Solaris 2.3 New Features

for Early Access

alpha draft

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 801-5335-01
Revision A, May 1993

SunSoft
A Sun Microsystems, Inc. Business

Contents

Preface

This book of *Solaris 2.3 New Features* is organized according to functional categories, with separate chapters for tasks performed by different users, such as system administrators and developers. These categories are only for convenience of presentation, and are not really hard and fast. Most of these features have implications in categories other than the one in which they are presented.

Who Should Use This Book

This book is for independent software vendors and other developers who are begin provided with Early Access to new version of Sun hardware and of Solaris. The information is presented “as is,” as a convenience to the reader, and does not represent any warranty, express or implied, about the suitability of Solaris 2.3 for any purpose.

How This Book Is Organized

Chapter 1, “Overview,” discusses the features which are omitted rather than added to this release of Solaris, and acts as a pointer to other relevant documentation.

Chapter 2, “Kernel Features,” explains the new Solaris features that are best explained as aspects of the SunOS kernel.

Chapter 3, "System Administration Features," summarizes the new Solaris features that are primarily of interest to system administrators.

Chapter 4, "Network Administration," describes the new Solaris features that are relevant to ethernet, TCP/IP and NFS network administration.

Chapter 5, "Programming Features," describes the new Solaris features that are of interest to application developers.

Chapter 6, "Open Windows Version 3.3 Server," describes the major changes in the window server.

What Typographic Changes and Symbols Mean

The following table describes the type changes and symbols used in this book.

Table P-1 Typographic Conventions

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

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

`%`	UNIX C shell prompt	`system%`
`$`	UNIX Bourne and Korn shell prompt	`system$`
`#`	Superuser prompt, all shells	`system#`

Overview 1

The changes to Solaris for 2.3 include improvements in all of these categories:

- Kernel
- System Administration
- Network Administration
- Programmer Interfaces for Developers
- OpenWindows Server

This manual is designed to give integrators and developers an early view of the new Solaris features, so that they can take maximum advantage of these features in product development and planning.

Changes to the OpenWindows Server

Major enhancements to the OpenWindows server are included in this release. The new version (now called OpenWindows Version 3.3) includes support for Level II Display Postscript, and for many of the features of X11 Release 5. These enhancements are summarized in Chapter 6, “Open Windows Version 3.3 Server.”

The version number of this release has recently been changed from 4 to 3.3. The two numbers refer to the exact same release of OpenWindows.

XGL and XIL Library New Features

The XIL library is the Solaris software's foundation imaging library. New XIL library features for this release are:

- CellB Compression/Decompression
- MPEG I Decompression
- H.261 Decompression

New XIL and XGL features are more fully documented in separate publications included with this Early Access release.

Online: Backup 2.0

This utility package allows Solaris administrators to configure a backup and restore scheme for system and user files that requires minimal operator intervention. An overview of Online: Backup is given in the section "Online: Backup 2.1" in Chapter 3, "System Administration Features" in this manual. Complete instructions for using this package are given in the following publication:

- *Online: Backup 2.0 Getting Started*
  (part number 801-2456-10)

- *Online: Backup 2.1 Administration Guide*
  (part number 801-8339-10)

Deleted Features

NeWS and TNT

Support for the NeWS windowing system and the TNT application programming interface are discontinued as of Release 2.3 of Solaris. Developers who are currently writing applications in Xview are encouraged to continue this process, since transition tools to future windowing systems are being developed by SunSoft. Similarly, developers who are currently writing applications using the Xt intrinsics and OLIT toolkit will be provided transition tools after Release 2.3 of Solaris.

RFS

The Remote File Sharing (RFS) system has been removed from Solaris 2.3. One implication of this is that `libns` is no longer available to developers.

Kernel Features 2

This chapter surveys the new OS kernel features of Solaris 2.3.

Several kernel-related features are covered in other chapters of this manual, such as Chapter 4, "Network Administration Features," and Chapter 5, "Programming Features." This is because the features also have implications for those topic areas.

Features Covered under Network Administration

Refer to Chapter 4, "Network Administration Features" for descriptions of new features in the following areas:

- PPP
- AutoFS
- Cache File System

Covered under Developer Features

Refer to Chapter 5, "Programming Features" for descriptions of new features in the following areas:

- MultiThread Safe Libraries
- QE Driver
- Other Streams Drivers

POSIX 1003.4 Realtime Extensions

The POSIX 1003.4 draft standard proposes realtime extensions to the POSIX 1003.1 standard. The extensions are in these areas:

- signals
- asynchronous I/O
- synchronization objects
- memory mapping
- scheduling
- timers
- message passing

These realtime extensions are planned for Solaris 2.3 in the beta timeframe, pending approval of the draft standard. They are not part of the Early Access release.

System Administration Features 3

This chapter surveys the new system administration features of Solaris 2.3.

Online: Backup 2.1

This utility package allows Solaris administrators to configure a backup and restore scheme for system and user files that requires minimal operator intervention.

Highlights of the Online: Backup 2.1 software package include:

- Online backups
- Local or remote backups
- Enhanced versions of dump and restore utilities
- Online database of backed-up files
- A configuration and execution system for setting up and running backups
- A recover program that selects and retrieves files from the online database
- Automated configuration and execution of a site-wide backup strategy
- A network-based operator monitor allowing you to run, track, and interact with Online: Backup 2.1 from any networked host
- Stacker support for the Exabyte EXB-10i device in sequential (stacker) mode

To use the Online: Backup 2.1 package, you must install it not only on your own workstation, but on every machine with file systems to back up.

Note – The Online: Backup 2.1 package is now bundled with this release of Solaris. The installation instructions in the earlier document set for Online: Backup 2.0 no longer apply.

Pluggable Authentication Model (PAM)

The main objective of the Pluggable Authentication Model (PAM) is to provide a consistent framework to allow access control applications (such as `login`) the flexibility of choosing any authentication scheme available on a system, without concern for the implementation details of the scheme. (Access control applications are those that perform account identification and authentication.) The PAM framework provides standard function calls for access control applications to access different authentication schemes; thus it helps to avoid the current inconsistency in doing authentication across access control applications. The framework also has a common API (application programming interface) available for a new authentication to be plugged in to the framework so that access control applications can use the new authentication scheme without modifying the applications.

PAM provides a foundation to deal with some of the inconsistent ways that the various access control applications (such as `login`, `rlogin`, `telnet`, etc.) handle authentication presently.

In summary, PAM

- Allows system administrators to easily choose the authentication scheme or schemes that best meet the needs of their specific environments.
- Allows third party vendors who develop and market various kinds of authentication software or hardware devices to easily fit their products into the framework.
- Allows a consistent interface to be used by all access control applications. This avoids inconsistent or conflicting styles of authentication, which cause customer confusion and difficulty in maintenance and support.

Motivation to Develop with PAM

Traditionally, UNIX has a very simplistic and restrictive authentication mechanism, based on the password model. It imposes the same mechanism regardless of how sensitive the user environment is to system security. It doesn't care whether the system will be used in a bank (where security will be of paramount importance) or in an engineering department (where there is a fairly open environment and a minimum level of security is acceptable).

This lack of flexibility is directly attributed to the root of UNIX, which was designed for a very specific kind of environment. However, as UNIX gains popularity in a diverse variety of commercial environments, the need arises for a more general and powerful authentication scheme that allows users (system programmers or administrators) to easily modify or add to the existing mechanism. This need has been made even more acute as many companies have begun to manufacture various kinds of authentication devices, such as optical/magnetic access cards and, for even more sensitive environments, voice and fingerprint recognition devices.

As UNIX becomes the de facto standard operating system, interest to adopt it into currently non-UNIX environments is very strong; and the capability for UNIX to support the variety of authentication devices currently supported on other systems becomes a requirement.

Another problem with traditional UNIX implementation is that there are many access control applications, each adopting its own way of performing authentication. Among these are `telnet`, `rlogin`, `rex`, `rexec`, `ftp`, and others. Some of them perform authentication by executing the `login` program; others copy code from `login` and do their own password decryption; still others find authentication too much trouble and ignore it altogether. This is largely due to the fact that there is not a framework for doing authentication consistently. Without it, the current confusion and inconsistency will only grow worse as more access control applications are added to the system.

The main objective of the PAM framework is to solve the above two problems. The framework is designed so that

- Users (system programmers/administrators) can easily add support for extended authentication mechanisms they need in their environment, without access to system source code (such as the source for `login`).
- All the access control applications that require (or should require) authentication can use the services provided by the framework to ensure consistency. Existing applications should be converted to use this framework; future applications can easily take advantage of it for their authentication needs.

C2 Security

The C2 Security module is an operating system enhancement package that provides the security features specified in the C2 Class requirements of the Trusted Computer Security Evaluation Criteria (TCSEC). As of the upcoming Solaris 2.3 release, the C2 Security module is bundled with Solaris.

The US Government has an oft-publicized mandate known generically as "C2 in '92." The actual policy statement is contained in NTISSP #200 (National Policy on Controlled Access Protection) dated 15 July 1987. The policy is defined to apply to all automated information systems operated by Executive-branch agencies and departments of the federal government and their contractors.

That policy stipulated a five year term of compliance, and Controlled Access Protection is defined as the C2 level of protection, hence "C2 in '92."

The IEEE POSIX 1003.6 draft standard has also been taken into consideration in the C2 design so that the POSIX 1003.6 interfaces can be easily layered over the C2 interfaces. The evolving NIST Minimum Security Functionality Requirements (MSFR) draft, and the X/Open Security Guide were also taken into consideration in the C2 design.

Technical Description

The features required to comply with C2 are generically defined in the TCSEC. They consist of enhancements to the identification and authentication (I&A) mechanisms, thorough enforcement of object reuse policy, and inclusion of a complete security auditing capability.

More details of the TCSEC C2 functionality requirements are as follows.

Discretionary Access Control

The terminal control block (TCB) shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

Object Reuse

All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Identification and Authentication

The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual adp system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

Audit

The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initiation), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security-relevant events.

For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

New Solaris Functionality for C2 Compliance

The primary new functionality is the security auditing subsystem. It consists of a kernel-based audit queuing and storage mechanism, a set of kernel system call event generation modules, a set of user-level event generation modules, audit administrative utilities and databases, and audit analysis tools.

An additional piece of functionality that's added to the system is a mechanism that allows for the handling of allocatable devices (such as tape drives and floppy disks) such that they can be used in a manner compliant with the TCSEC C2 object reuse requirements.

C2 Reference Documents

- Trusted Computer Systems Evaluation Criteria
  TCSEC, dated December 1986
- National Policy on Controlled Access Protection
  NTISSP #200, dated 15 July 1987
- Security Requirements for Automated Information Systems (AISs)
  DoD Directive 5200.28, dated 3/21/88
- SunOS 4.1.1 C2 External Specification
  SunFed dm0047, Revision 2.3

format(1) Changes for SCSI Disks

This new feature represents a set of changes to the `format(1)` program, and changes to Sun-supplied disks, which permit format to configure and label SCSI disks, without requiring disk definitions in format.dat.

SCSI disks can provide enough geometry and configuration information to permit the `format(1)` command to configure and label disks automatically. Prior to this new features, all disks needed to be defined in

/etc/format.dat, causing two major obstacles: the release of a new disk needed to be tied to an OS release; and customers wishing to use a third-party disk needed to somehow obtain a format.dat definition for that disk.

The primary goal of the auto-configure process is to make picking the disk type and labeling a SCSI disk an invisible step to the user. This means that when format sees an unlabeled disk, it will attempt the auto-configure process. By the time the main format menu is entered with the list of available disks, an unidentified disk would only be a result of failure of the auto-configure process (with one exception, noted below).

For each unlabeled disk, format constructs an internal label, and notes that it is "dirty". When the disk is selected, format asks "Do you want to label the disk?." Note that making any permanent change in the state of the disk specifically requires user intervention.

The auto-configure process can also be entered by user command, allowing the user to fix a problem, if the disk was previously labeled incorrectly.

The exception: when initially auto-configuring a disk with a sector size of other than 512, the initial configure process fails. Otherwise, we'd be forced to ask if the user wants to reformat that disk, and this would not be the appropriate time. Once the disk is selected, however, we can query the user and reformat and auto-configure the disk at that time.

A specific goal of this project is that, on a system with only SCSI disks, the format.dat file will no longer be required for format to work correctly. This does not imply that we are getting rid of format.dat, but that it is no longer strictly necessary for format itself. Format.dat will still be required on systems with disk types other than SCSI, and on systems with Sun disks which require format.dat entries. Prior to this new feature, all Sun disks did require format.dat entries, so the existence of format.dat was always required.

The format.dat Override

The first step of the format(1) auto-configure process is to check for an appropriate format.dat entry. This provides a manual override, to correct an error in auto-configuration via standard SCSI-2 mode pages.

The mechanics of this are: bytes 25-31 (the last 7 bytes of the Product Identification field) of the Inquiry data yield a string of 7 characters. If a disk definition in format.dat matches this name, use that definition.

Sun Disks

All current Sun disks still use disk definition entries in `format.dat`, matched by the Inquiry product name. This means the result of running the auto-configure `format` on any current Sun disk product will not be any different from the earlier `format` program.

Future Sun disk products will continue to work with `format`, either through new `format.dat` entries or by auto-configuration via standard SCSI-2 mode pages.

Third-Party Disks

For disks for which no matching `format.dat` entry can be found, `format` will ask for the standard SCSI-2 mode sense pages for the necessary configuration information. For this method to be successful, the disk must implement at least some minimum of the SCSI-2 mode page set, with consistent values.Any irregularities will cause the process to fail, for which Sun cannot be responsible. The vendor should rework the product to make the disk SCSI-2 compliant.

The specific values and mode sense pages used for each value are:

pcyl:page 4, bytes 2-4 (physical cylinders)

nhead: page 4, byte 5 (number of heads)

nsect: page 3, bytes 10-11 (sectors per track)

rpm: page 4, bytes 20-21 (medium rotation rate)

The following parameters are derived from the values determined in the mode sense pages:

acyl: 2 (fixed for format's requirements)

ncyl: pcyl - acyl (data cylinders)

Some disks populate their geometry sense pages with values related directly to the hardware capacity of the drives, not allowing for whatever portion of the disk may be reserved for the disk's private use. With such disks, the sensed geometry will be larger than that actually usable, and `format` will use the largest possible value of **pcyl** which permits "**pcyl** * **nhead** * **nsect**" to be less than or equal to the number of blocks available on the disk, as reported by the Read Capacity command.

Sector size is another possible complication. Sun's current OS and drivers cannot work with devices formatted with sector sizes other than 512 (DEV_BSIZE). Before auto-configuring, if the sector size of the device, as reported in page 4, bytes 12-13, is not equal to 512, `format` asks the user if s/he would like to change the sector size. If yes, `format` issues a mode select on page 4, bytes 12-13, then format the device with the P&G lists. After formatting, the geometry and capacity of the device may have changed, so `format` must reissue new commands to get the device's new state.

Once the disk's usable size is determined, `format` constructs a default partition table for the disk. This default partition table is then used to complete the disk label. Refer to the next section information about the partitioning process.

Default Partition Construction

When creating a label for an unlabeled disk, `format` must create a partition table for that disk. With earlier versions of the OS and installation, this would have been a more difficult process, but today's OS makes this much simpler. The OS's only requirement at installation time is that the root partition be at least 16 megabytes.

With this in mind, we have hardwired a set of partitioning rules into `format`. Should these rules change in a subsequent OS, it would be a simple matter of a minor change to `format`.

In all cases, partition 2 is the entire usable area of the disk, with respect to **ncyl**. The proposed rules for the other partitions are:

Disk Size	Root (0)	Swap (1)	Usr
<180mb	16mb	16mb	<remainder>
180-280mb	16mb	32mb	<remainder>
280-380mb	24mb	32mb	<remainder>
380-600mb	32mb	32mb	<remainder>
600-1000mb	32mb	64mb	<remainder>
1gb-2gb	64mb	128mb	<remainder>
>2gb	128mb	128mb	<remainder>

Diagnosing Problems

We expect that, as these features come into general use, there will be problems of various sorts: certainly with disks that do not report one or more mode sense values correctly, and probably with `format` itself. `format` at present has command-line switches to enable several levels of debugging and diagnostics. This debugging output can provide a large amount of information why `format` may be having problems formatting or configuring a particular disk. With these new auto-configure features, `format` maintains the same level of diagnostic information, and should provide enough information for a person knowledgeable about SCSI to be able to determine the cause of most problems, without requiring format source or a SCSI analyzer.

With two possible paths through the auto-configuration process, it becomes necessary for persons doing disk qualification and format testing to be able to exercise more control over the auto-configuration process. We have extended the meaning of the "scsi expert" command line argument -e. When auto-configuration is invoked with the "scsi expert" option in effect, the user is given the option of forcing one of the two possible paths:

> Auto configuration via `format.dat`[no]?

> Auto configuration via generic SCSI-2 pages[no]?

Each prompt has a default response of "no" if the user enters Return, and if no option is forced, auto-configuration proceeds as usual.

Serial Port Manager (SPM)

The Serial Port Manager, an `admintool` application, configures the serial port software to work with terminals and modems. It features:

- an OPEN LOOK graphical user interface
- the functionality of the Service Access Facility's `pmadm` command

When to Use the Serial Port Manager

Use the Serial Port Manager when you want to add a terminal or modem to a system. Serial Port Manager provides templates for the most common terminal and modem configurations that you can then modify for a particular device.

Terminal - Hardwired
Modem - Dial-in Only
Modem - Dial-out Only
Modem - Bidirectional
Initialize Only - No Connection

The system being configured may be your local system or a remote system. Because the Serial Port Manager needs to modify system information, you must have special privileges for that system.

Table 3-1 User privileges needed to run Administration Tool

For a...	you must...
local system	• run Administration Tool as root, or • be a member of the UNIX sysadmin group (GID 14) for that system
remote system	be a member of the UNIX sysadmin group (GID 14) for that system

How to Start the Serial Port Manager

The Serial Port Manager is started from the Administration Tool window. If you don't have Administration Tool running, start it by typing:

```
/usr/bin/admintool &
```

To start the Serial Port Manager, click on the Serial Port Manager icon in the Administration Tool window.

What you can do with the Serial Port Manager

With the Serial Port Manager, you can

- Initialize a port without configuring the service
- Add a service
- Modify a service
- Disable a service
- Delete a service

Network Administration Features 4

This chapter surveys the new network administration features of Solaris 2.3. These features fall under the following four categories:

- AutoFs
- PPP
- Cache File System for NFS
- NIS+

AutoFs

autoFs is a new kernel virtual file system that supports automatic mounting of file systems. Together with a re-implementation of `automountd` (the automount daemon), autoFs fixes several fundamental problems with the prior implementation of `automountd`: the troublesome symbolic links and the `/tmp/mnt` prepended to paths are replaced by in-place mounting.

Without symbolic links, indirection to mountpoints is now performed entirely within the kernel, improving performance. `automountd` is now a stateless name resolution daemon, allowing mountpoints to be added or deleted without rebooting. The daemon is not required to access a filesystem once it is mounted.

Safer unmounting behavior is now possible. Under the prior release of Solaris, the automounter could unmount a hierarchy of mounts, making a needed file system inaccessible to a process that is working within the hierarchy.

In addition, automountd now can mount other types of filesystems besides NFS, such as removable media filesystems. These improvements are backward compatible with existing maps and administrative procedures.

Simplified Automounter Operation

The automount daemon automountd starts at boot time from the /etc/init.d/autofs script. This script also runs the automount command, which reads a master map and installs autoFs mount points.

Unlike mount, automount does not read the file /etc/vfstab, which is specific to each machine, for a list of file systems to mount. Rather, automount is controlled within a domain (and on particular workstations) through the maps, saving a great deal of administrator time.

How the Automounter Navigates through the Network (Maps)

The automounter searches a series of maps to navigate its way through the network. Maps are files that contain information such as the password entries of all users on a network, or the names of all host machines on a network, i. e., network-wide equivalents of Unix administration files. Maps are available locally or through a network name service like NIS or NIS+. The administrator creates maps to meet the needs of your environment using Admin Tool.

There are three kinds of automounter maps:

- master
- indirect
- direct

How the Automounter Starts the Navigation Process (The Master Map)

The automounter reads the master map at system startup. Each entry in the master map is a direct or indirect map name, its path, and its mount options. The specific order of the entries is not important. The automounter compares entries in the master map with entries in the mount table.

Master Map Syntax

Each line in the master map /etc/auto_master, has the syntax:

```
mount-point   map-name    [ mount-options ]
```

mount-point

mount-point is the full (absolute) path name of a directory. If the directory does not exist, the automounter creates it if possible. If the directory exists, and is not empty, mounting on it hides its contents. In this case, the automounter issues a warning message.

map-name

map-name is the map the automounter uses to find directions to locations, or mount information. If the name is preceded with a slash (/), the automounter interprets the slash as a local file. Otherwise, the automounter searches for the mount information using a name service switch policy.

[mount-options]

mount-options is an optional, comma separated list of options that apply to the mounting of the entries specified in *map-name*, unless the entries in *map-name* list other options. The *mount-options* are the same as those for a standard NFS `mount`, except for `bg` (background) and `fg` (foreground). These do not apply.

A line whose first character is **#** is a comment. Everything that follows until the end of the line is ignored.

A backslash (\ ) at the end of the line permits splitting long lines into shorter ones.

The notation `/-` as a mount point indicates that the map in question is a direct map, and no particular mount point is associated with the map as a whole.

How the Automounter Finds Specific File Systems (Indirect Maps)

An indirect map is a directory containing automount points. Indirect maps are useful for accessing specific file systems, like the home directories.

How the Automounter Finds Reference-Oriented File Systems (Direct Maps)

A direct map is an automount point. It is used for accessing reference or application-oriented file systems, such as the man pages, or FrameMaker®. With a direct map, there is a direct association between a mount point on the client and a directory on the server. Of all the maps, the entries in a direct map most closely resemble, in their simplest form, the corresponding entries in `/etc/vfstab` (`vfstab` contains a list of all file systems to be mounted). An entry that appears in `/etc/vfstab` as:

```
dancer:/usr/local - /usr/local/tmp nfs - yes ro
```

appears in a direct map as:

```
/usr/local/tmp      -ro      dancer:/usr/local
```

Special Mount Points

The automounter recognizes some special mount points and maps, which are explained below.

Mount point /–

In the example above, the mount point `/-` acts as a place holder in a direct `map` that the automounter recognizes as a directive not to associate the entries in `auto_direct` (any direct map) with any directory. Indirect maps use mount points. Direct maps do not. Instead, the mount points are those specified in the auto_direct map. (Remember, in a direct map the key, or mount point, is a full path name.)

An NIS or NIS+ `auto_master` can have only one direct map entry. An auto_master that is a local file can have any number of entries.

Mount point `/home`

The mount point `/home` is the directory under which the entries listed in `/etc/auto_home` (an indirect map) are to be mounted.

Mount point `/net`

The automounter mounts under the directory `/net` all the entries in the special map `-hosts`. This is a built-in map that uses only the `hosts` database. For example, if the machine `gumbo` is in the `hosts` database, and it exports any of its file systems, a user entering the command:

```
cd /net/gumbo
```

will change directory to the root directory of the machine `gumbo`. However, the user may not see under `/net/gumbo` all the files and directories. This is because the automounter can mount only the *exported* file systems of host `gumbo`. Exported file systems are those file systems on a server available to network users as opposed to having file systems on your local disk.

Note that the automounter has to mount all the file systems that the server in question exports. Even if the request is as follows,

```
ls /net/gumbo/usr/include
```

the automounter mounts all of `gumbo`'s exported systems, not just `/usr`.

The unmounting that occurs after a certain amount of time has passed is from the bottom up (reverse order of mounting). This means if one of the directories at the top is busy, the automounter has to remount the unmounted file system(s) and try again later.

The `-hosts` special map provides a very convenient way for users to access directories in many different hosts without having to use `rlogin` or `rsh`. They no longer have to modify their `/etc/vfstab` files or mount the directories by hand as superuser. With the /net method of access, the server name is in the path and is location-dependent. If you want data relocated from one server to another, the path may no longer work. Also, since all exported file systems need to be mounted it is inefficient if only one filesystem is used. It's better to set up an entry in a map specifically for the file system you want.

Note – The automounter runs on all machines and supports /net and /home (automounted home directories) by default. These defaults may be overridden by entries in the NIS auto.master map or NIS+ auto_master map or by local editing of the /etc/auto_master and /etc/auto_home file.

New Documentation for AutoFS

The chapter entitled "Using the Automounter" in the SunSoft publication *Administering NFS* has been considerably expanded to reflect AutoFS functionality. This revised publication will be forthcoming with the Solaris 2.3 release.

PPP

PPP is a network protocol product that provides IP network connectivity over a variety of point-to-point connections. The kinds of connections that are included are:

- synchronous serial ports
- asynchronous serial ports
- modems
- ISDN (future release of Solaris)

As such, PPP is an alternative to the SLIP protocol, which also offers IP services over modems.

The PPP product has four major components:

- PPP-Kernel modules
  These provide the basic protocol support to run PPP on all the above listed connections.
- PPP-Async
  The user-level part of asynchronous PPP, including an Asynchronous Link Manager (two executables), default configuration files and (forthcoming) a graphic user interface for editing the configuration files. In the initial release, administration will be done with manual editing of the configuration files. Dialup configuration will be done through editing the files in /etc/uucp.

- PPP-Sync
  The PPP Synchronous User Interface and Link Manager (forthcoming SunConnect product).
- Administration information
  Includes man pages for the kernel modules, and (forthcoming) a document called *Asynchronous PPP System Administrator's Guide*, and an updated version of the document *SunOS 5.0 Administering TCP/IP and UUCP*.

PPP Performance Goals

The following are the performance goals for the PPP product:

- Able to operate two T1 lines at full speed using PPP-Sync. This is an approximate aggregate bandwidth of 3 MBit/sec.
- There should be no more than 20 milliseconds delay for a round trip on a PPP-Sync link on a line operating above 64 Kbaud.
- PPP-Async should be able to support both dial out and dial in, up to eight PPP dialup connections at 19.2K baud or faster, using one or two SPC/S (SPIF) 8-port asynchronous cards.
- There should be no more than 130 milliseconds delay for a round trip of an 8 bit packet on a PPP-Async link on a line operating at 19.2 Kbaud. Bulk file transfers (such as FTP) should offer at least 95% of the throughput of the underlying device driver.

RFC Compliance

The PPP product shall successfully implement RFC 1171 and 1172. New RFCs (1331, 1332 and 1333) currently proposed standards to supercede 1171 and 1172. Compliance with these proposed standards is optional.

PPP-Kernel Modules

The new kernel modules to support PPP consist of:

- PPP Streams module
- ipd device driver
- ipdcm device driver
- ipdptp device driver

The PPP-Kernel modules provide the following services:

- Implement both Sync and Async PPP encapsulation.
- Async character mapping
- Protocol and address field compression
- MRU negotiation
- Password Authentication Protocol
- Van Jacobson TCP/IP header compression

Additionally, the PPP-Kernel modules support the following device drivers:

- HSI/S
- HSI
- ZSH
- ZS (async)
- SPC (async)

Cache File System for NFS

The Cache File System (CacheFS) is a generic non-volatile caching mechanism to improve performance of certain file systems by utilizing a small, fast, local disk. The read-write capability of a local disk can also be used to fake updates to file systems on read-only media such as CDROM.

The primary applications targeted for CacheFS are NFS (to improve server scalability) and HSFS (to improve performance and add update capability). In theory, CacheFS can be useful in any situation where the local disk provides better performance than the file system being cached.

CacheFS can also be used for installation of SunOS and applications via "boot and execute from CDROM."

Note that the ease of use of the CacheFS is enhanced considerably by AutoFS. (Refer to "AutoFs" on page 19.) And without AutoFS, automounted NFS filesystems are not cache-able.

Background

The VNODE/VFS (virtual file system) interface has enabled many different kinds of file systems to be implemented and used under SunOS. Several of these file systems are limited by performance and lack of functionality. CacheFS utilizes a fast local disk to enhance performance and extend the functionality of these file systems in a transparent manner.

New NIS+ Setup Scripts

This release of NIS+ includes three new scripts. These scripts enable you to set up an NIS+ domain much more quickly and easier than if you used the individual NIS+ commands to do so. With these scripts you can avoid a lengthy manual process. Table 4-1 describes the scripts. See the `nisserver`(1M), `nispopulate`(1M), and `nisclient`(1M) man pages for a list of their options and how to use them. See the *NIS+ Quickstart* (to be released with beta) for a detailed explanation of when and how to use them. A glossary has also been

Table 4-1 The NIS+ scripts

NIS+ script	What it Does
`nisserver`(1M)	sets up the root master, non-root master and replica servers with level 2 security (DES)
`nispopulate`(1M)	populates NIS+ tables in a specified domain from their corresponding system files or NIS maps
`nisclient`(1M)	creates NIS+ credentials for hosts and users; initializes NIS+ hosts and users; restores the network service environment

added to the NIS+ books.

Programming Features 5

This chapter surveys the new application, network and system programming features of Solaris 2.3.

Drivers for New Hardware Interfaces

New drivers have been developed for several interface cards. Most prominent of these interfaces are SPC/S and Sport16D/SE.

ISP Host Adapter Driver

The ISP host adapter driver is a Solaris 2.x Sun Common SCSI Architecture (SCSA) driver that uses the existing Solaris 2.x SCSA and DDI/DKI interfaces to operate over the Sport16D/SE SCSI Host Adapter (SBus) card. There are no new nor modified software interfaces.

The Sport16D/SE SCSI Host Adapter card allows a maximum of eight target devices and data transfer rates of 10MB per second, with one or fewer interrupts per I/O cycle. This meets the requirements of higher-end servers and desktop workstations that require more disks and less CPU overhead.

The ISP driver consists of C code and header files, and is documented in the `isp(7)` man page.

SPC/S Driver

This driver is documented in the man page `stc(7)`.

Extensions to sysconf(3C)

The application programming interface to `sysconf()` has been extended to permit reporting of available system memory. Two new arguments have been added:

_SC_PHYS_PAGES	number of installed physical pages of memory
_SC_AVPHYS_PAGES	number of available physical pages of memory, (pages that are not currently in use)

This new interface should be used instead of `sysmem()` and `asysmem()`, which are part of the `libadm` library. `sysmem()` and `asysmem()` return the installed and available physical memory of a system in bytes. They return a signed long, which is insufficient for systems with more than 2G of memory, such as SPARCcenter 2000.

Caution: Changes to fopen()

In the Solaris 2.3 release, the acceptable "mode" arguments to `fopen()` have been somewhat tightened to conform to the behavior specified in the man page. Because of dynamic libraries, this creates a situation where existing binaries might break. If a application failure occurs when running with the new dynamic libraries, this is a possible explanation. The error messages in such a case may be misleading, and probably will not seem to point to `fopen()`.

MultiThread Safe Libraries

In order to facilitate the use of the user-level threading capability within Solaris, the basic function libraries within Solaris have been made MT safe. That is, they have been modified to allow their use by multiple threads. The libraries affected are:

- Core Networking
- libc
- Xlib

The changes to these libraries are described in the following sections.

Revised documentation for these features will also be provided. Refer to "MultiThread Safe Documentation" on page 36.

All changes shall be compatible with the SVID ABI.

Affected Files

The following library files are affected by these changes:

- lib/libnsl (listed components only)
  - NIS+
  - RPC and XDR
  - netdir
  - netselect
- lib/libsocket
- lib/libc (get X by Y interfaces only)
- lib/nametoaddr
- lib/nsswitch
- lib/libdl_stubs (for support of static switch compilation)
- lib/libresolv (support for thread-specific errno only)

Additional file affected by these changes:

- cmd/rpcgen
- header files (as required)

Additional RPC Key Requirements

The following Remote Procedure Call requirements are satisfied by the MT safe libraries:

- Multiple threads can share client handles in an MT safe way
- An RPC client thread can send requests to an RPC service dispatch loop thread in the same process

Core Networking Libraries

These new features provide MT safe implementations of core networking libraries, which include:

- libsocket
- Name to Address Translation
- Portions of libnsl
- Name Service Switch components in several libraries.

MultiThread Safe libc

The goal of this new feature is to make libc MT safe, which incidently is a requirement to be SVID compliant. This basically involves making changes to the libraries so that several threads can execute without race conditions.

Solaris 2.2 contained a single libc library (both static and dynamic) that worked for both existing single-threaded and multi-threaded applications. For Solaris 2.3, the interfaces in libc have now been made MT safe except for a few that have new reentrant counterparts.

Interfaces Made MT Safe

- getdate()
- cuserid()
- readdir()
- getgrent()
- getgrgid()
- getgrnam()
- fgetgrent()
- getutent()
- getutid()
- getutline()
- getutxent()
- getutxid()
- getutxline()
- localtime()
- gmtime()
- asctime()
- ctime()
- getlogin()
- getpass()
- l64a()
- ttyname()
- crypt()

- encrypt()
- setkey()
- getopt()
- getpwent()
- getpwnam()
- getpwuid()
- fgetpwent()
- getspent()
- getspnam()
- fgetspent()
- strtok()
- ecvt()
- fcvt()
- qecvt()
- qfcvt()
- rand()
- tmpnam()
- ptsname()
- setlocale()
- nl_langinfo()
- localeconv()
- getvfsent()
- getmntent()

New Reentrant Interfaces

- getdate()
- cuserid_r()
- ctermid_r()
- readdir_r()
- getgrent_r()
- getgrgid_r()
- getgrnam_r()
- fgetgrent_r()
- getutent_r()
- getutid_r()
- getutline_r()
- getutxent_r()
- getutxid_r()
- getutxline_r()

- localtime_r()
- gmtime_r()
- asctime_r()
- ctime_r()
- getlogin_r()
- getpass_r()
- l64a_r()
- ttyname_r()
- crypt_r()
- encrypt_r()
- setkey_r()
- getopt_r()
- getpwent_r()
- getpwnam_r()
- getpwuid_r()
- fgetpwent_r()
- getspent_r()
- getspnam_r()
- fgetspent_r()
- strtok_r()
- ecvt_r()
- fcvt_r()
- qecvt_r()
- qfcvt_r()
- rand_r()
- tmpnam_r()
- ptsname_r()
- setlocale_r()
- nl_langinfo_r()
- localeconv_r()
- getvfsent_r()
- getmntent_r()

Unlocked functions (for stdio only)

- getc_unlocked()
- putc_unlocked()
- getchar_unlocked()
- putchar_unlocked()
- ungetc_unlocked()

- rewind_unlocked()
- feof_unlocked()
- ferror_unlocked()
- fileno_unlocked()
- clearerr_unlocked()

New Interfaces

- flockfile()
- funlockfile()
- ftrylockfile()

MultiThread Safe libw

In the process of making `libc` MT safe, changes were also needed in `libw`.

`errno` is now a macro expanded to `*_thr_errno_addr())`.

All the interfaces in `libw` were safe except for the `wstok()` function, which has a new reentrant counterpart, `wstok_r()`.

MultiThread Safe libintl

In the process of making `libc` MT safe, changes were also needed in `libintl`.

Interfaces in `libintl` are safe except for the following functions, which now have been replaced with new reentrant counterparts:

- `textdomain()`
- `bindtextdomain()`
- `dgettext()`
- `gettext()`

The new reentrant counterparts are:

- `textdomain_r()`
- `bindtextdomain_r()`
- `dgettext_r()`
- `gettext_r()`

Other Libraries made MultiThread Safe

`libmapmalloc` for Solaris 2.3 consists of a single library (both static and dynamic) that works with both single-threaded and multi-threaded applications. No new _r functions were needed.

`libmalloc` for Solaris 2.3 continues to be only a single library (static). No new _r functions were needed.

For `libcrypt` the plans are to ship a single static library. No plans are available yet for `libgen`, `libcmd`, `libcurses`, `libeti` and `libplot`.

MultiThread Safe Documentation

Changes to the documentation are forthcoming with the Beta and final releases of Solaris 2.3. A summary of these changes is:

- Updates to existing man pages for some functions
- New man pages for new reentrant and locking functions
- New chapters in the *Guide to Multi-Thread Programming*:
  - Chapter 1: Introduction to Multi-thread
  - Chapter 2: Using Threads
  - Chapter 7: The Thread Environment
  - Chapter 10: Conversion Strategies

multisource Safe Xlib

Changes were made to the functions in `Xlib` to make them MultiThread safe. No changes to the API are apparent to the developer.

OpenWindows Version 3.3 Server 6

OpenWindows Version 3.3 includes major changes in the window server. In particular, MIT's X11R5 window server and Adobe's Display PostScript™ (DPS) imaging extension replace the X11/NeWS™ technology. The Version 3.3 window server also includes many of the R5 extensions.

All other components of the OpenWindows environment are largely unchanged from Version 3.2, except that it *does not* support NeWS and The NeWS Toolkit (TNT).

The OpenWindows server includes *all* features of the X11R5 sample server, plus the DPS imaging extension and SunSoft enhancements. Books available through SunExpress or your local bookstore document the X11R5 and DPS features. (See the Preface of the *OpenWindows 3.3 Programmer's Guide* for a complete list of recommended books.)

The server's program name for this release is `Xsun`. See the `Xsun` man page and the *OpenWindows 3.3 Programmer's Guide* for detailed information on the server and its command line options. The font directory is changed to be compatible with MIT's X11R5 server. See the *OpenWindows 3.3 Programmer's Guide* for details.

The following two sections discuss the enhancements to the X11R5 server and the DPS imaging extension implemented in the OpenWindows server.

X11R5 Enhancements

This section outlines the OpenWindows 3.3 server enhancements to X11R5.

Server Extensions

- DPS
- XInput
- Multi-Buffering
- MIT-SHM
- SHAPE
- XTEST
- MIT–SUNDRY–NONSTANDARD

See the *OpenWindows 3.3 Programmer's Guide* for more information on these extensions.

Additional Outline and Bitmap Fonts

- Support for F3 Latin and Asian fonts (`.f3b`)
- 72dpi pre-scaled bitmap fonts for the 57 F3 fonts
- Support for the X11/NeWS bitmap font format (`.fb`)

See the *OpenWindows 3.3 Programmer's Guide* for more information on font features.

Font Encoding Support

This release supports the following charset encodings for F3 scalable fonts:

- jisx0208
- jisx0201
- iso8859-1
- symbol
- dingbats

If you have an F3 format scalable font that has an encoding other than one of the above, you must provide a `.enc` file for the charset encoding your font supports. The directory `/usr/openwin/lib/X11/fonts/encodings` contains the default encoding descriptions.

Graphics Hardware Support

- Multiple Plane Group support with minimized exposure
- Multiple Hardware Colormap support
- Multibuffering support
- Direct graphics access for XGL and XIL libraries

See the *OpenWindows 3.3 Programmer's Guide* for more information.

Keyboard Enhancements

- Compose key processing

  This allows for certain special symbols and international characters that may not be physically engraved on the keyboard.
- International keyboard

  Supports all Sun International keyboards.
- Keyboard LED light
- Numeric keypad keys and NumLock

DPS Enhancements

The OpenWindows server includes the following font enhancements to DPS:

- Support for F3 Latin fonts
- Support for X11R5 bitmap fonts in the portable compiled format (`.pcf`)
- Scaling of Type 1 fonts by Adobe software

Unsupported X11/NeWS Features

The OpenWindows 3.3 server *does not* support the following X11/NeWSwindow server features:

- SunView compatibility
- EVQ input driver support
- NeWS and TNT runtime support
- CPS (C-to-PostScript) functionality
- Non-outline fonts not available from DPS
- XVPS library
- `_SUN_PER_DEPTH_CMAPS`

  The `_SUN_PER_DEPTH_CMAPS` root window property is no longer supported. This property provided detailed information about exactly how many LUTs (lookup tables) are used by a given depth.

Differences Between NeWS and DPS

DPS is an extension to the Version 3.3 window server; the NeWS programming language is limited to servers in prior releases. This section briefly highlights the differences between NeWS and DPS features.

The NeWS programming language is primarily PostScript Level I. It uses PostScript operators to display text and images on a graphics console. NeWS is a fully functional window system, complete with its own memory management and input system.

The DPS imaging extension is PostScript Level II. It is only an imaging system. Since DPS is PostScript Level II, it has the following features:

- Memory management including memory reclamation
- Resources
- Compression/decompression
- Filters
- Binary encoding
- User paths
- Forms
- Color spaces

- Efficient graphics operators (rectangle ops, xyshow)
- Patterns
- Improved halftoning