pixel in the namestripe); dimensions do NOT include borders. Thus the first subwindow goes at (5,18).

Note that instead of using explicit values for the size of the namestripe, border width, and spacing, we could have used the functions `tool_stripeheight()`, `tool_borderwidth()FP`, and `tool_subwindow_spacing()`. These are described in section 6.2.5 of the SunWindows Reference Manual.

To compile:

```
cc layout.c -o layout -lsuntool -lsunwindow -lpixrect
```

```
#include <stdio.h>
#include <suntool/tool_hs.h>
#include <suntool/msgsw.h>

struct tool *tool;                          /* The tool. */
struct toolsw *sw1, *sw2, *sw3;             /* The subwindows. */
struct pixfont *font;                       /* Font for writing in subwins. */



 Routine to handle SIGWINCH.  Nothing special. */
 sigwinchcatcher()
{
        tool_sigwinch(tool);
}



main(argc,argv)
int argc;
char **argv;
{
        /* Create the tool. */
        tool = tool_make(WIN_LABEL, argv[0],
                         WIN_WIDTH, 500,
                         WIN_HEIGHT, 428,
                         0);
        if (tool == NULL) {
                printf("Aborting: cannot create tool.0);
                exit(1);
        };



        font = pw_pfsysopen();

        /* Create 3 subwindows.  Dimensions don't matter, since we're */
        /* going to override them and the tiling algorithm.           */
        sw1 = msgsw_createtoolsubwindow(tool,"",
                TOOL_SWEXTENDTOEDGE, 100,
                "This is the first subwindow.", font);

        sw2 = msgsw_createtoolsubwindow(tool,"",
                TOOL_SWEXTENDTOEDGE, 100,
                "This is the second subwindow.", font);

        sw3 = msgsw_createtoolsubwindow(tool,"",
                TOOL_SWEXTENDTOEDGE,TOOL_SWEXTENDTOEDGE,
                "This is the third subwindow.", font);

        if (sw1==NULL || sw2==NULL || sw3==NULL) {
```

```
                printf("Aborting: cannot create subwindows.0);
                exit(2);
        };


        /* Install the tool */
        signal(SIGWINCH, sigwinchcatcher);
        tool_install(tool);

        /* Main loop. */
        tool_select(tool,0);

        /* Clean up */
        tool_destroy(tool);
        exit(0);
}
```

This is our customized version of the layout algorithm. All we do is format rects to tell the subwindows where we really want them to go.

```
tool_layoutsubwindows(t)
struct tool *t;
{
        struct rect rect1, rect2, rect3;

        rect1.r_left = 5;
        rect1.r_top = 18;
        rect1.r_width = 100;
        rect1.r_height = 200;
        win_setrect(sw1->ts_windowfd, &rect1);
        rect2.r_left = 5;
        rect2.r_top = 223;
        rect2.r_width = 100;
        rect2.r_height = 200;
        win_setrect(sw2->ts_windowfd, &rect2);
        rect3.r_left = 110;
        rect3.r_top = 18;
        rect3.r_width = 385;
        rect3.r_height = 405;
        win_setrect(sw3->ts_windowfd, &rect3);
}

#ifndef lint
static  char sccid[] = "@(#)onewaytomakeacolorbar.c 2.0 85/05/20 Copyr 1985 Sun Micr
#endif
```

colormap experimentation -- how to manipulate colormap and bit-planes

This program draws 2**n bars of different colors (0 <= n <= 8). It loads a gray-scale colormap, enables all the bit-planes it needs for the number of bars it is about to draw, and then draws the bars.

Remember, in the window system, all colors will show up in any given window only when the mouse is in that window.

To compile:

cc onewaytomakeacolorbar.c -o onewaytomakeacolorbar -lsuntool -lsunwindow -lpixrec

```
#include <stdio.h>
#include <suntool/tool_hs.h>
#include <suntool/gfxsw.h>

main(argc, argv)
        int argc;
        char **argv;

{
        struct gfxsubwindow *gfx = gfxsw_init(0, argv);
        struct pixwin *win = gfx->gfx_pixwin;
        unsigned char red[256], green[256], blue[256];
        int colors, i, height, width, planes;
        char maps[20];

        for (;;) {
        for (colors=2; colors<257; colors *= 2) {

                printf("Number of colors: %d0, colors);
                planes = colors-1;

                /* generate a colormap with 'colors' entries where the
                   0th entry is black and the color'th entry is white
                   with a gray scale in the intervening values*/
                for (i=0; i<colors; i++) {
                        red[i] = green[i] = blue[i] = i*255/(colors-1);
                }

                /* find the dimensions of the rect */
                height = win_getheight(gfx->gfx_windowfd);
                width = win_getwidth(gfx->gfx_windowfd);

                /* make up a colormap name and load it */
                sprintf(maps, "testcolor%d", colors);
                pw_setcmsname(win, maps);

                /* put in the colormap we made above */
                pw_putcolormap(win, 0, colors, red, green, blue);

                /* enable all the bit-planes used */
                pw_putattributes(win, &planes);

                /* clear the window */
                pw_write(win, 0, 0, width, height, PIX_CLR, NULL, 0, 0);

                /* determine the width of the colorbars */
                width /= colors;
                if (width < 1) width = 1;
```

```
                /* draw the colorbars -- one for each color we're displaying */
                for (i=0; i<colors; i++) {
                        pw_write(win, i*width, 0, width, height,
                          PIX_SRC|PIX_COLOR(i), NULL, 0, 0);
                }

                sleep(3);

                /* make and insert the inverse colormap of the first one
                   we made */
                for (i=0; i<colors; i++) {
                        red[i] = green[i] = blue[i] = 255 - (i*255/(colors-1));
                }
                pw_putcolormap(win, 0, colors, red, green, blue);

                sleep(3);

        }
        }
}

#ifndef lint
static  char sccid[] = "@(#)anotherwaytomakeacolorbar.c 2.0 85/05/20 Copyr 1985 Sun
#endif
```

colormap experimentation -- another way to manipulate colormap and bit-planes

This program draws 2**n bars of different colors (0 <= n <= 8). It loads a gray-scale colormap and then draws the colorbars. In this version, the programs tries to write to all the bit-planes each time it writes a bar. However, before it writes the bar, it only enables the proper set of bit-planes for the color it wants to draw.

Remember, in the window system, all colors will show up in any given window only when the mouse is in that window.

To compile:

cc anotherwaytomakeacolorbar.c -o anotherwaytomakeacolorbar -lsuntool -lsunwindc

```
#include <stdio.h>
#include <suntool/tool_hs.h>
#include <suntool/gfxsw.h>

main(argc, argv)
        int argc;
        char **argv;

{
        struct gfxsubwindow *gfx = gfxsw_init(0, argv);
        struct pixwin *win = gfx->gfx_pixwin;
        unsigned char red[256], green[256], blue[256];
        int colors, i, height, width, planes;
        char maps[20];

        for (;;) {
        for (colors=2; colors<257; colors *= 2) {

                printf("Number of colors: %d0, colors);
                planes = colors-1;

                /* generate a colormap with 'colors' entries where the
                   0th entry is black and the color'th entry is white
                   with a gray scale in the intervening values*/
                for (i=0; i<colors; i++) {
                        red[i] = i*255/(colors-1);
                        green[i] = 255 - (i*255/(colors-1));
                        blue[i] = 250;
                }

                /* find the dimensions of the rect */
                height = win_getheight(gfx->gfx_windowfd);
                width = win_getwidth(gfx->gfx_windowfd);

                /* make up a colormap name and load it */
                sprintf(maps, "testcolor%d", colors);
                pw_setcmsname(win, maps);

                /* put in the colormap we made above */
                pw_putcolormap(win, 0, colors, red, green, blue);

                /* enable all the bit-planes used */
                pw_putattributes(win, &planes);

                /* clear the window */
                pw_write(win, 0, 0, width, height, PIX_CLR, NULL, 0, 0);

                /* determine the width of the colorbars */
```

```
                width /= colors;
                if (width < 1) width = 1;

                /* draw the colorbars -- one for each color we're displaying
                   Note: this time we write out to all the bit-planes with
                   the pw_write() call, but we only enable some of the
                   bit-planes.  This is a different way of making colors */
                for (i=0; i<colors; i++) {
                        planes = i;
                        pw_putattributes(win, &planes);
                        pw_write(win, i*width, 0, width, height,
                          PIX_SET, NULL, 0, 0);
                }

                sleep(3);

                /* make and insert the inverse colormap of the first one
                   we made */
                for (i=0; i<colors; i++) {
                        red[i] = 255 - (i*255/(colors-1));
                        green[i] = i*255/(colors-1);
                        blue[i] = 250;
                }
                pw_putcolormap(win, 0, colors, red, green, blue);

                sleep(3);

        }
        }
}
```

#ifndef lint static char sccid[] = "@(#)animation.c 2.0 85/05/20 Copyr 1985 Sun Micro"; #endif

Move a square around the screen randomly -- a random walk

This program uses "animation", a technique of displaying one bit-plane while drawing to another. This gives the appearance of smooth change from one image to the next. You can sort of see what animation looks like without multiple bit-planes if you run this program on a monochrome display. A good exercise, however, is to modify this program slightly to use only one bit-plane so you can see the difference between smooth multi-plane and jumpy single-plane animation.

To compile:

```
machine% cc animation.c -o animation -lsuntool -lsunwindow -lpixrect
```

```
#include <stdio.h>
#include <suntool/tool_hs.h>
#include <suntool/gfxsw.h>

#define RECTX 20
#define RECTY 35

        /* declare all the functions we use to generate random walk */
        long random(), getpid();
        int srandom();

main(argc, argv)
        int argc;
        char **argv;

{

        int posx, posy;
        int old_posx, old_posy;
        int planes;
        int height, width;

#define BCKRND 230
#define BCKGRND BCKRND-10

        /* set up two color maps, red0, green0, blue0, and red1, green1,
           blue1 */
        static unsigned char red0[4] = {0, BCKRND, 0, BCKRND};
        static unsigned char green0[4] = {0, BCKRND, 0, BCKRND};
        static unsigned char blue0[4] = {BCKGRND, BCKGRND, BCKGRND, BCKGRND};

        static unsigned char red1[4] = {0, 0, BCKRND, BCKRND};
        static unsigned char green1[4] = {0, 0, BCKRND, BCKRND};
        static unsigned char blue1[4] = {BCKGRND, BCKGRND, BCKGRND, BCKGRND};

        /* set up a graphics subwindow */
        struct gfxsubwindow *gfx = gfxsw_init(0, argv);
        struct pixwin *win = gfx->gfx_pixwin;

        /* initialize the randomizing variable */
        srandom((int)(getpid()));

        /* set the colormap name */
        pw_setcmsname(win, "animate");

Restart:
```

```
/* window size may have changed, so get current coordinates */
width = win_getwidth(gfx->gfx_windowfd);
height = win_getheight(gfx->gfx_windowfd);
old_posx = posx = width/2;
old_posy = posy = height/2;

/* put in the red1, green1, blue1 colormap */
pw_putcolormap(win, 0, 4, red1, green1, blue1);

/* clear the window */
planes = 3;
pw_putattributes(win, &planes);
pw_write(win, 0, 0, width, height, PIX_NOT(PIX_SRC), NULL, 0, 0);

/* do some initialization */
planes = 1;
pw_putattributes(win, &planes);
pw_write(win, posx, posy, RECTX, RECTY, PIX_SRC, NULL, 0, 0);
planes = 2;
pw_putattributes(win, &planes);

/* draw the thing zillions of times */
for (;;) {

        /* check for damage */
        if (gfx->gfx_flags&GFX_DAMAGED) gfxsw_handlesigwinch(gfx);
        if (gfx->gfx_flags&GFX_RESTART) {
                gfx->gfx_flags &= ~GFX_RESTART;
                goto Restart;
        }

        /* draw the next square */
        pw_write(win, posx, posy, RECTX, RECTY, PIX_SRC, NULL, 0, 0);

        /* swap colormaps so the polygon just drawn shows and
           the polygon about to be drawn won't show */
        if (planes == 2) {
                pw_putcolormap(win, 0, 4, red1, green1, blue1);
                planes = 1;
        } else {
                pw_putcolormap(win, 0, 4, red0, green0, blue0);
                planes = 2;
        }

        /* wipe out the old square */
        pw_putattributes(win, &planes);
        pw_write(win, old_posx, old_posy, RECTX, RECTY,
          PIX_NOT(PIX_SRC), NULL, 0, 0);
        old_posx = posx;
        old_posy = posy;

        /* determine the next posx and posy */
        if (random() < 1073741824) {
```

```
                posx -= random()/268435456 + 1;
        } else {
                posx += random()/268435456 + 1;
        }
        if (random() < 1073741824) {
                posy -= random()/268435456 + 1;
        } else {
                posy += random()/268435456 + 1;
        }
        if (posx < 0) posx = width-1;
        if (posx > width) posx = 0;
        if (posy < 0) posy = height-1;
        if (posy > height) posy = 0;
    }
}


#ifndef lint
static  char sccid[] = "@(#)parse_all.c 2.0 85/05/20 Copyr 1985 Sun Micro";
#endif
```

This program illustrates correct use of the "tool_parse_all" call, which is used to extract tool-related command line options from argv.

After we call "tool_parse_all", all parameters left in argv are suitable for parsing. All we do is concatenate them in a string which we display in a message subwindow. Note the use of "tool_free_attribute_list", which releases resources grabbed by "tool_parse_all".

To see the tool in action, type "parse_all", followed by several command line options. Some of these should be tool-related (see Table 6-2 on p. 6-10 of the SunWindows Reference Manual for a list); these will be filtered out by "tool_parse_all" and will affect the tool. The other options will be seen in the message subwindow (a serious tool would parse them).

```
                                        Sample command line:
                                                parse_all -xxx -yyy -width 100 -zzz -heig
```

To compile:

```
cc parse_all.c -o parse_all -lsuntool -lsunwindow -lpixrect
```



```
#include <stdio.h> #include <suntool/tool_hs.h> #include <suntool/msgsw.h>

struct tool *tool;              /* The tool */
struct toolsw *subwin;              /* The subwindow */
struct pixfont *font;
int sigwinchcatcher();

main(argc, argv)
        int argc;
        char *argv[];
{
        char **tool_attributes = NULL;
        char *tool_name = argv[0];
        char msg_string[80];
        argv++;
        argc--;
        if(tool_parse_all(&argc, argv, &tool_attributes, tool_name) == -1) {
                tool_usage(tool_name);
                exit(1);
        }
        msg_string[0] = 0;
        while (argc > 0  &&  **argv == '-') {
                strcat(msg_string, *argv);
                strcat(msg_string, " ");
                argv++;
                argc--;
        }

        /* Create the tool. */
        tool = tool_make(WIN_LABEL, argv[0],
                        WIN_ATTR_LIST, tool_attributes,
                        0);
         if (tool == NULL) {
                fputs("Can't make the tool.0, stderr);
             exit(1);
        }
         tool_free_attribute_list(tool_attributes);
        font = pw_pfsysopen();
```

```
    subwin = msgsw_createtoolsubwindow(tool,"",
        TOOL_SWEXTENDTOEDGE, TOOL_SWEXTENDTOEDGE,
            msg_string, font);
    if (subwin == NULL) {
        fputs("Can't make the subwindow.0, stderr);
        exit(1);
    }

    /* Install the tool */
    signal(SIGWINCH, sigwinchcatcher);
    tool_install(tool);

    /* Main loop. */
    tool_select(tool,0);

    /* Clean up */
    tool_destroy(tool);
    exit(0); }


/* Standard SIGWINCH handler. */ sigwinchcatcher() {
    tool_sigwinch(tool); }


#ifndef lint static  char sccid[] = "@(#)findroot.c 2.0 85/05/20 Copyr 1985 Sun Micro"; #endif
```

This program illustrates how to find out what a tool's root window's file descriptor is. This is necessary, for example, if you want to use certain of the Window Manager calls which are documented in section 8.5.1 of the 1.1 SunWindows Reference Manual (section 9.6.1 of the 2.0 Manual).

This program creates a tool with a single message subwindow. When the user clicks the right-hand mouse button, the wmgr_close call is used to shut the tool down to iconic form.

The interesting part is the function get_my_root_fd(). See comments there for details.

```
#include <stdio.h>
#include <sys/file.h>
#include <suntool/wmgr.h>
#include <suntool/tool_hs.h>
#include <suntool/msgsw.h>

struct tool *tool;                      /* The tool. */
struct toolsw *subwin;                  /* The single subwindow. */
struct pixfont *font;                   /* Font for writing in the subwin. */
struct inputmask inputmask;             /* Input mask for the subwindow. */
int subwin_selected();                  /* "Selected" routine for subwin. */


/* Routine to handle SIGWINCH.  Nothing special. */
sigwinchcatcher()
{
        tool_sigwinch(tool);
}


main()
{
        font = pw_pfsysopen();

        /* Create the tool. */
        tool = tool_create("FINDROOT",
           TOOL_NAMESTRIPE|TOOL_BOUNDARYMGR,
           NULL, NULL);
        if (tool == NULL) {
             printf("Couldn't create the tool.0);
             exit(1);
        };

        /* Create and init the subwindow. */
        subwin = msgsw_createtoolsubwindow(tool, "",
           TOOL_SWEXTENDTOEDGE, TOOL_SWEXTENDTOEDGE,
           "Click Right Button to go iconic.", font);
        if (subwin == NULL) {
             printf("Couldn't create the subwindow.0);
             exit(1);
        };
        /* Set up inputmask to accept only RB clicks. */
        input_imnull(&inputmask);
        win_setinputcodebit(&inputmask,MS_RIGHT);
        win_setinputmask(subwin->ts_windowfd, &inputmask,
            NULL,WIN_NULLLINK);
        subwin->ts_io.tio_selected = subwin_selected;

        /* Install the tool. */
        signal(SIGWINCH, sigwinchcatcher);
```

```
        tool_install(tool);

        /* Main loop. */
        tool_select(tool,0);

        /* Cleanup. */
        tool_destroy(tool);
        exit(0);
}
```

If we get here, the user must have clicked the right-hand mouse button. This routine calls get_my_root_fd() to figure out the file descriptor of its root window, then calls wmgr_close() to close the tool to icon form.

```
/
subwin_selected()
{
        struct inputevent ie;
        char c[WIN_NAMESIZE];
        int rootfd;

        input_readevent(subwin->ts_windowfd, &ie);
        rootfd = get_my_root_fd(tool->tl_windowfd);
        if (rootfd == -1) {
                printf("get_my_root_fd() failed.0);
                exit(3);
        }
        wmgr_close(tool->tl_windowfd, rootfd);
}
```

This is the interesting part of the program. This function takes as its argument the file descriptor for a window, and returns the file descriptor of the argument's root window (or -1 if something goes wrong). See section 4.4 of the SunWindows Reference Manual for details on the window hierarchy.

The strategy is to recursively call win_getlink, working our way up the window tree, until we get to the top (i.e. win_getlink returns WIN_NULLLINK). The only problem is that win_getlink returns a window number, not a file descriptor. To convert the window number to a file descriptor, we first call win_numbertoname, which converts the window number to a string such as "/dev/win3". Then we open the device in order to get a file descriptor.

```
get_my_root_fd(fd)
int fd;
{
        int original_fd, parentfd;
        char c[WIN_NAMESIZE];

        original_fd = fd;
        while ( (parentfd = win_getlink(fd,WL_PARENT)) != WIN_NULLLINK ) {
                if (fd != original_fd)
                        close(fd);
                win_numbertoname(parentfd,c);
                fd = open(c,O_RDONLY,0);
                if (fd == -1)
                        return(-1);
        }
        return(fd);
}
```

B

Appendix B:Contents of *get_arch_f* File

Appendix B:Contents of *get_arch_f* File

This is a list of the differences between files in Release 2.2 and 2.0.

```
/bin/as
/bin/test
/dev/MAKEDEV
/etc/gp1cg2.1024.ucode
/etc/gp1cg2.1152.ucode
/etc/gpconfig
/etc/inetd
/etc/mount
/etc/nfsd
/etc/showmount
/etc/shutdown
/etc/umount
/etc/yp/makedbm
/etc/ypbind
/etc/ypserv
/lib/ccom
/lib/f1
/lib/libc.a
/usr/bin/adjacentscreens
/usr/bin/clear_colormap
/usr/bin/clocktool
/usr/bin/coretool
/usr/bin/dbxtool
/usr/bin/fonttool
/usr/bin/gfxtool
/usr/bin/icontool
/usr/bin/lockscreen
/usr/bin/perfmeter
/usr/bin/perfmon
/usr/bin/screendump
/usr/bin/screenload
/usr/bin/shelltool
/usr/bin/suntools
/usr/bin/tektool
/usr/bin/toolplaces
/usr/diag/sysdiag
/usr/etc/in.tftpd
/usr/etc/nfsstat
```

```
/usr/etc/ping
/usr/etc/rpc.mountd
/usr/etc/rpc.yppasswdd
/usr/etc/trpt
/usr/include/cgicbind.h
/usr/include/cgiconstants.h
/usr/include/cgidefs.h
/usr/include/cgipw.h
/usr/include/gpl_pwpr.h
/usr/include/netinet/ip.h
/usr/include/nfs/nfs.h
/usr/include/nfs/nfs_clnt.h
/usr/include/pascal/devincpas.h
/usr/include/pixrect
/usr/include/sun/fbio.h
/usr/include/sun/gpio.h
/usr/include/suntool
/usr/include/usercore.h
/usr/lib/font/ftS
/usr/lib/font/fttS
/usr/lib/f77pass1
/usr/lib/libF77.a
/usr/lib/libF77_p.a
/usr/lib/libI77.a
/usr/lib/libI77_p.a
/usr/lib/libc_p.a
/usr/lib/libcgi.a
/usr/lib/libcgi77.a
/usr/lib/libcore.a
/usr/lib/libcoresky.a
/usr/lib/libdbm.a
/usr/lib/libpfc.a
/usr/lib/libpfc_p.a
/usr/lib/libpixrect.a
/usr/lib/libsuntool.a
/usr/lib/libsunwindow.a
/usr/lib/lint/lint1
/usr/lib/lint/llib-lc.ln
/usr/lib/lint/llib-lcore.ln
/usr/lib/lint/llib-lcurses.ln
/usr/lib/lint/llib-lm.ln
/usr/lib/lint/llib-lmp.ln
/usr/lib/lint/llib-lpixrect.ln
/usr/lib/lint/llib-lsuntool.ln
/usr/lib/lint/llib-lsunwindow.ln
/usr/lib/lpd
/usr/lib/sendmail
/usr/lib/sendmail.main.cf
/usr/lib/sendmail.subsidiary.cf
/usr/lib/vwidth
/usr/src/sun/suntool/get_view_surface.c
/usr/sys/OBJ/bw1_rop.o
/usr/sys/OBJ/cg2_colormap.o
```

```
/usr/sys/OBJ/cg2_rop.o
/usr/sys/OBJ/cgtwo.h
/usr/sys/OBJ/cgtwo.o
/usr/sys/OBJ/conf.o
/usr/sys/OBJ/consfb.o
/usr/sys/OBJ/gp1_colormap.o
/usr/sys/OBJ/gp1_kern_sync.o
/usr/sys/OBJ/gp1_rop.o
/usr/sys/OBJ/gpone.h
/usr/sys/OBJ/gpone.o
/usr/sys/OBJ/if.o
/usr/sys/OBJ/if_ec.o
/usr/sys/OBJ/if_ether.o
/usr/sys/OBJ/if_ie.o
/usr/sys/OBJ/if_loop.o
/usr/sys/OBJ/ioconf.o
/usr/sys/OBJ/ip_icmp.o
/usr/sys/OBJ/ip_input.o
/usr/sys/OBJ/ip_output.o
/usr/sys/OBJ/kbd.o
/usr/sys/OBJ/keytables.o
/usr/sys/OBJ/kudp_fastsend.o
/usr/sys/OBJ/machdep.o
/usr/sys/OBJ/mem_rop.o
/usr/sys/OBJ/mti.o
/usr/sys/OBJ/nfs_server.o
/usr/sys/OBJ/nfs_subr.o
/usr/sys/OBJ/nfs_vfsops.o
/usr/sys/OBJ/nfs_vnodeops.o
/usr/sys/OBJ/nfs_xdr.o
/usr/sys/OBJ/raw_ip.o
/usr/sys/OBJ/sc.o
/usr/sys/OBJ/sd.o
/usr/sys/OBJ/st.o
/usr/sys/OBJ/sys_generic.o
/usr/sys/OBJ/tcp_debug.o
/usr/sys/OBJ/tcp_input.o
/usr/sys/OBJ/tcp_output.o
/usr/sys/OBJ/tcp_subr.o
/usr/sys/OBJ/tcp_timer.o
/usr/sys/OBJ/tcp_usrreq.o
/usr/sys/OBJ/tty.o
/usr/sys/OBJ/udp_usrreq.o
/usr/sys/OBJ/ufs_alloc.o
/usr/sys/OBJ/ufs_nd.o
/usr/sys/OBJ/vers.o
/usr/sys/OBJ/vm_drum.o
/usr/sys/OBJ/vm_machdep.o
/usr/sys/OBJ/vm_pt.o
/usr/sys/OBJ/vm_text.o
/usr/sys/OBJ/xy.o
/usr/sys/conf/GENERIC
/usr/sys/conf/ND120
```

```
/usr/sys/conf/README
/usr/sys/conf/RELEASE
/usr/sys/conf/SDST160GP
/usr/sys/conf/devices.sun
/usr/sys/conf/files
/usr/sys/conf/files.sun
/usr/sys/conf/makefile.sun
/usr/sys/sun/conf.c
/usr/sys/sun/fbio.h
/usr/sys/sun/gpio.h
/usr/sys/sundev/scmb.h
/usr/sys/sundev/streg.h
/usr/ucb/ex
/usr/ucb/ftp
/usr/ucb/man
```

Revision History

Revision	Date	Comments
03	11 September 1985	Preliminary release of this Release 2.2 Manual.
04	20 September 1985	Preliminary release of this Release 2.2 Manual.
05	30 September 1985	Beta Draft of this Release 2.2 Manual