sun®
microsystems

SunOS™ 4.0 Change Notes

Part Number: 800-1731-10
Revision A of 9 May 1988

Contents

Tables

1

Moving From 3.x to 4.0

1.1. Advantages of 4.0

SunOS 4.0 provides new functionality and features not available in SunOS 3.x releases. For a detailed discussion of changes, refer to the *4.0 Change Notes* chapter of this document.

Key improvements incorporated by SunOS 4.0 include:

- New system architecture that promotes system resource sharing and portability across different hardware platforms.
- Shared library facility that reduces program size and swap space requirements.
- Resizable swap area for diskless clients.
- Secure networking through the use of RPC (Remote Procedure Call).
- NFS (Network File System) replaces ND (Network Disk) for diskless client systems. The effect of this is to make system administration easier and more flexible.
- All the 4.3BSD network changes are incorporated including TCP (Transmission Control Protocol) and IP (Internet Protocol) performance improvements and subnetting.
- Automount facility that automatically mounts accessible remote filesystems as needed.

1.2. *Install* vs *Upgrade*

Moving from SunOS 3.x to 4.0 cannot be accomplished through a standard upgrade procedure, but requires a re-installation of the entire system.

In a standard upgrade, you selectively replace files in each architecture. Due to the filesystem reorganization that separates host-dependent, nonshared files from architecture-dependent, shared files, selective replacement of files is not possible. The only way to incorporate the SunOS 4.0 enhancements is to re-install the entire system.

The new SunOS 4.0 filesystem organization is transparent to most users and simplifies the administration of diskless clients. The new filesystem layout also makes it easier for many clients of different architectures to work with a single server.

1.3. Issues to Consider Before Moving to 4.0

In general, the migration from 3.x to 4.0 is recommended. As with any release, however, there are isolated differences that require special consideration before migration to the new release. Review the following items and consider their impact to your site before migrating to *SunOS 4.0*.

- If you install the *entire* SunOS 4.0 system, it requires more disk space. Unnecessary modules should be removed from your system to optimize disk usage. For more information regarding required disk space, see Appendix B of *Installing the SunOS*.
- The SunOS 4.0 kernel must be reconfigured to realize SunOS 3.x performance levels. A reconfigured kernel optimizes memory usage and is preferable to the GENERIC kernel.
- Programs compiled with pre-SunOS 4.0 releases should be recompiled to take advantage of new SunOS 4.0 enhancements such as shared libraries.
- Source code changes may be required for modules that use changed or obsoleted library and system calls.

Installing SunOS 4.0

For information regarding the installation of SunOS 4.0, see *Installing the SunOS*.

2

SunOS 4.0 Change Notes

2.1. Introduction

This document highlights the changes and enhancements contained in SunOS 4.0 that were not available in the SunOS 3.x releases. All users migrating from a SunOS 3.x release should read this document. For new users this is not as critical, however it may be helpful in noting the most current advances in SunOS.

The change and enhancement descriptions in this document are summaries. For more in-depth information see the suggested documentation.

Compatibility

SunOS 4.0 software runs on Sun-2, Sun-3, and Sun-4 systems. As with any major release, there are compatibility issues with previous releases. The compatibility issues for SunOS 4.0 are discussed below.

NOTE *Pre-SunOS 4.0 object files should not be used with SunOS 4.0. This means that all* `.o` *and* `.a` *files built under previous systems must be recompiled prior to their use under SunOS 4.0.*

- All binary or source code developed on a Sun-3 under SunOS 3.x runs on a Sun-3 under SunOS 4.0 with the following exceptions:
  - Programs that depend on system file locations that have changed, and a symbolic link was not left behind.
  - Programs that read kernel data structures via `/dev/kmem` or `/dev/mem`.
  - Programs that use the current `mmap` system call require minor source changes due to the new VM system, however, existing binaries work.
  - Programs that depend on the "unmap on close" semantics of the `mmap` system call in the old system will not run under SunOS 4.0.
  - Almost any customer written kernel code will require changes.
  - Any code that uses the `LTILDE` flag or the `t_dsuspc` interrupt character must be changed due to the new System V/BSD-compatible `tty` driver.
  - Upgrading to the 4.3BSD networking system has introduced the following incompatibilities: `ioctls` for looking at the network configuration, socket process group handling, the interface to the `setsockopt` call.

  - The NIT facility has changed and the interface is different. It now requires a source change for customers to write to this level.
  - Programs with embedded executable names may fail due to the file system reorganization.

- Programs that use `getpwent()` and associated routines should be rebuilt in SunOS 4.0 or they may not work correctly in a C2 secure environment.
- Programs written with the previous signal-handler function type continue to function properly but, there will be new compiler and `lint` messages. The messages warn that the code does not conform to standard, but are harmless.
- `init` no longer uses `/etc/ttys` so any administrative procedures that modify it must be changed to modify `/etc/ttytab` instead.
- The `syslog` facility has been upgraded to be compatible with 4.3BSD. For a discussion of the incompatibilities, see the `syslog` compatibility under the *4.3BSD Enhancements* section of the *System Software Changes* in this document.
- Due to driver, loader, and PROM attributes associated with the QIC24 high density tape, SunOS 4.0 only loads when the system has a PROM revision of 1.8 or greater. See *QIC 24 PROM* under the *System Software Changes* section of this document for more information.
- Source calls to `getdirentries` must be changed to calls to the new `getdents` system call. It is not sufficient to simply replace a `getdirentries()` call with a `getdents` call. It must be converted.

Availability of Sun Unbundled Software

Most of Sun's unbundled products are being revised to take advantage of the functionality introduced with SunOS 4.0. With the exception of the core compilers and communications products, many of the existing versions work with SunOS 4.0. The following chart titled "Sun Unbundled Software Availability" provides information on availability of Sun's unbundled software for releases 3.X and 4.0. Note that versions designed to take full advantage of SunOS 4.0 generally support the Sun-2, 3, and 4 platforms. Contact you local sales representative for specific product availbility information.

Table 2-1 *Sun Unbundled Software Availability*

Unbundled Software Availability				
OS and Platfrom	SunOS 3.X Sun-2,3	Sun-4	SunOS 4.0 Binary Compatibility	SunOS 4.0 Full Functionality
Product				
NeWS	1.1	1.1	1.1	1.1
SunGKS	2.1	2.1	2.1[1]	†
Sun FORTRAN	1.0	1.05	—	1.1
Sun Cross-Compilers	2.0	2.0	—	2.1
Sun Pascal	—	1.05	—	1.1
NSE	1.0	1.05	—	1.1
SunTrac	1.0/1.1	1.0/1.1	1.0/1.1	1.2
PC-NFS	3.0	3.0	3.0[2]	3.0
SunIPC	1.1	—	—	†
SunINGRES	5.0/5.1	—	5.0/5.1[1]	6.0
SunUNIFY/SunSimplify	3.0	3.0	3.0[1]	3.0
SunAlis	2.1	—	2.1	†
Sun Common Lisp	2.1	2.1	—	†
SPE 1.0 for SCLisp	2.1	2.1	—	†
Sun Modula-2	2.0	—	2.0[3]	2.1
Transcript	2.0	—	2.0[4]	2.1
X.25	5.0	5.1	—	6.0
DNI	5.0	5.1	—	6.0
SNA 3270	5.0	5.1	—	6.0
TE100	4.0	5.1	—	6.0
IR	5.0	—	—	6.0
BSC 3270	3.0	—	—	6.0
Channel	5.0[5]	—	—	6.0
Local 3270	5.0[6]	—	—	6.0
DDN	5.0	—	—	6.0
MCP	5.0[5]	—	—	6.0
SCP 3.0	3.0	—	—	6.0[4]
OSI 5.0	5.0	—	—	6.0
SNA Peer 5.0	5.0	—	—	6.0
MHS 5.1	5.1	—	—	6.0

† Announcement pending.

[1] Restricted to running existing applications.

[2] Availability of PC-NFS support for serial lines under SunOS 4.0 to be announced later.

[3] Minor changes needed for the Modula-2 libraries. Documented in Software Technical Bulletin.

[4] Sun-2,3 only.

[5] Sun-3 only.

[6] Local 3270 gateway runs only on Sun-3; Local 3270 client runs on Sun-2 and Sun-3.

2.2. New Hardware Support

Software support is available for the following new hardware:

- The Sun-4 Workstation
- SCSI 1/2 Inch Tape Support
- Xylogics 7053 VME SMD Disk Controller

  There is a new driver for the Xylogics 7053 SMD disk controller called XD. Up to four controllers are supported in a system with a vme bus. Up to four disks are supported per controller.

- Sun Type-4 Keyboard

  The keyboard is compatible with all Sun-2, Sun-3, Sun-4 and Sun386i workstations.

For information on Sun-4 power-up procedures, tests and PROM monitor commands, and EEPROM programming procedures, refer to the *PROM User's Manual* in the *System Administration Docubox.*

2.3. System Software Changes and Upgrades

This section covers the significant changes and upgrades to system software for SunOS 4.0.

New Kernel Architecture

The SunOS 4.0 kernel is significantly restructured to accomodate a new virtual memory management scheme. This scheme promotes system resource sharing and portability across diverse hardware platforms.

Swap space requirements are generally reduced and system resources are used more efficiently. Caching of frequently accessed data is more efficient and files are treated as part of virtual memory, making access to large files more convenient and efficient.

Diskless NFS Server

SunOS 4.0 handles support for diskless client systems through NFS (Network File System) rather than ND (Network Disk) mechanism. The migration to NFS-supported diskless nodes offers the following benefits:

- Easier maintenance of servers and clients
- Easier mixing of remote and local filesystems
- Cleaner support of multiple architectures
- Minimal disruption to existing programs
- Minimal symbolic link confusion

Support of heterogeneous environments is improved since implementations of the diskless NFS server are available on non-Sun servers. A diskless Sun workstation is now able to boot and swap off a non-Sun server.

Eliminating ND does not affect the number of clients a server can support. No PROM change is required for Sun-2 systems that do not support `tftp` booting, since a user-level boot block server is provided.

For complete information on the new filesystem layout, see Chapter 1 in *Installing the SunOS* and *System and Network Administration*. Also refer to the following man pages: `hier(7)` and `filesystem(7)`.

Filesystem Reorganization

SunOS 4.0 includes a reorganization of the filesystem that simplifies the administration of diskless clients. The reorganization separates host-dependent, nonshared files from architecture-dependent, shared files. This filesystem layout makes it easier for clients of different architectures to work with a single server.

All files and directories that define the individual identity of the machine or are dedicated to the machine, (such as `/etc/passwd`) are now in `/usr/etc`. These include the spool directories from `/usr/spool` and the `adm` files from `/usr/adm` which are now in `/etc/spool` and `/etc/adm` respectively.

Architecture-dependent files, including all executable files and libraries, have been moved to the `/usr` filesystem. `/bin` no longer exists and its contents are now in `/usr/bin` with a symbolic link left behind. The contents of `/lib` are in `/usr/lib` and all executables from `/etc` are now in `/usr/etc`.

The new `/usr` is designed to be mounted read-only. Very few executable files are left in the `root` filesystem; these include:

- `vmunix`
- `init`
- `sh`
- `ifconfig`
- `hostname`
- `mount`

All executable files in the root directory, except `vmunix` are now located in the new `/sbin` directory. Because of the filesystem reorganization, `/usr` must be mounted even when booting into single user mode.

See *Installing the SunOS* and *System and Network Administration* for more information.

Automounting of Remote Filesystems

The optional automount facility automatically mounts filesystems transparently. The automount command invokes a background daemon that intercepts directory references and mounts accessible remote filesystems as needed. Automatic dismounting occurs after a specified period of inactivity. Remote filesystem mounting uses YP (Yellow Pages) maps and local map files.

Changes to `boot`

Prior to SunOS 4.0, booting a stand-alone workstation or a server from a local disk was done through code in `boot` which accessed and interpreted a file system on a disk.

The following changes have been made to `boot` in SunOS 4.0:

- ND code has been eliminated, since servers no longer support ND operations.

- The `boot(8)` program now understands how to perform NFS file operations over the net to a server with which the client workstation is registered.

For a complete explanation of this see *System and Network Administration* (800-1733).

New Virtual Memory System

The memory management facilities of SunOS have been completely replaced for 4.0. The new implementation unifies the system's operations on memory objects around the concept of file mapping. This treats processor memory as a very large cache of file pages. The affects of this change include the following:

- A fully implemented `mmap` system call supporting regular files in addition to a few character special devices.
- Changes in the system's handling of memory sharing so that sharing occurs on the basis of single file pages, rather than entire program texts.
- A more general structure to the address space available to applications.

For more information, see Chapter 5 of the *System Services Overview.*

Shared Libraries

The system supports a *shared libraries* facility built upon a new dynamic linking capability and the file mapping facilities provided by the new virtual memory system. Shared libraries are constructed through the application of several changes to each of the following: compiler, assembler, and link-editing tools.

SunOS 4.0 is integrally dependent on shared libraries. Almost every program in the release uses at least the shared C library. The window tools use shared versions of the Suntools and Sunwindows libraries.

Library sharing reduces program size and swap space requirements, and simplifies the incorporation of revisions to libraries.

You can build your own shared libraries.

The C compiler and assembler have been enhanced to generate position independent code (PIC) used to build shared libraries.

SunOS 4.0 includes the following shared libraries:

- BSD and System V versions of `libc`
- `libkvm` for interpreting the kernel's address space
- `libsunwindow` and `libsuntool`, the window system libraries
- `libpixrect`, the Pixrects library

The archive (`.a`) form of these libraries are also included in SunOS 4.0.

For more information, see *Programming Utilities and Libraries.*

Resizable Swap Area

Resizing diskless client swap space no longer requires taking a server and its clients off line or reinstalling the system. Only clients whose swap space is being modified need to be halted. The resizing process is transparent to other clients.

NIT Streams Instead of Sockets

The Network Interface Tap is now available through the System V STREAM mechanism instead of the socket mechanism. A general packet filter is now provided as well as a STREAM module.

4.3BSD Enhancements

The items in this section are enhancements from 4.3BSD in SunOS 4.0.

Alternative Protocol Architectures

The kernel can now be configured to support other protocols from a binary release without having to recompile the source to the Ethernet driver. This makes the installation of Sunlink products easier.

`/etc/ttys` Compatibility

The system uses a new style `/etc/ttys` file that is now called `/etc/ttytab`. `init` reads this new file and writes an `/etc/ttys` file with the same relative positions within the file for each tty. The `/etc/ttys` file written by `init` is read-only. Users are not allowed to modify it directly since doing so does not have any effect on `init`. `/etc/ttys` is rewritten whenever `init` reads `/etc/ttytab`. Sun programs written under previous releases still run compatibly with the `/etc/ttys` emulation.

Since `init` no longer uses `/etc/ttys`, any administrative procedures that modify it must be changed to modify `/etc/ttytab` instead.

The new `/etc/ttytab` file format is completely compatible with 4.3BSD, but the file name is different. It is possible for `/etc/ttys` to be a symbolic link to `/etc/ttytab` (or vice versa) where complete 4.3BSD compatibility is required. Doing this, however, breaks compatibility with old Sun programs. See `ttys(5)` and `ttytab(5)` for more information.

`/etc/ttytab` includes the information that used to be in `/etc/ttytype` and `/etc/securetty`.

`fsck(8)`

`fsck(8)` now creates and grows directories. This allows it to rebuild the root of a file system as well as create and enlarge the `lost+found` directory when necessary.

FTP Fully Implemented

The File Transport Protocol incorporates several new commands in SunOS 4.0. The default transfer type now is ASCII, so transfer of binary files requires an explicit command. See `ftp(1c)` and `ftp(8c)`.

Full ICMP Support

Prior releases did not implement all of the ICMP (Internet Control Message Protocol) which caused problems on multi-vendor networks such as the Defense Data Network.

Full IP Subnetting

The restrictions on IP subnets in Release 3.3 have been removed. In SunOS 4.0, each interface can have its own network mask. A new YP map `netmasks.byaddr` is used to enable subnets.

`getty(8)`

`getty(8)` is upgraded to the 4.3BSD version. It now uses `/etc/ttytab`.

`inetd` compatibility

The new `inetd` uses a different file format in a file of a different name (`/etc/inetd.conf` vs. `/etc/servers`).

The new `inetd` uses different conventions to start the programs it runs. Previously, `inetd` would call the program with a single argument which contained the port number the connection was on. In SunOS 4.0, `inetd` gets the port number by using `getpeername()`.

SunOS 4.0 `inetd` is completely compatible with the 4.3BSD `inetd`. Previous Sun programs should work without change with the new `inetd`, as long as the `/etc/inetd.conf` file is set up properly. For more information, see `inetd.conf(5)` and `inetd(8c)`.

`inetd` now supports selected Internet services internally, such as echo, day, and character generator.

IP Options

More general support is provided for options at the Internet level.

Line Printer Spooler

The line printer daemon, `lpd(8)`, has 4.3BSD enhancements incorporated. Error logging is handled with the new `syslog` daemon. The `/etc/hosts.lpd` file can be used to extend remote access. This is in addition to the `/etc/hosts.equiv` file. Various bug fixes have also been incorporated both from 4.3BSD and customer reports to Sun.

The other line printer support facilities, `lpr(1)` and `lpc(8)`, have also been enhanced with the changes from 4.3BSD. These facilities include the `up` and `down` commands in `lpc(8)`, and the support for restricted access printing in `lpr(1)`.

`login`

`login` is upgraded to the 4.3BSD version. When you log in, your terminal is changed to be owned by the `tty` group and has the `rw--w----` mode. Programs such as `write` that write to other user's terminals have been changed to run set-group-ID to group `tty`. These programs only permit printable ASCII characters and white-space characters to be written unchanged. This closes a security hole caused by permitting arbitrary programs to write to other user's terminals.

New Buffering Conventions

The kernel uses 4.3BSD conventions for handling `mbuf` structures within the socket system. Customer network drivers may need to be rewritten.

New Interface Structure

The kernel `ifnet` structure has changed as in 4.3BSD to allow more generality, such as the use of a single interface by several different address families. All Sunlink products require new releases for SunOS 4.0.

`passwd`

`passwd(1)` is upgraded to the 4.3BSD version. It now permits you to change the full-name and login shell for your entry in `/etc/passwd`.

NOTE *These features are not present in* `yppasswd(1)`. *You can change your password in the Yellow Pages database, but not your full-name or login shell.*

`ps`

`ps(1)` is upgraded to the 4.3BSD version. By default, it now prints wait channels in symbolic form, rather than numeric form. Many wait channels, such as the wait channel for a process blocked doing I/O on a stream (such as a terminal), do not have a symbolic name; they print out as `kernelma`.

Due to differences between the 4.3BSD and SunOS virtual memory implementation the following conditions result:

- The `-s` flag is not supported.
- The `ADDR` field is not printed when the `-l` flag is specified.
- The `TSIZ` and `TRS` fields are not printed when the `-v` flag is specified.
- When the `-u` flag is specified, the start time of the process is printed.
- The `-r` flag restricts the printout to processes that are running or sleeping on a *fast wait* (sleeping with negative priority).

`rcp`

`rcp(1c)` is upgraded to the 4.3BSD version. This version no longer supports host/userid specifiers of the form *host.user* because domain-based hostnames can contain periods.

Socket Process Group IDs

The interpretation of the process and process group ID for a socket as set by the `F_SETOWN fcntl` operation or the `IOCSPGRP ioctl` operation has changed.

Prior to SunOS 4.0 the 4.3BSD guidelines were as follows:

- A positive value was a process group ID.
- A negative value was a process ID.

In SunOS 4.0, the 4.3BSD guidelines are as follows:

- A positive value is a process ID.
- A negative value is a process group ID.

`syslog` Compatibility

In SunOS 4.0, the `syslog` facility is now compatible with 4.3BSD. This results in the following incompatibilities:

- Pre-4.0 program logging to 4.0 `syslog` daemon

  Pre-4.0 programs log messages with no facility code but with priorities in the range 1 - 9. Since the 4.0 `syslog` accepts priorities in the range 0 -

7, priorities 8 (LOG_INFO) and 9 (LOG_DEBUG) look like priorities 0 (LOG_EMERG) and 1 (LOG_ALERT) from facility 1 (LOG_USER). Unfortunately, this has the effect of making low priority messages seem to be much higher priority than they really are.

Almost all of the values for logging levels have changed. This causes, for instance, messages logged at the old LOG_CRIT level to be logged at the new LOG_NOTICE level. In general, old log messages appear to be less important than intended.

The 4.0 `syslog` daemon forces messages that claim to be from the LOG_KERNEL facility to look like they came from the LOG_USER facility, unless they come from the local kernel. The `syslog.conf` file on the "loghost" machine is set up to log all LOG_USER messages in the log file used to log LOG_MAIL messages.

- SunOS 4.0 program logging to pre-4.0 `syslog` daemon

  All SunOS 4.0 programs using `syslog` send their log messages to the local `syslog` daemon. The default 4.3BSD `syslog` configuration file causes all `syslog` messages to be logged in local files, although it does provide a facility to forward the `syslog` message on to a `syslog` daemon on another machine. `sendmail` log messages and "authorization system" log messages are forwarded to "loghost".

  These forwarded messages include a facility code in their log message. Except for LOG_USER|LOG_EMERG (== 8) and LOG_USER|LOG_ALERT (== 9) (which, by default, are not forwarded to the `syslog` daemon at "loghost"), these all cause the priority field in the message to be two digits. The old `syslog` daemon does not understand multi- digit priority fields, and therefore logs the message with a default priority of LOG_ERR (== 4). Using the default configuration file, this causes the message to be logged with all the `sendmail` log messages in `/usr/spool/log/syslog`. For more information, see `syslog(3)` and `syslogd(8)`.

`syslog` Daemon

The `syslog` daemon has been upgraded to the 4.3BSD version. It reads kernel `printf` messages from a new device and logs them. Many system daemons have been changed to log to the `syslog` daemon as well.

`su`

If the `wheel` group (group 0) has members, only they can *su* to `root`, even with the root password. Successful and unsuccessful attempts to `su` to root are logged to the `syslog` daemon.

`tcopy(1)`

`tcopy(1)` is a new tape copying program that preserves tape blocking.

TCP Performance

The TCP software now estimates the round-trip time as well as the variation of round-trip time. This provides for continuously good performance on both fast and slow networks.

TCP Urgent Data

The interpretation of TCP Urgent data has been changed to be closer to the official specification.

Telnet Improvements

The SunOS 4.0 `telnet` daemon supports terminal type negotiation. The `telnet` program now supports optional local X-ON/X-OFF flow control. See `telnet(1c)`.

`tftp`

`tftp(1c)` is upgraded to the 4.3BSD version. The default transfer mode is now ASCII. To transfer to a non-ASCII file, binary mode must be explicitly selected.

The SunOS TFTP server `tftpd` is normally configured to run in a *secure* mode. This only allows for bootstrapping machines over the network and for transfers from the `/tftpboot` directory. TFTP should not be used for general file transfer operations unless the TFTP server on the remote machine is configured to permit this. As TFTP requests carry no access credentials, TFTP servers often do not permit general file transfers. Other mechanisms, such as FTP, `rcp`, `rdist`, or NFS, should be used to access files on remote machines.

`tftpd`

`tftpd(8C)` is upgraded to the 4.3BSD version. This version supports both net-ascii and binary transfer modes.

In SunOS 4.0, the TFTP server also supports special modes that specially configure it to act solely as a bootstrap server for booting diskless machines over the network. By default, the TFTP server operates in this mode, which leaves it inoperable for general file transfer. It can be configured to run as a general file transfer service, but this can result in security problems as it allows any other machine on the network to retrieve any publicly-readable file.

`tset`

`tset(1)` is upgraded to the 4.3BSD version. It sets the terminal driver's notion of the screen size from the number of columns and lines in the `termcap` entry for the terminal. The Sun console driver ignores this, however, so the size will be hardwired at the value configured in the PROM.

2.4. General Software Changes

This section covers the general software changes incorporated in SunOS 4.0 under the following categories:

- General Conceptual Changes
- System Call Changes
- General C Library Changes

General Conceptual Changes

This section covers the general conceptual software changes and enhancements in SunOS 4.0.

C Program Exit Status

In a C program, if the main function `main()` returns, its return value is used as the exit status of the program. Previously, 0 was used as the exit status if `main()` returned; this change makes SunOS more compatible with 4.3BSD and System V. Note, however, that some erroneous programs do not return **any** value from `main()`. Thus, they can return a non-zero exit status.

Filename Completion

Filename completion is a new feature in SunOS 4.0. In addition to the wild card characters (?,*), the C shell provides a *filename completion* utility that fills in the rest of a filename after you type a few characters.

To implement this feature, you need to include the following line in your `.cshrc` file or type it on the command line:

```
set filec
```

Once included in your `.cshrc` file, you can type in the first few letters of a file's name and the C shell will complete the rest for you.

NOTE *Currently, filename completion does not work in SunView command or text windows unless scrolling in that window is disabled. Filename completion works in* `shelltool` *windows. See the SunView Beginner's Guide for information on how to enable and disable scrolling.*

For more information on *filename completion*, see *Doing More with SunOS: Beginner's Guide.*

Group ID for Newly-Created Files

The rules used to specify the group ID that own a newly created file have changed in SunOS 4.0. Previously, the group ID for any newly-created file was set to the group ID of the directory in which the file was created. Now, if the set-GID bit is not set on a directory, if a file is created in that directory its group ID will be set to the effective group ID of the process that created the file. This is the behavior exhibited by non-4.2BSD systems, including System V. If the set-GID bit is set on a directory, if a file is created in that directory its group ID will be set to the group ID of the directory. This is the behavior exhibited by 4.2BSD and SunOS releases prior to 4.0.

If a directory is created, its set-GID bit is set if the set-GID bit was set in the directory in which it was created.

All directories that are part of the SunOS 4.0 release have the set-GID bit set. By default, the behavior for files created in those directories, or directories created under those directories, will be the same as the behavior in previous releases. Directories in existing file systems do not have this bit set, so the default behavior for files created in those directories changes.

When you install SunOS 4.0, do one of the following:

- Use the `find` command to find all directories in those file systems and turn the set-GID bit on for those directories.
- Mount the file systems with the `grpid` mount option that specifies that the 4.2 BSD behavior should be exhibited for all newly-created files in directories in that file system, regardless of the setting of the set-GID bit.

The `make` command is enhanced. It is upwardly compatible with System V `make`, and the following additional features:

- Hidden dependency checking
- Conditional macros
- Pattern-matching implicit rules

For more information, see *Programming Utilities and Libraries*. Appendix B is a summary of enhancements; Chapter 8 is a tutorial for using the SunOS 4.0 version of `make`.

New Directory Entry Format

SunOS 4.0 directory entries are returned in a new format. The associated changes for the new directory entry format have been made in the kernel, user libraries, and include files. The changes to directory entry format include the following:

- A new `d_off` field has been added to the directory-entry structure. This field is a cookie that is only interpretable by the filesystem type that generated it. The `d_off` field contains the offset of the next entry in the directory. The only valid use for it in a user program is as an argument to `lseek(2)` to seek to the next entry in a directory.
- The new directory entry format is contained in `/usr/include/sys/dirent.h`, a new include file. The name of the structure in this file has been changed from `struct direct` to `struct dirent`. A second new include file, `/usr/include/dirent.h` contains the directory information relevant to the C library `directory(3)` routines and the `/usr/include/sys/dirent.h` file. These two new files supersede the `/usr/include/sys/dir.h` file.
- The `/usr/include/sys/dir.h` file remains in SunOS 4.0 and has a new directory entry structure including the `d_off` field. The `struct direct` name is retained. This allows programs which used the C library `directory(3)` routines and the `/usr/include/sys/dir.h` file to re-compile and run under SunOS 4.0 without modifications to the source code.
- The vnode/VFS interface routine, `vn_readdir()`, now returns directory entries in the new format.
- A new system call, `getdents(2)`, supersedes `getdirentries(2)` and returns directory entries in the new format. `getdirentries(2)` returns directory entries in the old format and can not be used with the `/usr/include/sys/dir.h` file. `getdirentries(2)` remains in

SunOS 4.0 to preserve binary compatibility. It will be removed in a future major release.

- The C library `directory(3)` routines have been modified to use the new directory entry format contained in `/usr/include/sys/dirent.h` and the `getdents(2)` system call.
- Some directory entry changes are transitional to preserve compatibility across releases. The `getdirentries(2)` system call and the `/usr/include/sys/dir.h` file are of this nature.

Does This Affect You?

The possible affects of the directory entry changes to your environment include the following:

- If you have a pre-4.0 binary which uses the `getdirentries(2)` system call and/or the `directory(3)` library routines, your binary should continue to run correctly without re-compilation under SunOS 4.0.
- If you have a program that currently uses the `directory(3)` library routines and the `/usr/include/sys/dir.h` file, your source should compile and run correctly without changes under SunOS 4.0. `/usr/include/sys/dir.h` will be removed in a future major release, so you should plan to convert your program to use `/usr/include/dirent.h`.
- If you have a program that currently uses the `getdirentries(2)` system call and the file `/usr/include/sys/dir.h`, you must convert it to use the `getdents(2)` system call and the `/usr/include/sys/dirent.h` file so it will run under SunOS 4.0. Note that the writing of programs that directly use the system call is emphatically discouraged. If possible, it is recommended that programs use the `directory(3)` library routines.

NOTE *The filesystem-independent directory entry format was formerly identical to the Berkeley 4.2 directory entry format defined in* `/usr/include/ufs/fsdir.h`. *With the new directory entry format, this is no longer the case. Code which relies on this identity between the two directory entry formats must be modified.*

New Terminal Driver

A new terminal driver is incorporated into SunOS 4.0. It fully supports the functionality specified in the System V Interface Definition, as well as all the functionality of the old V7/4BSD terminal driver. The exceptions to this are the `LTILDE` mode and the *delayed suspend* character. The new driver uses the System V Release 3 STREAMS mechanism.

As such, the driver permits the character size to be set to 5, 6, 7, or 8 bits per character, with or without a parity bit. It supports a full 8-bit data path when 8 bits per character is selected. It does not strip off the 8th bit on output, and can be told not to strip off the 8th bit on input.

If a terminal supports an 8-bit character set, it can be operated in 8 bits, no parity bit mode. The V7/4BSD compatibility features do not work correctly unless the character size and parity specifications are 7 bits plus a parity bit, or 8 bits and no

parity bit. Other settings, such as 5 or 6 bits per character, or 8 bits and a parity bit, should only be used in specialized programs talking to devices that require those settings. They should not be used for regular login terminals.

If you want to use a terminal that supports 8-bit characters, set the terminal modes to a mode that supports an 8-bit data path. Do this with the `stty pass8` command, or use the `p8` capability in `/etc/gettytab`. This also applies to terminals where the 8th bit is controlled by a meta key. EMACS correctly operates in this mode. It turns *raw* mode on and off correctly, leaving the 8-bit data path in place regardless of whether *raw* mode is on or off.

If you are logged into a machine and have not set the terminal driver to have an 8-bit data path, you cannot use `rlogin` to log into another machine and set the terminal driver on that machine to have an 8-bit data path. You must set the 8-bit data path on the first machine you log in to. You can then set it on all other machines you log in to.

`PASS8` mode does not strip off the 8th bit on output as it did in SunOS releases prior to 4.0.

The `stty(1V)` command is updated to print and set all new modes.

The following additional functionality is available in the pseudo-tty driver:

- If the baud rate is set to 0 with an `ioctl`, all subsequent I/O on the controller gives an `EIO` error, just as if the slave side had been closed. This is analogous to setting the baud rate to 0 on a real terminal. The DTR drops and causes the other end to hang up. The `EIO` is normally treated as a signal to the process on the controller side to exit.

  This can cause programs that carelessly get terminal modes, without checking whether the `ioctl` fetching the modes succeeded, and then set the terminal modes on a pseudo-terminal from the modes they fetched, to fail. If the fetch fails, the mode structure into which the modes were to be fetched contains the previous values. This may be initialized or uninitialized garbage (this was the case in 3.x also). If the baud rate value is 0, this causes all subsequent I/O on the controller to get an `EIO` error.

- Asynchronous I/O is supported on pseudo-tty controllers. The controller must be put into asynchronous mode using the `F_SETFL fcntl()` call or the `FIOASYNC ioctl()` call. The process group to which the controller belongs must be set with the `F_SETOWN fcntl()` call or the `FIOSETOWN ioctl()` call.

  NOTE: For backwards compatibility, the `TIOCSPGRP ioctl()` call on the controller sets the process group owner of the *slave*, not the *controller*. It should not be used here.

- The number of pseudo-terminals can be configured without source code.

`newgrp`

The `newgrp` command is a new command that changes the group ID. The group ID of the file is set to the user's group ID. `newgrp` allows the user to change his group ID.

In most cases, BSD semantics for group ID's should be in effect and the group ID is inherited from the parent directory. `newgrp` is not required under these circumstances. `newgrp` is required if System V semantics are in effect. For more information, see the `newgrp` man page.

Non-blocking I/O

Non-blocking I/O is now fully supported in both 4.2BSD-style and System V-style. Two separate flags are used to request the different forms of non-blocking I/O. The `FNDELAY` and `FNBIO` flags are defined if a program includes `<fcntl.h>`. `FNDELAY` selects 4.2 BSD-style non-blocking I/O and `FNBIO` selects System V-style.

There are two versions of `<fcntl.h>`: one in `/usr/include` and one in `/usr/5include`.

- `/usr/include` defines `O_NDELAY` as `FNDELAY`.
- `/usr/5include` defines `O_NDELAY` as `FNBIO`.

If a program uses `O_NDELAY` and is compiled with `/usr/bin/cc` it gets 4.2BSD-style non-blocking I/O. If the same program is compiled with `/usr/5bin/cc`, it gets System V-style non-blocking I/O.

`O_NDELAY` is no longer defined in `<sys/file.h>`. Programs that use `O_NDELAY` must include `<fcntl.h>`.

Obsolescence Mechanism

The `/usr/old` directory contains obsolete modules. Modules placed in this directory are subject to removal in future major releases. The following programs have been moved to `/usr/old` in SunOS 4.0:

- `filemerge`

  An enhanced version is provided with the Network Software Environment (NSE) product.

- `sun3cvt`

  This module was needed only for the transition to SunOS 3.0.

- `compact, uncompact`

  These modules are replaced by faster and more efficient, but incompatible `compress` and `uncompress` programs from 4.3BSD.

- `eyacc`

  This module was used to implement Pascal and has been removed in 4.3BSD.

- `make`

  The pre-SunOS 3.4 version of `make` has been replaced.

- `prmail`

  This module has been replaced with `mail -u` in 4.3BSD.

- `pti`

  This module has been replaced with troff -a in 4.3BSD.

- `setkeys`

  Input from `defaults(1)` should be used instead.

Also included in `/usr/old`:

- `coretool`
- `perfmon`
- `setkeys`
- `syslog` (old version)
- `tektool`
- `ttytool`
- `VC`

Open Files Per Process

Prior to SunOS 4.0, the limit was 30 open files per process. In SunOS 4.0, the limit is 64 open files per process.

QIC24 PROM

If you use Sun-3 software and 1/4" format tapes, this section describes the new 1/4" tape format and how to check to see if your Sun-3 workstation needs EPPROM's to read the new format.

To improve the quality of software distribution tapes and reduce software installation time, the format of 1/4" cartridge tapes is changing for Sun-3 products. This change is incorporated in SunOS 4.0 and applies to software cartridge tapes for Sun-3, 68020-based workstations.

Tape formats remain the same for Sun-2, 68010-based software tapes and Sun-4 SPARC-based software tapes.

NOTE *Sun-4 systems already use the new tape format.*

Do You Need a New PROM?

The following Sun-3 products that can be affected by the change in tape format include:

- Sun-3/50
- Sun-3/75
- Sun-3/260 (upgraded from a Sun-3/1xx)
- Sun-3/60
- Sun-3/160

- Sun-3/280 (upgraded from a Sun-3/1xx)

Early revisions of the Sun-3 PROM's do not understand the QIC24 tape format. Any system with a PROM of revision 1.8 or higher reads the new 9-track, QIC24 tapes. To check the revision level of your PROM, use the following steps:

1. Halt the system.

   For information on how to halt the system, see *System Administration Manuals Minibox.*

2. Type `kb` after the ">" prompt from the PROM monitor.

   If you have a revision of less than 1.8, install a newer revision of the PROM.

If you load a QIC24 tape into a workstation that can only read QIC11 tapes, you receive an `86A0` or `86A8` error from the controller. This error indicates that the controller was unable to read the header block of the tape.

NOTE *It is possible that this error is a result from a faulty tape. Check your PROM revision level, or test another tape.*

If You Need a New PROM

Upgrade PROM's for Sun-3's are available from Sun. If you have an On-site Hardware or Comprehensive Support contract, Sun will install the new PROM for you. If you are an on-site hardware support customer, phone the Sun Response Center at 800 USA-4-SUN and request Field Service to schedule your PROM installation. If you want to install the PROM yourself, call the 800 USA-4-SUN number, request Field Service, and ask for the Sun-3 PROM Upgrade Kit to be mailed to you.

If you do not have an On-site or Comprehensive Support contract, Sun will mail you the Sun-3 PROM Upgrade Kit free of charge. The kit contains instructions for replacing the PROM chip on your CPU board, a process that takes about 10-15 minutes. Call the 800 USA-4-SUN number, request Field Service, and ask for a Sun-3 PROM Upgrade Kit. Sun will install the PROM's for you, if you desire, and bill you for installation time only.

Sun-3 systems that are upgraded with the latest version of the PROM continue to read QIC11 tapes. The upgraded system reads all tapes produced before the upgrade.

Signal Handlers

Functions that are specified as handlers for signals are now expected to be of type "function returning `void`", not "function returning `int`". This is the type specified by the current IEEE POSIX standard draft. Since no use is made of the return value of a signal handler, `void` is the correct type of such a function, not `int`.

Code written assuming that these functions were of type `int` will continue to run and need not be recompiled. If this code is recompiled, it does not need to be changed, even though warning messages about type clashes will be printed.

In addition, a number of signals provide and additional `addr` parameter to their handlers. This parameter describes a memory address relating to the exception condition indicated by the signal. For instance, `SIGSEGV` provides an `addr`

as the address reference that causes the signal to be delivered. Signals delivering an `addr` parameter include the following:

- `SIGSEGV`
- `SIGBUS`
- `SIGFPE`
- `SIGEMT`
- `SIGILL`

Time Zones

The handling of time zones has been upgraded. The following are now read from files:

- The offset difference of a time zone in relation to Greenwich Mean time.
- The rules that define when daylight savings time starts and ends.
- The amount of time change when daylight savings time starts and ends.
- The time zone names for standard and daylight savings time.

By default, a file that contains the rules for the local time zone is used. If the `TZ` environment variable is set to the name of a file containing the rules for a different time zone, that file is used instead; this applies to all utilities except selected ones, such as `uucico`, that must always use the local time zone.

These files are generated from a textual description of the rules. The text files from which the distributed set of rule files are generated, and the command `zic` used to generate them, are provided.

New routines `timelocal` and `timegm` are provided that convert a date and time, specified as month/day/year/hour/minute/second, from local or Greenwich Mean time respectively to the standard UNIX system date/time format. They perform the inverse of the conversion performed by the `localtime` and `gmtime` routines. These routines use the information from the appropriate time zone file; applications should not perform this conversion themselves.

The structure returned by `localtime` now contains extra fields that indicate the name and the offset from GMT of the current time zone at the time specified by the argument to `localtime`.

The command `tzsetup` attempts to set the kernel's notion of the offset from GMT and DST rule type based on the default time zone rules; this is done so that binaries built prior to 4.0 will run. In some cases, there is no set of DST rules that will work; if this is the case, `tzsetup` will indicate that DST is not observed. This program is run at boot time; there is no longer any need to specify the offset or DST rule type when building a kernel.

`vacation`

In SunOS 4.0, `vacation` is run with no arguments and now allows you to interactively turn it on and off. It creates a `.vacation.msg` file or allows you to edit an existing `.vacation.msg` file using the editor specified by the `VISUAL` or `EDITOR` environment variable. `vi(1)` is used if the environment variables are not set.

If a `.forward` file exists in your home directory, it asks whether you want to remove it and turn off `vacation`. If a `.forward` file does not exist in your home directory, it creates one for you and automatically performs `vacation -I` to turn on `vacation`.

For more information, see *Mail and Messages: Beginner's Guide.*

`ypset` and `ypbind`

In SunOS 4.0, the `ypset` and `ypbind` commands have greater security. You are no longer able to set the Yellow Pages server of the local or remote machines unless you have an effective UID of 0.

8-bit Characters in Filenames

The file system code has been changed to permit file names to contain characters with the 8th bit set. Previously, attempts to create or manipulate such files were rejected. Note that such files are impossible to remove using the C shell, as the C shell still uses the 8th bit to quote characters; the Bourne shell, which has been upgraded in 4.0, and now does not use the 8th bit to quote characters, must be used to remove these files.

System Call Changes

This section covers the system call changes and enhancements for SunOS 4.0.

`close`

Prior to SunOS 4.0, a `close` on a descriptor on which an `mmap` had been performed resulted in an implicit `munmap` on the mapped pages. This has been eliminated in SunOS 4.0. Mappings remain established even after the descriptor from which they were obtained is closed. The only way to explicitly remove a mapping is through the `munmap`, `exec`, `brk`, and `exit` system calls.

`exec`

`NMAGIC` (0410) format executable files no longer have their *shared text* behavior. They still provide a write-protected text area for the program, but the memory used to hold that text is no longer shared with other processes running the same program.

The parameter `NCARGS`, describing the number of characters which can be passed in the argument vectors to `exec`, has been greatly increased from 10240 to 1048576 (1Mbyte).

The default stack size for a new process has also been changed as a result of the `NCARGS` change, from .5 to 2Mbytes.

The address space established by `exec` consists entirely of `MAP_PRIVATE` mappings to the memory holding the program and stack (see `mmap` below).

`fork`

Prior to SunOS 4.0, `fork` copied the parent's address space when creating a child. In SunOS 4.0, `fork` copies the *mappings* describing an address space *not* the address space itself. `exec` and other system operations specify `MAP_PRIVATE` mappings when address space objects are created. This behavior is completely compatible with previous releases in that changes made in parent and child are invisible to each other.

For those applications which employ `MAP_SHARED` mappings to regular files (new in SunOS 4.0), a `fork` which copies such a mapping provides a means for parent and child to communicate through a shared memory region.

`getdents`

A new System V compatible `getdents()` system call has been added to read entries. As with the old `getdirentries()` system call, direct use of this call should be avoided. Use the `directory`(3) routines instead.

The `getdirentries()` system call is still supported by the kernel in SunOS 4.0. The C library interface for `getdirentries()` has been removed, however. This means that pre-SunOS 4.0 binaries that are not invalidated by some other change and use `getdirentries()` should continue to run.

NOTE *Source programs calling* `getdirentries()` *must be converted to use* `getdents`.

`kill`

The `kill` system call now supports the System V rule for processes that are allowed to send signals to other processes. A process is now allowed to send a signal to another process in either of the following cases:

- The sending process has an effective user ID of super-user.
- The real or effective user ID of the sending process is equal to the real or saved set-user ID of the receiving process.

`mincore`

This is a new system call to return the residency of pages mapped into an address space.

`mmap`

The `mmap` system call is now fully implemented. In addition to character-special files such as frame buffers, programs can now map almost any random-access memory object into their address space via `mmap`. The most common usage is to map a regular file.

File access through mapping is extremely efficient. The system can share multiple accesses with the same physical memory resources. The overhead involved in copying data via `read` and `write` is avoided with mapped files. Files can be mapped so that changes made by the mapping process are either invisible to or shared with others.

File mapping is used as the basis for other system operations such as `brk` and `exec`. Other operations, such as `fork` have been respecified to deal with the new system structure. In these cases, the calls are completely backwards compatible. However, some behaviors are slightly different. These instances are noted elsewhere in this document as needed.

The programmer can use the whole address space supported by a processor. The program is no longer constrained to just the text, data, and stack *segments* formerly provided by the system. These segments exist as conventions of the language tools rather than requirements of the system's operation. As a result, a program is free to treat its address space as a simple vector of pages, each of which can be manipulated as an independent entity.

Although `mmap` is compatible with pre-SunOS 4.0 binaries, to support the new functionality, and to be compatible with the system specification for Berkeley 4.3BSD UNIX, the `mmap` system call is *source-code incompatible* with previous releases of SunOS. This means that pre-SunOS 4.0 binaries that call `mmap` will continue to run. However, sources containing a call to `mmap` must be changed to run correctly under SunOS 4.0.

The changes to `mmap` include the following:

- The `addr` parameter was previously used to specify the location in the address space at which mapping was established. In SunOS 4.0, `addr` serves this purpose *only* if the flag `MAP_FIXED` is included in the `flags` parameter of the call. `flags` was previously called `share`. Without this flag, `mmap` determines the appropriate address for the mapping.
- The return value of a successful `mmap` is *always* the address at which the system placed the mapping. `mmap` should be declared as a function of type `caddr_t`. `<sys/mman.h>` takes care of this automatically. The *only* error return from a `mmap` call is the value `(caddr_t)-1`. In particular, code sequences of the following form used to check errors are *not* correct.

```
if (mmap(...) < 0)
```

`mprotect`

This is a new system call to change the access protections on memory mappings.

`msync`

This is a new system call to synchronize mapped addresses with their backing storage.

`O_SYNC`

The flag `O_SYNC` may be specified in the flags argument to `open` and `fcntl`; if this flag is set on a file descriptor that refers to a regular file, all "write"s to that file will block until the data is completely written to disk. This happens regardless of whether the file is on a local file system or is being accessed over NFS.

NOTE `O_SYNC` *has no effect on files accessed by using* `mmap`.

`ptrace`

There are two new `ptrace` requests:

- `PTRACE_SYSCALL` which is used by the system call trace command `/usr/bin/trace`.
- `PTRACE_DUMPCORE` which is used by the `/usr/ucb/gcore` command.

A restriction has also been removed on the `PTRACE_DETACH` request. It is no longer necessary for the process to be stopped.

For more information, see the `ptrace(2)` man page.

The restrictions on the `PTRACE_POKETEXT` and `PTRACE_WRITETEXT` prohibiting the writing of shared text have been removed.

`select` — The `select` system call is upgraded to the 4.3BSD version. It now handles more than 32 file descriptors. Macros have been provided to manipulate sets of descriptors.

`truncate/ftruncate` — The `truncate` and `ftruncate` calls now *set* the length of a file. These calls now allow you to extend a file, in addition to shortening it.

`utimes` — The `utimes` system call now accepts a `NULL` pointer as its second argument. If a `NULL` pointer is specified, `utimes` attempts to set the accessed and modified times on the specified file to the current time. You must either be the owner of the file or have write permission on the file for this command to succeed.

General C Library Changes

This section describes the changes in the general C library for SunOS 4.0.

New `hostent` Structure — Any programs that call `gethostbyname`, `gethostbyaddr` or `gethostent` need to be recompiled. The new `hostent` structure contains a list of addresses instead of just one. Many programs, however, only look at the first entry.

`printf()` and `scanf()` — The `printf()` routines now support the `%i` and `%li` format items; they are synonyms for `%d` and `%ld`. The `scanf` routines now support `%i`, `%hi`, `%li`, and `%n`. `%i`, `%hi`, and `%li` are similar to `%d`, `%hd`, and `%ld`, respectively, except that if the number being converted begins with "0x" or "0X" it is assumed to be hexadecimal, and if the number begins with "0" it is assumed to be octal. `%n` returns the total number of characters that have been scanned so far by the current `scanf` call. These are from the current ANSI C draft standard.

`regexp` — The `regexp` regular-expression scanner has been upgraded to support the "\<" and "\>" characters from `vi`. If a regular expression is enclosed in "\<" and "\>", it is constrained to match a "word"; the "\<" must match the beginning of a "word", i.e. the beginning of a line or just before a letter, digit, or underline and after a character (not one of these), and the "\>" must match the end of a "word".

2.5. Graphics Software Changes

The major changes and enhancements for SunOS 4.0 in graphics software are in the following products:

- GPSI

- Pixrect
- SunCGI

GPSI

In SunOS 4.0, GPSI supports the GP2 graphics accelerator and the CG5 color board. There is also a new set of macros that provide source code compatibility for GPSI application code between Sun3 and Sun4 architectures.

Please refer to the *GPSI Programmers Guide* for more specific information.

Pixrect

The significant enhancements to *Pixrect* for SunOS 4.0 include:

- `pr_flip()`
- Pixrect shared library facility

`pr_flip()`

The use of the Intel 80386 processor by the Sun-386 products brings with it the issue of portability between different workstation architectures. The Sun-386i is based on the 80386 processor, which handles byte ordering differently than the 680X0 and sparc processors used by other Sun workstations.

`pr_flip()` was added to Pixrect to facilitate portability across these different workstation architectures. See the *Pixrect Reference Manual* for more detailed information.

Pixrect Shared Library Facility

Pixrect implements a shared library facility in SunOS 4.0.

SunCGI

CGI was a proposed standard when SunCGI was written. CGI does not appear to be moving towards approval in the near term, while GKS, PHIGS, and CGM have been approved as standards. The process of transition to new standards means a phase-out of SunCGI in a future major release. This provides a transition period for customers to new software technology.

SunCGI is fully supported in SunOS 4.0 and includes major bug fixes and a revised manual. See the *SunCGI Reference Manual* for more information.

2.6. Diagnostics

The primary changes to diagnostics in SunOS 4.0 are in `sysdiag` which is covered below.

`sysdiag`

This section covers the changes in `sysdiag` for SunOS 4.0.

`ipctest` and `sunlink`

In SunOS 4.0, `sysdiag` will not contain the `ipctest` or `sunlink` tests. The SunOS 4.0 versions of SunIPC, Sunlink Data Communications Processor (SCP), and Sunlink Multiprotocol Communications Processor (MCP) software required to test the system will not be available until after SunOS 4.0.

The `sysdiag` tests are used to verify the functionality of the SunIPC, SCP, and MCP boards within the system. The `ipctest` and `sunlink` tests will be reintegrated into `sysdiag` at the earliest possible UNIX release.

pmem Test

The pmem test on the SunOS 4.0 version of `sysdiag` should not be run on Sun-3/260 and Sun-3/280 systems. Instead, choose the `Select Mode` or `Single Test Mode` of `sysdiag`, not pmem test.

2.7. Utilities

This section covers the changes and enhancements to user-level commands and utilities that are incorporated into SunOS 4.0.

`etherfind`

The `etherfind` program incorporates new options that print out more information that help in tracking down network problems. `etherfind` also now understands Sun RPC headers. Refer to `etherfind(8)`.

`file`

`file` has been enhanced to recognize dynamically linked objects and programs that require dynamic linking in order to execute.

`finger`

The `finger` command has changed so that it does not print entries for `shelltool` and `cmdtool` windows. It also determines the idle time for users logged in on the console from the idle time of `/dev/kbd` and `/dev/mouse`, so that meaningful results are diaplayed on workstations. In addition, it prints out the *comment* field for a terminal port's entry in `/etc/ttytab` as the location of a user logged in on that port.

`format`

`format` is a SunOS utility that allows you to format, label, repair and analyze disks on your Sun system. Unlike previous maintenance programs, `format` runs under SunOS. This offers a user friendly, menu based interface to disk maintenance. For complete instructions on how to use `format` see *System and Network Administration.*

`gcore`

The `gcore` command has a new `-o` flag that specifies the name of the core file. For more information, see the `gcore` man page.

`grep`

`grep` now uses the `regexp` package to interpret regular expressions. `grep` regular expressions are now compatible with those of `ed`.

`id`

The `id` command now prints your group set, as well as real and effective user and group IDs.

`install`

Prior to SunOS 4.0, `install` was a shell script. In SunOS 4.0, `install` has changed to a C program. For more information, see the `install` man page.

`ldd`

List Dynamic Dependencies, `ldd`, is a new utility that reports the dynamically linked objects (generally shared libraries) on which an executable depends for its execution.

Mail Transport System

This section covers the changes and enhancements to the *mail transport system* in SunOS 4.0.

Automatic Domain Configuration

`sendmail` can use the domain name set in the kernel instead of having to modify each `sendmail.cf` file.

Error Message Improvements

Several minor improvements were made to the error messages generated by `sendmail`. For example, more messages from mailing lists are sent to the owner of the list instead of the sender of the message.

Inverted Alias Mapping

A new mechanism is provided in `sendmail` that can rewrite an address through any Yellow Pages' map. A new map is provided that contains the inverse of the `mail.aliases` map, so that mail going outside of a domain can be simplified.

Mail Exchanger Support

In addition to the normal version of `sendmail` that uses the Yellow Pages to resolve names, another version of `sendmail` is supplied in SunOS 4.0 that uses the domain name resolver directly. This version can be used on the Defense Data Network and to access Mail Exchanger (MX) records.

Mailboxes on Servers

Workstations can use NFS to mount mailbox directories from file servers. Outgoing mail can be sent through the machine from which the mailbox directories are mounted. Typical diskless workstations should no longer need to run `sendmail` daemons.

`mkdir`

The `mkdir` command has a new `-p` flag to create parent directories. For more information see the `mkdir` man page.

`on` **Command Suspension**

Commands started with the remote execution service, the `on` program, can be suspended (for example, by typing Control-Z) and continued.

`pstat`

`pstat` has been enhanced to print information about the queues for all active streams when the `-S` flag is specified. The `-u` flag has changed to take the process ID of the process whose area is printed out, rather than the address of that U area.

SunOS 4.0 does not have shared text structures, therefore, the `-x` flag is not supported. The `ADDR` and `TEXTP` fields are not printed when the `-p` flag is specified. Swap space allocation is reorganized in SunOS 4.0 and therefore, the information printed when the `-s` flag is specified is different.

`reboot`

The `reboot` command has a new `-d` flag to force a crash dump. It now provides boot arguments. For more information, see the `reboot` man page.

`sed`

`sed` now uses the `regexp` package to interpret regular expressions. `sed` regular expressions are compatible with those of `ed`, and the `\<` and `\>` operators from `vi` are supported.

`suninstall`

`suninstall` is the new installation tool replacing `setup`. It is a terminal based interface that provides a user-friendly installation editor which allows you to customize your systems and configurations.

`tftp` **Defaults to Secure**

The *trivial file transfer protocol* should only be used to bootstrap machines over the network. Use programs such as `ftp` or `rdist` to transfer other files. The default transfer mode to `tftp` is now ASCII, so binary files need an explicit command. See `tftp(1)`.

`trace`

`trace` is a new command that is a system call tracer. For more information, see the `trace` man page.

`/usr/bin/troff`

The `troff` command in SunOS 4.0 now supports PostScript™ printers instead of the CAT/4 phototypesetter. This is based on the assumption that the CAT/4 is obsolete.

The following related modifications are incorporated in `troff`:

- The default font width tables now correspond to the Times Roman, Italic, and Bold fonts supplied with PostScript™.
- The maximum line length has been increased from 6½ to 11 inches, which makes landscape mode usable.
- The error message `Typesetter busy` has been changed to `No /dev/cat: try -t or -a`.

The first modification means that users can produce output with `troff -t`, then print quality documents with `lpr -t`. Before, it was necessary to have TranScript™ installed or remote mounted, and then to invoke `ptroff`.

Also, two ligature-related bugs in `troff` were fixed. The first fix prevents the delayed interpolation of the `f` number register. The second fix avoids letters in the wrong font if a font change takes place in the midst of a ligature, when inside a diversion.

`/usr/pub/eqnchar`

The specially constructed mathematical symbols documented by `eqnchar(7)` are now optimized for PostScript™ printers, rather than for the CAT/4 phototypesetter. The `eqn` symbols now defined in `/usr/pub/eqnchar` look best when used with old `troff` and TranScript™ software.

2.8. New Security Features

The following security enhancements have been incorporated into SunOS 4.0:

- Improved network security, with DES authentication of user and host, and public key cryptography.
- An install-time option to run the system at a moderate level of security, patterned after the widely accepted C2 classification.†

† Defined by the National Computer Security Center (a branch of the NSA), the C2 category adds password hiding and security auditing to the UNIX system.

To improve network security, a new set of RPC library routines offers DES authentication to check the validity of both user ID and host address. Previously, UNIX authentication checked only the validity of user ID, which allowed users to impersonate each other over the NFS.

To meet C2 specifications, Sun's operating system was extended to provide improved password security, and flexible, fail-proof auditing of all events that affect security. Other extensions involve single-user booting, enhanced yellow page security, and stricter permission settings for system files. For more information, see *Security Features Guide.*

NOTE *While SunOS 4.0 should meet C2 specifications, Sun Microsystems does not intend to have the system actually certified as C2 secure.*

2.9. System V Enhancements

This section covers the enhancements to System V Release 3 in SunOS 4.0.

New `at` and `cron`

The System V, Release 3 `cron`, `at`, and `crontab` utilities have been provided.

The form of the `crontab` file has not changed. Each user, however, can now have a `crontab` file so that jobs that run from that `crontab` file run with that user's specified privileges. The `crontab` files are located in the directory `/var/spool/cron/crontabs`. The `crontab` file for a particular user carries the user's login name as its file name. The standard system `crontab` file is owned by `root`.

The `crontab` command is used to create, update, delete, and list `crontab` files. These files must not be edited directly, as `cron` does not reread them automatically.

The `at` command is upwardly compatible with the previous version. However, it now supports more than one job queue. One queue that it supports is a "batch" queue; jobs submitted to this queue run "as soon as possible". `at` can be told to limit the number of jobs that run simultaneously in any particular queue. This queue can be used to limit the number of simultaneous background jobs running on the system.

The set of users that can use `cron` and `crontab`, or `at`, can be restricted by the system administrator. By default, all users can use either `cron` or `at`.

For more information, see *System and Network Administration* and *Doing More with SunOS: Beginner's Guide.*

`curses/terminfo`

The `curses` library and `terminfo` database have been upgraded to the System V, Release 3.1 versions. This version fixes many bugs, and is faster and more compact than the previous version; it also includes many new capabilities. It supports eight-bit character sets.

ed

The `/usr/5bin/ed` and `/bin/ed` commands have been merged into one version of `ed` that is compatible both with the 4.3BSD and the System V, Release 3.0 `ed`, with one exception: if the `%` character appears by itself in the replacement string of an `s` command, it restores the replacement string used in the previous `s` command.

If `ed` scripts containing `s` commands with a single `%` as the replacement string are changed by putting a backslash before the `%`, they work in all known versions of `ed`. If the `%` is not the only character in the replacement string, it has no special meaning and the command need not be changed.

For example, the following command:

```
s/#/%/
```

Should be changed to:

```
s/#//
```

The following command does not need to be changed:

```
s/60 percent/60%/
```

`ed` now uses the `regexp` package to interpret regular expressions and supports the `\<` and `\>` operators from `vi`.

The new version of `ed` returns a non-zero exit status if an error occurs. An attempt to run `ed` on a non-existent file is considered an error. In this case, `ed` returns a non-zero exit status if a file name argument is specified and the file does not exist. This condition can affect Makefiles and shell scripts. A non-zero exit status can cause the `make` operation or shell script to terminate.

`ed` did not previously return a non-zero exit status under any circumstances. To work around the new condition, modify the affected Makefiles and shell scripts to ignore the exit status of `ed`.

sh

The shell has been upgraded to the System V, Release 3.1 version. This version no longer uses the 8th bit of a character to quote that character, so that it can handle command names and arguments containing characters with the 8th bit set.

The new shell includes the `getopts` built-in command, which supercedes the `getopt` command. `getopt` is still provided. `getopts` is preferred, however, as it corrects several inadequacies of `getopt`.

The `getoptcvt` command can be used to convert shell scripts to use `getopts`.

NOTE `getopts` *is only supported by the System V Release 3 Bourne shell. Scripts that must also work on systems using the System V Release 2 Bourne shell, including SunOS releases prior to 4.0, should continue to use* `getopt`.

`stty`

The `stty` command is now derived from the System V, Release 3 version, to support the new terminal driver. When the terminal modes are printed, they are different from the modes printed by the 3.x `stty` command. Most of the mode settings supported by the 3.x `stty` command, such as `cbreak`, are still supported by the 4.0 `stty` command.

System V STREAMS Interface

STREAMS is a facility with a set of tools for development of UNIX system communication services from networking protocol suites to individual device drivers.

STREAMS defines standard interfaces for character I/O within the UNIX kernel, and between the kernel and the rest of the UNIX system. STREAMS consists of a set of system calls, kernel resources, and kernel utility routines. STREAMS is not limited to a specific network architecture. STREAMS offers the following major features:

- Buffer management
- Flow control
- Scheduling
- Multiplexing
- Asynchronous operations of STREAMS and user processes

There is a new STREAMS-based `tty` driver.

There is a STREAMS-based Network Interface Tap (NIT). The old provisional socket-based facility has been replaced by a set of STREAMS modules and drivers that collectively provide a superset of the old versions functionality. A major enhancement is the addition of a packet filtering module that makes selecting relevant packets out of all incoming packets much more efficient.

For information on implementing STREAMS drivers and modules, see *Part III* of *Writing Device Drivers*. For information on implementing STREAMS applications, see *Programming Utilities and Libraries*.

2.10. Lightweight Processes

The 4.0 *lightweight process library* provides primitives for manipulating threads of control, as well as for managing events (interrupts and traps). It is an excellent abstraction for implementing service processes which must maintain state for multiple connections, and for programs which manage asynchrony. At present, there is no kernel support for lightweight processes, so concurrent system calls must be implemented by forked UNIX processes.

The functions supported by the library include the following:

- Thread creation, destruction, status gathering, priority manipulation, sleeping, suspension and resumption. It is possible to implement your own scheduler as a lightweight process. For example, a high priority lwp can implement time-slicing for lower priority lwps by periodically waking up to reshuffle the lower priority lwp queues. The clock is multiplexed, so many threads can sleep concurrently for different time intervals.

- Individualized context switching (e.g., it is possible to specify that a given set of threads will touch floating point registers and only those threads will context switch these registers).
- Monitors and condition variables to synchronize threads.
- Extended rendezvous (message send-receive-reply) for communication between threads.
- An exception handling facility that provides both *notify* and *escape* exceptions.
- A way to map interrupts (asynchronous signals) into extended rendezvous.
- A way to map traps (synchronous signals) into exceptions.
- Utilities to allocate red-zone-protected stacks, and to provide some stack integrity checking for environments that lack sophisticated memory management.
- A non-blocking I/O library is available that simulates the effect of concurrent system calls by using asynchronous, non-blocking I/O.

For more information, see the *Lightweight Processes Tutorial* in *System Services Overview*.

2.11. Programming Environment Changes and Upgrades

This section covers the changes and upgrades incorporated in the programming environment in SunOS 4.0.

C Compiler Changes

The following are changes and enhancements to the C compiler:

- Has an "opaque pointer" type of `void *` which is conformable with any other pointer type. It may be assigned to or from any other pointer type without a warning.
- Does not accept "old-fashioned initialization" and "old-fashioned assignment operators" in the language.
- Treats `enum` types as integral types.
- Accepts the new `-pic` option.

The new `-pic` option instructs the compiler to generate *position-independent code* (PIC). PIC is used to improve the memory utilization performance of dynamically linked programs, such as shared libraries. Code generated as PIC uses indirect references to access global objects, such as global data or functions. These indirect references are slower as a result of the extra indirection.

NOTE *A program that makes many indirect references over a brief period of time may see a performance degradation if compiled with the* `-pic` *option.*

The C preprocessor has been upgraded to support the `#elif` control line from the proposed ANSI C standard. It now predefines `sparc` on the Sun-4. A new command option `-B` causes it to handle the C++ comment indicator "`//`". This

symbol, and everything after it on a line, is treated as a comment.

Changes to `as`

The assemblers now accept a new flag, `-k`, which informs the assembler that the source module is written in a position- independent manner. When assembling with `-k`, the assembler issues different relocation information for the link editors. More importantly, however, the assembler interprets some operating syntax differently than when `-k` is not specified. `-k` is primarily intended to assemble code produced by compilers that generate code appropriate for the new syntax interpretation.

See, *Assembly Language Reference* for more information.

Changes to Debuggers

`adb` and `dbx` have been enhanced to handle the incremental appearance of programs through dynamic linking. Both can be used with dynamically linked programs.

NOTE `dbx` *cannot be used on the dynamically linked objects themselves. This is a restriction in SunOS 4.0 and will be removed in a future release.*

Changes to `lint`

`lint` incorporates the above changes from the C compiler. The `enum` as integral type change is only enabled if the `-q` flag is specified.

The new version takes the new flags `-target=`*foo* and `-host=`*bar* where *foo* and *bar* are restricted to `sparc`, `mc68020`, `mc68010`. Both `-host` and `-target` default to the machine type that you are on. For portability help between Sun-3's and `SPARC` systems, specify `-host=mc68020` if you are running on a `SPARC` system or `-target=sparc` if you are running on a Sun-3. These flags are for specific portability between these machines and should not be used with the `-p` flag.

In addition, `lint` performs the following functions:

- Detects and flags different alignment of structure members between Sun-2 C, Sun-3 C, and `SPARC` C (when host and target are different).
- Detects and flags possible alignment problems on structure-pointer coersions. The old version assumed all structure pointers to have the same alignment.
- Treats `long` type as `int`, and `unsigned long` as `unsigned int` (if `-q` is specified).
- Treats a 0 supplied as a parameter value as being conformable with any pointer, if `-q` is specified.
- Issues a better message, when using the `-x` flag, about external declarations in `.h` files. The previous version could not figure out the file name, and printed *???* in its place.
- Allows `/*VARARGS0*/`. The old version treated this as the absence of varargs.
- The preprocessor treats the empty comment, `/**/`, just as it is treated by the C compiler.

2.12. SunView Enhancements

SunOS 4.0 incorporates substantial changes in the SunView user interface that were introduced with SunOS 3.0 and refined in subsequent 3.x releases. This section describes the enhancements made to SunView in SunOS 4.0.

Summary of New SunView Features from Release 3.4

The main features SunOS 4.0 introduces to SunView include the following:

- `cmdtool(1)` supports `vi`, `more`, `man`, `su`, and other programs that use "raw" mode and full-screen terminal mode.
- You request menu *Stay_up* in `defaultsedit(1)` and it will be set as the default. This allows you to click the right mouse button and bring up a menu. The displayed menu remains displayed until you click the right mouse button again.

Pull-right Menus are Now the Default

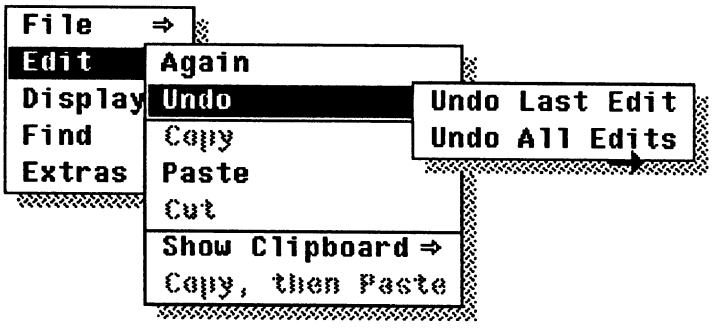
The default style is now the pull-right (walking) menu style introduced in SunOS 3.0.

You can set the default style back to stacking menus by disabling *Walking_Menus* in the *SunView* category of `defaultsedit(1)`; however, other menu defaults have no effect with old-style menus.

New Text Menu

The menu shown below in the text has been expanded and reorganized. Many tools use the Text window `textedit(1)`, `cmdtool(1)`, `mailtool(1)` and `dbxtool(1)`.

The Text menu in command windows has an additional pull-right menu, `cmd`, 'Mode ⇒' which allows you to edit the transcript and disable the scrolling to enter the `tty` Mode.



Since the default value for *SunView/Walking_Menus* has changed in 4.0, `defaultsedit` will always write out your choice for *SunView/Walking_Menus*. This forces tools that haven't been recompiled for 4.0 to pick up the default behavior.

Text Menu Layout

The new text menu is organized into several pull-right sub-menus, with "industry-standard" names; for example, 'File,' 'Edit,' 'Display,' and 'Find.' All basic editing operations such as 'Cut' are available from the new menu, as well as from function keys and keyboard accelerators. The various flavors of 'Save' are available from the 'File ⇒ Finishing Up' pull-right menu.

For more information, see *SunView 1 Beginner's Guide.*

Delimiter Matching

A new item in the 'Find' pull-right menu is 'Match Delimiter'. If you select a delimiter in the text window, and then choose this menu item, the selection will be extended to the matching delimiter.

Miscellaneous Text Enhancements

The miscellaneous text enhancements include the following:

- Word Wrap
- Find and Replace Pop-up Frame
- Field Delimiters

The *Change Line Wrap* pull-right menu lets you change the way physical lines are displayed in a text window or command window.

The *Wrap at Word* option splits lines at word boundaries when they are too long for the window. Pressing Return starts a new paragraph. This split does not change the way the file is saved. In the saved file, the text up to Return is one continuous line regardless of how it appears on the screen.

Other options in this menu are *Wrap at Character* and *Clip Lines.*

Find & Replace pop-up frames can be invoked from the Text menu. One text string can be replaced by another and you can replace the current string, the next string, or all occurrences of a designated string.

Field delimiters in text windows have the following appearance:

```
|>  <|
```

Any typing is placed between the field delimiters.

For more information, see *SunView 1 Programmer's Guide.*

'Extras' Menu

In SunOS 4.0, you can operate on the selection from a new 'Extras⇒' pull-right menu. The default 'Extras⇒' menu in `/usr/lib/.text_extras_menu` includes filters to 'Format', 'Capitalize', 'Shift Lines', 'Insert Brackets', 'Remove Brackets', and 'Pretty-print C' . These work the same way as FILTER keys in your `.textswrc` file.

You are encouraged to create your own 'Extras⇒' menu. See *SunView 1 Beginner's Guide* for information on to create this menu.

The text extras file is re-read every time you bring up the Text menu. Once you create it, you can change it while running tools. The changes are immediately displayed in the 'Extras⇒' pull-right menu.

Find and Replace Pop-Up Frame

If you select 'Find' or its first pull-right item, 'Find⇒ Find and Replace', a small pop-up frame is displayed. Type in the text string you want to find, and a text string (optional) to replace it. The following buttons are included in the 'Find and Replace' menu:

- Find
- Replace
- Find then Replace
- Replace then Find
- Replace All
- Done
- Blink Owner

For more information, see *SunView 1 Beginner's Guide.*

Keyboard Control of the Caret and Editing

There are new enhanced keyboard accelerators for caret movement and menu actions such as editing. For more information, see the "All About Accelerators" chapter in the *SunView 1 Beginner's Guide.*

Alerts

Various errors, warnings and queries now appear in pop-up alert windows. The *Alerts* package replaces a previous utility that displayed boxed error messages.

For more information, see *SunView 1 Beginner's Guide* and *SunView 1 Programmer's Guide.*

Shadowed Frames and Menus

The shadow under an alert that indicates it is a transient window also appears under pop-up frames (frames with 'Done' in their menu instead of 'Close').

All shadows are now opaque.

File Size Limit on Editing Logs

In SunOS 4.0, you can set a bound on the size of the edit log in `cmdtool` by setting the defaults entry `Text_wraparound_size` in the `Tty` category of `defaultsedit`. You can also set it using the command line option for `cmdtool -M` *maximum/minimum size.*

For more information, see *SunView 1 Programmer's Guide.*

Key Mapping

This enhancement lets you remap keys so that they perform other operations. There is no keymap unless you ask for it, by specifying a *KeymapFile* in the *Input* category of `defaultsedit`.

New `mailtool`

SunOS 4.0 has a new version of `mailtool` with many enhancements. To provide backwards capability, you can change back to the previous version of mailtool by turning off the options in the *Mail* category of `defaultsedit` and restarting `mailtool`.

The new version of `mailtool` has the following enhancements:

- A simplified control panel with fewer button items.
- Multiple composition windows each with its own reply control panel.
- Separate mail composition window for each message.
- A hierarchical folders menu with subdirectories.
- Input focus that automatically places the caret in the message composition window when composing or replying.
- Fields in outgoing messages are easier to fill.

The new `mailtool` works with a new version of the `Mail`(1) program and quickly incorporates new mail into your mail file.

`mailtool` no longer changes the selection in the header window, when you do a `mailtool` operation.

`hold` is the default in `mailtool` pop-up composition windows. Mail that is read is not automatically moved to `mbox`, but is kept in your mailbox. `hold` is not the default for `mail`.

When you read new mail, or switch to a folder, `mailtool` tells you the number of new, unread, and deleted messages.

Double-clicking on 'New Mail,' 'Done,' and 'Folder' buttons (i.e., the time-consuming operations) results in only one invocation of the selected operation. Additional attentions made before completion of the selected operation are ignored.

Specifying the font to be used in the tool via the `-font` or `-Wt` command line argument now works correctly.

If you edit the header line of a received message, the header window is updated to show the new header when you move to another message.

For more information, see *Mail and Messages: Beginner's Guide.*

8-Bit Support in `shelltool` and `cmdtool`

In conjunction with the new tty driver and Bourne shell in SunOS 4.0, `cmdtool`, `shelltool`, and the tty window package (used in `cmdtool` in 'Disable Scrolling' mode and in `shelltool`) support 8-bit characters, also known as extended character sets.

NOTE *Most of the fonts in* `/usr/lib/fonts/fixedwidthfonts` *do not have characters defined above hexadecimal* `0x7F`*; you can use* `fontedit`*(1) to add another 128 glyphs to each.*

Underlining and Inverse in `shelltool`

The tty window package used to use the 8th bit in its character memory to determine if a character was to be displayed in bold or not. The change to support 8-bit characters described above also allows the tty window code to support three graphic rendition modes:

Table 2-2 *Tty Display Modes*

Mode	*Escape Sequence*	`termcap` *Name*	*Description*
standout	`Esc[7m`	`so`	This is the same mode that the tty window supported before. It displays by inverting characters if *Tty/Standout_Mode* is enabled in `defaultsedit`.
underline	`Esc[4m`	`us`	This displays by underlining characters, if *Tty/Underline_Mode* is enabled in `defaultsedit`.
bold (extra-bright)	`Esc[1m`	`md`	This is the mode whose visual representation is controlled by the *Tty/Bold_style* setting in `defaultsedit`
all other graphic rendition display modes	`Esc[`*n*`m`	—	These display the same as bold (extra-bright)

The two new capabilities have been added to the `sun` entry in `termcap`(5).

When there was only one graphic rendition mode, the tty window displayed everything in that mode — any kind of character highlighting would show up in your chosen *Bold_style* (default inverse video). Now that there are three different modes, some things that used to display in your chosen bold style will now display inverted or underlined. In fact, "bold (extra-bright)" mode is rarely used, and this is the mode that you can change to many different styles by setting `Tty/Bold_style` in `defaultsedit`.

You can get the old behavior by setting `Tty/Inverse_mode` and `Tty/Underline_mode` to `Same_as_bold` in `defaultsedit`. Also, if you need further control over what gets displayed in the different modes, you can modify `termcap(5)`.

Frame Menu Changes

'Props' Item in the Menu

There is a 'Props' item in the frame menu, corresponding to the [Props] key (by default [L3]) on the keyboard. In applications that have a property sheet (for example, the optional `canvas_demo` program) the 'Props' menu item displays the property sheet. In other applications, 'Props' is grayed out. Unbundled applications and future tools use property sheets.

Other Name Changes

'Hide' is renamed 'Back' and 'Expose' is renamed 'Front.'

The [Left] and [Right] keys are renamed "Meta"(♦) keys.

[Get], [Put], and [Delete] are renamed [Copy], [Paste], and [Cut] respectively.

Unconstrained Move and Resize

In the Frame menu, the default (top-most menu item in the pull-right menu) for 'Move' and 'Resize' is now 'Unconstrained.'

Other SunView Changes

These changes are of interest to current SunView users.

Files Renamed

The program you run to start SunView has been renamed `sunview` (although typing `suntools` still works). The file in which you store your start-up tool positions has been renamed `.sunview`, to parallel the new program name. SunView looks for a `.suntools` file when it starts up, however, if it cannot find a `.sunview` file.

`clocktool` has been renamed to `clock`.

New SunView "Root" Menu

The default "root" menu is displayed when you bring up a menu over the background pattern has been changed. The old menu is still available on `/usr/lib/.rootmenu.old`. You can still create your own SunView menu file, `~/.rootmenu`.

SunView Changes Visible to the Programmer

Alerts

The new alert package used throughout the new SunView tools is documented in the 4.0 version of the *SunView Programmer's Guide*.

More File Descriptors

The 4.0 kernel supports more than twice as many file descriptors per process, so applications are less likely to run out of windows.

Lines in Menus

You can put lines in menus in SunOS 4.0 using the `MENU_LINE_AFTER_ITEM` attribute. This takes a value of either `MENU_HORIZONTAL_LINE` or `MENU_VERTICAL_LINE`. If you create an item with the `MENU_LINE_AFTER_ITEM` attribute, there will be a line between it and the next menu item; if you create a menu with `MENU_LINE_AFTER_ITEM`, then the entire menu has vertical or horizontal lines after items.

Props Attribute

You can use the new `FRAME_PROPS_ACTION_PROC` to specify a function to be called when the user chooses the 'Props' frame menu item, or hits the [Props] key.

Shadowed Frames

A new boolean frame attribute, `FRAME_SHADOW`, controls whether frames have shadows or not. You set this attribute at the time of creating the frame; thus it can be used in `window_create()` and `window_get()`, but not in `window_set()`.

All subframes (frames owned by another frame, with 'Done' in their menu) have shadows by default.

SunView Incompatibilities

The new SunView features such as the new text menu are a dramatic improvement over their Release 3.X counterparts. However, many customers are affected by *any* change in the SunView user interface, usually because they have screendumps and instructions in documentation that assume the old SunView "look." If you are such a customer, this section lists all the changed areas and the steps you can take to ensure visual compatibility with the past.

Table 2-3 *SunView User Interface Changes*

Change	`defaultsedit` ***Work-Around***
Walking menus are the default	Set *SunView/Walking_Menus* to *Disabled*
New text menu	Set *Compatibility/New_Text_Menu* to *Disabled*
New frame menu	Set *Compatibility/New_Frame_Menu* to *Disabled*
Alerts replace "menu prompt"	Set *Compatibility/Alerts* to *Disabled*
New keyboard accelerators	Set *Compatibility/New_keyboard_accelerators* to *Disabled*
New root menu	Set *SunView/Rootmenu_filename* to `/usr/lib/rootmenu.old`
Many new `mailtool` features	Set *Compatibility/New_Mailtool_features* to *Disabled*
New tty menu	Set *Compatibility/New_Tty_Menu* to *Disabled*
Standout and Underline Modes	Set *Tty/Standout_Mode* to *Same_as_bold* Set *Tty/Underline_Mode* to *Same_as_bold*

2.13. PROM Changes for Sun-4 Architecture

Due to Sun-4's new, RISC based architecture, the Boot PROM-based power-up self-tests are slightly different, as shown in the *Installation Notes for the Sun 4200 Board Set*, and in the *PROM User's Manual.* These differences show up only on the *CPU Board LED* display and on a dumb terminal attached to Serial Port A during a diagnostic boot-up.

Some PROM monitor commands were introduced to support the Sun-3/200 series during UNIX Release 3.2, and are also used to support Sun-4/xxx workstations. Commands such as `i`, `j` and `n` supported cache memory on Sun-3/2xx workstations and will now support Sun-4 cache memory.

Changed Commands

The `d` command now dumps the state of the processor instead of opening a CPU data register.

The `h` (help) command now provides a more extensive user interface, described in the *PROM User's Manual.*

The `r` command, which previously displayed MC68020 registers, now displays SF9010 processor registers. Optional arguments are available for displaying floating point, global or special registers. You can also specify a register number to display a particular register. These registers may be observed after an unexpected trap or after a program or the user has aborted into the monitor.

The `s` command now sets Address Space Identifiers for the SF9010 processor, rather than Function Codes for the MC68020.

The `x` command provides, through a centralized diagnostic interpreter, a new user interface to the same extended tests that appeared in Sun-3 firmware.. This `Extended Test System` provides more comprehensive tests than the power-up self-tests, yet resides in the Boot PROMs. Rather than stepping through a menu hierarchy, you may now enter multiple commands from any menu to select tests and set test options. The *PROM User's Manual* describes the new command line options.

All other PROM monitor commands remain the same for this release.

Deleted Commands

The `a` and `t` commands are not present in the Sun-4 PROM monitor.

Index

Notes