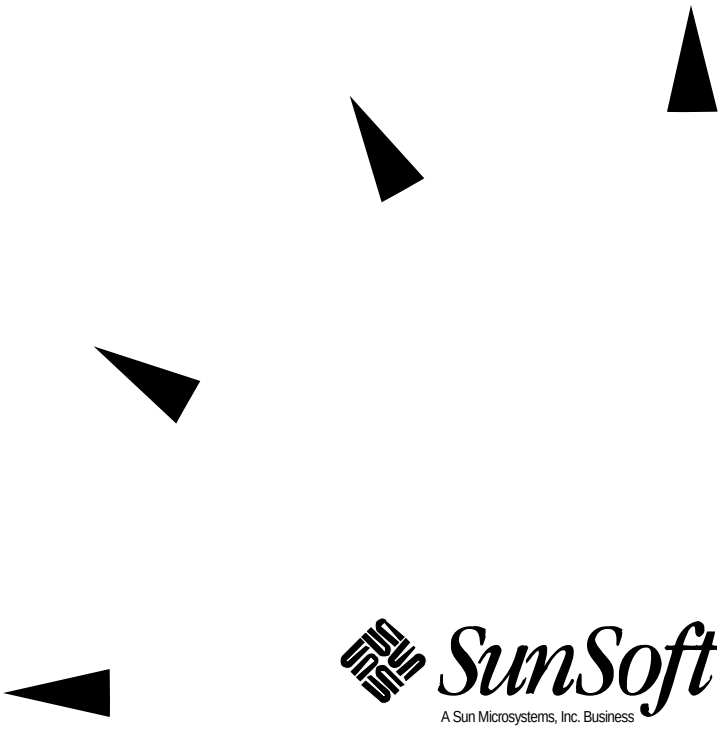
Security, Performance, and Accounting Administration



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

SunSoft
A Sun Microsystems, Inc. Business

Contents

Figures

Tables

Preface

This book is part of a multibook set describing system and network administration for the Solaris™ operating environment and the SunOS™ system software. This book describes how to monitor and improve the performance of your system, how to use the accounting packages that are available with SunOS system software, and how to maintain a secure system. It assumes that you have already installed SunOS software on your systems. This book is intended as a guide for maintaining both client and server systems.

How to Use This Book

Each chapter in this book is divided into three major sections:

- The *About* section
- The *Instructions* section
- The *Reference* section (where applicable)

The *About* section explains all the background and concepts you need to perform a particular task.

The *Instructions* section presents the steps you follow to complete various tasks.

The *Reference* section provides any charts, tables, and other reference material you may need to consult when performing tasks.

You can read the whole chapter or go directly to the type of information you need.

Who Should Use This Book

This book is intended for administrators or users responsible for one or more systems. Some basic knowledge of the UNIX® operating system and text editing is expected, although experience administering systems is not assumed. The book can be used by both beginning and experienced system administrators.

How This Book Is Organized

This book is organized as follows:

Part I—Security

These chapters describe how to maintain security for your system, including protecting your system, files, data, and network.

Chapter 1, "Introduction to Security Administration," gives an overview of security administration.

Chapter 2, "Securing System Access," describes how to secure a system from unauthorized users.

Chapter 3, "Securing Files and Data," describes how to secure your files and the data contained in files.

Chapter 4, "Securing the Network," describes how to secure the hosts connected on a network.

Chapter 5, "Monitoring and Controlling Security Using ASET," describes how to use the Automated Security Enhancement Tool (ASET), an automated tool for performing many common security checks.

Part II—Performance and Accounting

These chapters describe how to maintain and monitor the performance of your system, and describe the available accounting tools.

Chapter 6, "Introduction to Performance," gives an overview of managing performance.

Chapter 7, "Managing Processes," describes how to look at and change the priority of processes on the system**.**

Chapter 8, "Monitoring Performance," describes the tools that allow you to monitor system activity.

Chapter 9, "A Guide to Network Performance," describes the tools that allow you to monitor network performance.

Chapter 10, "Setting Up and Maintaining Accounting," describes setting up the accounting utility and maintaining the resulting files.

Appendix A, "Tuning Kernel Parameters," lists and describes the kernel parameters that you can change.

Appendix B, "The Scheduler," describes the process scheduler and how it works.

Appendix C, "Error Messages," lists and describes error messages produced by the accounting and ASET tools.

Related Books

In addition to this book, the following books offer valuable information:

Managing NFS and NIS, by Hal Stern, O'Reilly & Associates, Inc.

Practical Unix Security, by Garfinkel and Spafford, O'Reilly & Associates, Inc.

System Performance Tuning, by Mike Loukides, O'Reilly & Associates, Inc.

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:

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

In addition, the following convention is used in this book:

`command(n)` refers to the section in the *SunOS Reference Manual* in which the command is described. For example, `chown(1M)` refers to the `chown` command in Section 1M of the Reference Manual. Most system administration-related commands are described in Section 1M.

Part 1— Security

Part 1 contains the following chapters:

- “Introduction to Security Administration”
- “Securing System Access”
- “Securing Files and Data”
- “Securing the Network”
- “Monitoring and Controlling Security Using ASET”

Introduction to Security Administration 1

Keeping the computer system's information secure is an important system administration responsibility. This chapter gives an overview of the various aspects of security that an administrator must be concerned with, both at the system and network levels.

Overview of Security Administration

Security is provided at three levels:

- User
- System
- Network

At the user level, the SunOS operating system provides some standard security features that you can use to protect files, directories, and devices.

At the system and network levels, the security issues are mostly the same. In the workplace, a number of systems connected to a server can be thought of as one large multifaceted system. The system administrator is responsible for the security of this larger system or network. Not only is it important to defend the network from outsiders trying to gain access to the network, but it is also important to ensure the integrity of the data on the systems within the network.

Several tools are available to help the system administrator implement and maintain a secure environment. These tools include Secure RPC and the Automated Security Enhancement Tool (ASET).

1

Secure RPC provides an effective method to authenticate a user requesting access to files and data over the network. ASET enables you to automatically monitor and control system security.

Granting Access to a Computer System

The first line of defense is to control access to your system. You can control access by:

- Maintaining physical site security
- Maintaining login control
- Restricting access to data in files
- Maintaining network control
- Monitoring system usage
- Setting the path variable correctly
- Monitoring `setuid` programs
- Tracking root login
- Installing a firewall

Maintaining Physical Site Security

To control access to your system, you must maintain the physical security of your computer environment. For instance, if a system is logged in and left unattended, anyone who can use that system can gain access to the operating system and the network. You need to be aware of your computer's surroundings and physically protect it from unauthorized access.

Maintaining Login and Access Control

You also must restrict unauthorized logins to a system or the network, which you can do through password and login control. All accounts on a system should have a password. An account without a password makes your entire network accessible to anyone who can guess a user name. Chapter 2, "Securing System Access," outlines methods for password and login control.

Solaris 2.x system software restricts control of certain system devices to the user login account. Only a process running as root or console user can access a system mouse, keyboard, frame buffer, or audio device unless `/etc/logindevperm` is edited. See `logindevperm(4)`for more information.

Restricting Access to Data in Files

After you have established login restrictions, you can control access to the data on your system. You may want to allow some people to read some files, and give other people permission to change or delete some files. You may have some data that you do not want anyone else to see. Chapter 3, “Securing Files and Data,” discusses how to set file permissions.

Maintaining Network Control

Computers are often part of a configuration of machines called a *network.* A network allows connected machines to exchange information and access data and other resources available from machines connected to the network. Networking has created a powerful and sophisticated way of computing. However, networking has also jeopardized computer security.

For instance, within a network of computers, individual systems are open to allow sharing of information. Also, because many people have access to the network, there is more chance for allowing unwanted access, especially through user error, for example, through a poor use of passwords. Chapter 4, “Securing the Network,” discusses ways that you can use to maintain a secure network.

Monitoring System Usage

As system administrator, you need to monitor system activity, being aware of all aspects of your systems, including the following:

- What is the normal load?
- Who has access to the system?
- When do individuals access the system?

With this kind of knowledge, you can use the available tools to audit system use and monitor the activities of individual users. Monitoring is very useful when there is a suspected breach in security.

Setting the Correct Path

It is important to set your path variable correctly; otherwise, you may accidently run a program introduced by someone else that harms your data or your system. This kind of program, which creates a security hazard, is referred

to as a “Trojan horse.” For example, a substitute `su` program could be placed in a public directory where you, as system administrator, might run it. Such a script would look just like the regular `su` command; since it removes itself after execution, it is hard to tell that you have actually run a Trojan horse, rather than just mistyped your password.

The path variable is automatically set at login time through the startup files: `.login`, `.profile`, and `.cshrc`. Setting up the user search path so that the current directory (.) comes last prevents you or your users from running this type of Trojan horse. The path variable for root should not include the current directory at all. The ASET utility examines the startup files to ensure that the path variable is set up correctly and that it does not contain a dot (.) entry.

`setuid` *Programs*

Many executable programs have to be run as root (that is, as superuser) to work properly. These executables run with the user ID set to 0 (`setuid=0`). Anyone running these programs runs them with the root ID, which creates a potential security problem if the programs are not written with security in mind.

Except for the executables shipped with `setuid` to root, you should disallow the use of `setuid` programs, or at least restrict and keep them to a minimum.

Tracking Root Login

Your system requires a root password to boot into superuser mode. In the default configuration, a user cannot remotely log in to a system as root. When logging in remotely, a user must log in as himself and then use the `su` command to become root. This enables you to track who is using root privileges on your machine.

Installing a Firewall

Another way to protect your network is to use a firewall or secure gateway machine. A firewall is a dedicated machine separating two networks, each of which approaches the other as untrusted. You should consider this setup as mandatory between your internal network and any external networks, such as Internet™, with which you want internal network users to communicate.

A firewall can also be useful between some internal networks. For example, the firewall or secure gateway computer will not send a packet between two networks unless the gateway computer is the origin or the destination address of the packet. A firewall should also be set up to forward packets for particular protocols only. For example, you may allow packets for transferring mail, but not those for `telnet` or `rlogin`. The ASET utility, when run at high security, disables the forwarding of Internet Protocol (IP) packets.

Reporting Security Problems

If you experience a suspected security breach, you can contact the Computer Emergency Response Team/Coordination Center (CERT/CC), which is a Defense Advanced Research Projects Agency (DARPA) funded project located at the Software Engineering Institute at Carnegie Mellon University. It can assist you with any security problems you are having. It can also direct you to other Computer Emergency Response Teams that may be more appropriate to your particular needs. You can call CERT/CC at its 24-hour hotline: (412) 268-7090, or contact the team via email to `cert@cert.sei.cmu.edu`.

1

Securing System Access 2

This chapter describes how to safeguard your computer against unauthorized access. It discusses how to prevent an intruder from logging in to your machine, how to maintain the password files, and how to prevent unauthorized root access to sensitive system files and programs.

The first section of this chapter gives you some background information about logins, passwords, and privileged access. You can skip this information and go right to the instructions. Use the following table to find the page where instructions for a specific task begins.

2

About Restricting Access to Your System

The first security barrier an intruder must cross is the login program. To cross this barrier, a user must supply a user name and a corresponding password known by the local machine or by the name service (NIS or NIS+).

A second line of defense makes sure the system files and program can be changed or removed by root or superuser only. A would-be superuser must supply the root user name and its correct password.

Restricting Login Access

When a user logs in to a machine, the login program consults the appropriate database according to the information listed in the `/etc/nsswitch.conf` file. The entries in this file can include `files` (designating the `/etc` files), `nis` (designating the NIS database), and `nisplus` (designating the NIS+ database). See *Name Services Administration Guide* or the reference page, `nsswitch.conf(4)`, for a description of this file.

The login program verifies the user name and password entered. If the user name is not in the password database or the password is not correct for the user name, the login program denies access to the machine. When the user supplies a name from the password database and the correct password for the name, the system grants the user access to the machine.

Passwords

When logging in to a machine, users must enter both a user name (or login) and a password. Although logins are publicly known, passwords must be kept secret, and known only to users. You should ask your users to choose their passwords carefully, and change them often.

Choosing a Password

Many breaches of computer security involve guessing a legitimate user's password. While the `passwd` command enforces some criteria for making sure the password is hard to obtain, a clever person can sometimes guess a password just by knowing something about the user.

Bad Choices for Passwords

Bad choices for passwords include:

- Your name, forwards, backwards or jumbled
- Names of family members or pets
- Car license numbers
- Telephone numbers
- Social Security numbers
- Employee numbers
- Names related to a hobby or interest
- Seasonal themes, such as Santa in December
- Any word in the dictionary

Good Choices for Passwords

Good choices for passwords include:

- Phrase (`beammeup`)
- Nonsense words made up of the first letters of every word in a phrase (`swotrb` for SomeWhere Over The RainBow)
- Words with numbers, or symbols substituted for letters (`sn00py` for snoopy)

Password Databases

The password databases include the databases maintained by the network information services (NIS and NIS+) and the `/etc` files.

NIS Password Database

If your network uses the NIS name service, you can edit your machine's `/etc/passwd` file so that it additionally uses the NIS password facilities. If you add a special entry to this file (an line containing a plus sign (+)), the login program will also search for user name in the NIS maps.

When a user runs the `passwd` command to change his or her password, the program first checks to see if the user has an entry in the local `/etc/passwd` file. If there is no such entry, the program looks for the + escape line in the `passwd` file. If the escape line is there, and NIS is running, the program

changes the user's password on the NIS master server for the `passwd` map. The command, `yppasswd`, accesses NIS directly. For information about the NIS name service, see *NFS Administration Guide.*

NIS+ Password Database

If your network uses NIS+, the password information is kept in the NIS+ databases. Information in these databases can be protected by restricting access to authorized users. You can use Administration Tool to access or change information in the NIS+ files. For information about the NIS+ databases and NIS+ security, see *Name Services Administration Guide.*

`/etc` *Files*

The `/etc` files include `/etc/passwd` and `/etc/shadow`. The user name and other information is kept in the password file `/etc/passwd`, while the encrypted password itself is kept in a separate *shadow* file `/etc/shadow`. This is a security measure that prevents a user from gaining access to the encrypted passwords. While the `/etc/passwd` file is available to anyone who can log in to a machine, only root can read the `/etc/shadow` file. The `passwd` commands described in this chapter affect the `/etc` files on the local machine.

Password Aging

For instructions on how to enable password aging, turn to "How to Enable and Disable Password Aging" on page 18.

To make your computer systems more secure, ask users to change their passwords periodically. For a high level of security, you should require users to change their passwords every six weeks. Once every three months is adequate for lower levels of security. System administration logins (such as `root` and `sys`) should be changed monthly, or whenever a person who knows the root password leaves the company or is reassigned.

The password-aging mechanism forces users to change their passwords periodically. It also prevents a user from changing a password before a specified interval. You can apply password aging to a login account by using the `passwd(1)` command. If you require more access control than that provided by password aging, you can also create a dial-up password that requires a second access code as part of the login process. See "Password Protection Using Dial-Up Passwords" on page 11.

Password Protection Using Dial-Up Passwords

For instructions on how to create dial-up passwords, turn to "How to Create a Dial-Up Password" on page 22.

You can add a layer of security to your password mechanism by requiring a *dial-up password* for users who access the computer through a modem or dial-up port. A dial-up password is an additional password that a user must enter before being granted access to the machine.

Only superuser can create or change a dial-up password. To ensure the integrity of the system, the password should be changed about once a month. The most effective use of this mechanism is to require a dial-up password to gain access to a gateway machine.

Two files are involved in creating a dial-up password, `/etc/dialups` and `/etc/d_passwd`. The first contains a list of ports that require a dial-up password, and the second contains a list of shell programs that require an encrypted password as the additional dialup password.

The `/etc/dialups` file is a list of terminal devices, for example:

```
/dev/term/a
/dev/term/b
```

The `/etc/d_passwd` file has two fields. The first is the login shell that will require a password, and the second is the encrypted password, for example:

```
/usr/lib/uucp/uucico:encrypted_password:
/usr/bin/csh:encrypted_password:
/usr/bin/ksh:encrypted_password:
/usr/bin/sh:encrypted_password:
```

When a user attempts to log in on any of the ports listed in `/etc/dialups`, the login program looks at the user's login entry stored in `/etc/passwd`, and compares the login shell to the entries in `/etc/d_passwd`. These entries determine whether the user will be required to supply the dial-up password.

The basic sequence is illustrated by the following figure:

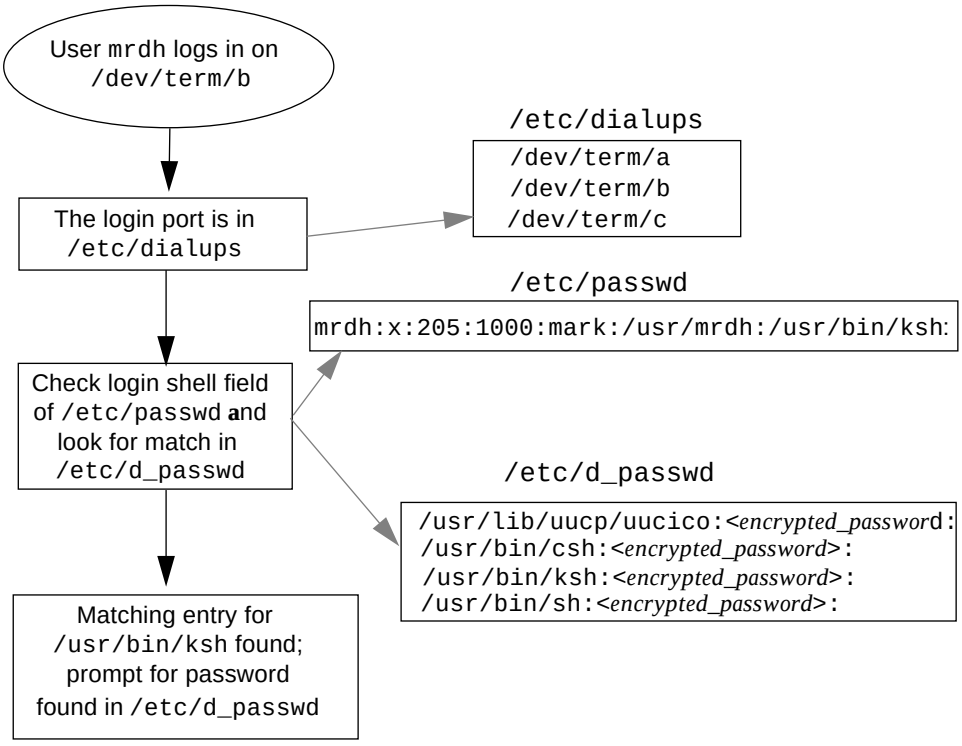


Figure 2-1 Basic Dial-up Password Sequence

The `/etc/d_passwd` *File*

Because most users will be running a shell when they log in, all shell programs should have entries in `/etc/d_passwd`. Such programs include `uucico`, `sh`, `ksh`, and `csh`. If some users run something else as their login shell, include that login shell in the file, too.

If the user's login program (as specified in `/etc/passwd`) is not found in `/etc/d_passwd`, or if the login shell field in `/etc/passwd` is null, the password entry for `/usr/bin/sh` is used.

- If the user's login shell in `/etc/passwd` matches an entry in `/etc/d_passwd`, the user must supply a dial-up password.
- If the user's login shell in `/etc/passwd` is not found in `/etc/d_passwd`, the user must supply the default password. The default password is the entry for `/usr/bin/sh`.
- If the login shell field in `/etc/passwd` is empty, the user must supply the default password (the entry for `/usr/bin/sh`).
- If `/etc/d_passwd` has no entry for `/usr/bin/sh`, then those users whose login shell field in `/etc/passwd` is empty or does not match any entry in `/etc/d_passwd` will not be prompted for a dial-up password.
- Dial-up logins are disabled if `/etc/d_passwd` has only the following entry: `/usr/bin/sh:*:`

Restricted Shell

The standard shell allows a user to open files, execute commands, and so on. The restricted shell can be used to limit the ability of a user to change directories, and execute commands. The restricted shell (`rsh`) is located in the directory `/usr/lib.` (Note that this is not the remote shell, which is `/usr/sbin/rsh`.) The restricted shell differs from the normal shell in these ways:

- The user is limited to the home directory (can't use `cd` to change directories).
- The user can use only commands in the `PATH` set up by the system administrator (can't change the `PATH` variable).
- The user can access only files in the home directory and its subdirectories (can't name commands or files using a complete path name).
- The user cannot redirect output with `>` or `>>`.

The restricted shell allows the system administrator to limit a user's ability to stray into the system files, and is intended mainly to set up a user who needs to perform specific tasks. The `rsh` is not completely secure, however, and is only intended to keep unskilled users from getting into (or causing) trouble.

See the reference page for `sh(1)` for information about the restricted shell.

2

Restricting Root Access

The root (or superuser) account is used by the operating system to accomplish basic functions, and has wide-ranging control over the entire operating system. It has access to and can execute essential system programs. For this reason, there are almost no security restraints for any program that is run by root.

The system administrator can protect the root account by restricting root access to a specific device. For example, if root access is restricted to the console, a user can log in to a machine as root only from the console. A user who remotely logs in from another machine to perform an administrative function must first log in as himself and then use the `su` command to become superuser.

The `/etc/default` *Directory*

The /etc/default directory contains ASCII files that are used to control and monitor root access.

- An entry in the file `/etc/default/login` determines the root access restrictions.
- Entries in the file `/etc/default/su` determine the default conditions of the `su` command.
  - One entry enables or disables a log of each time the `su` command is used to change to another user.

    A record of every time the `su` command is used, who uses it, and when is made in the log file, `/var/adm/sulog`, enabling you to track who is using the superuser account.
  - Another entry enables or disables a display on the console each time an attempt is made to use the `su` command to gain root access from a remote system.

For instructions on how to control root access and monitor the superuser log, turn to “How to Monitor and Control su Use” on page 23.

Maintaining a Log of Unsuccessful Login Attempts

For instructions on how to create a login log file, turn to "How to Enable Login Logging" on page 20.

You can create a log to keep track of unsuccessful attempts to log in to the computer. After a person makes five consecutive unsuccessful attempts to log in, all these attempts are recorded in the file `/var/adm/loginlog` (as long as it exists).

If a person makes fewer than five unsuccessful attempts, none of them is logged. If `/var/adm/loginlog` does not exist, nothing is logged.

Special Logins

There are two common ways to access a machine—by using a conventional user login or by using the root login. In addition, a number of special *system* logins allow a user to perform administrative commands without using the root account. The administrator assigns password to these login accounts.

Table 2-1 lists the system login accounts and their uses. The system logins perform special functions, and each has its own group identifier number (GID). Each of these logins should have its own password, and these passwords should be distributed on a need-to-know basis.

Table 2-1 System Logins

Login Account	GID	Use
`root`	`0`	Has almost no restrictions and overrides all other logins, protections, and permissions. The root account has access to the entire system. The password for the root login should be very carefully protected.
`daemon`	`1`	Controls background processing.
`bin`	`2`	Owns most of the commands.
`sys`	`3`	Owns many system files.
`adm`	`4`	Owns certain administrative files.
`lp`	`71`	Owns the object and spooled data files for the printer.
`uucp`	`5`	Owns the object and spooled data files for UUCP, the UNIX-to-UNIX copy program.
`nuucp`	`9`	Is used by remote machines to log in to the system and start file transfers.

You should also set the security of the `eeprom` to require a password. See the reference page for `eeprom(1M)` for more information.

Instructions for Securing and Controlling System Access

This section gives instructions on how to use the `passwd` and `login` commands to control and track access to accounts on the system. The instructions in this section describe the commands that act upon a local machine. To add and manage user accounts using `admintool`, see *User Accounts, Printers, and Mail Administration*. For information about managing and controlling access to the network databases, see *NFS Administration Guide* and *Name Services Administration Guide.*

▼ How to Change, Lock, or Show Status of Passwords

If you need more information about passwords, turn to "Passwords" on page 8.

The `passwd` command enables you to change, delete, or lock a password, as well as enable and disable password aging. You must be superuser to use most of these commands.

To change your own password:

- **Type** `passwd` **and press Return.**
  You are prompted for your old password. Then you are prompted for your new password. The password is not displayed as it is typed, and you must enter the new password again for confirmation.

Code Example 2-1

```
example% passwd
Changing password for charlie
Old password:
New password:
Retype new password:
Password changed for charlie
example%
```

To change a user's password:

- **As root, type** `passwd` *username* **and press Return.**
  You are not prompted for the old password. But you are prompted to enter the new password. The password must be entered again for confirmation.

To prevent a user from changing a password:

♦ **As root, type** `passwd -n 10 -x 7` *username* **and press Return.**
Because *min* (`-n 10`) is greater than *max* (`-x 7`), the password is locked and cannot be changed. The user can still log in to the machine, but only root can change this password.

To display information about passwords:

♦ **As root, type** `passwd -s` *username* **and press Return.**
The following example displays information about user `charlie`, if password aging is enabled.

```
# passwd -s charlie
charlie PS 6/23/94 1 90 7
```

(If password aging is not turned on, only the first two fields appear.) The six fields contain the following information:

- Login name (`charlie`)
- Password status (`PS`) as follows:
  `NP`—No password for this `login`
  `LK`—Login is locked
  `PS`—Anything else
- Date the password was last changed (`6/23/94`)
- Minimum number of days after the last password change before the user can change the password (`1`)
- Maximum number of days between password changes (`90`)
- Number of warning days before the password must be changed (`7`)

Thus, the information obtained for this example shows that there is a password for user `charlie` that cannot be changed before June 24, 1994 and that must be changed by September 21, 1994. On September 14, 1994, this user will begin seeing a warning message that the password will expire and should be changed.

To display password status for all users:

♦ **As root, type** `passwd -s -a` **and press Return.**
Only a privileged user can use the `-a` option for the `passwd` command.

▼ How to Enable and Disable Password Aging

To enable password aging:

♦ **As root, type the following command:**

`passwd -n` *min* `-x` *max* `-w` *warn username*

This command sets the minimum number of days between password changes (`-n` *min*), the number of days the password is valid (`-x` *max*), and the number of days before the password expires that the user is warned (`-w` *warn).*

Example of Enabling Password Aging

This example sets up password aging for user `charlie`. User `charlie` must change his password every 90 days, and must wait a day before he can change it again. Starting seven days before the password expires, `passwd` prints a warning whenever `charlie` logs in to his machine.

```
# passwd -n 1 -x 90 -w 7 charlie
```

To disable password aging:

♦ **As root, type** `passwd -x -1` *username* **and press Return.**
Setting *max* to `-1` turns off aging.

▼ How to Force a User to Enter a New Password

♦ **As root, type** `passwd -f` *username* **and press Return.**
The `-f` option forces a password change. At next login, the user is prompted for a new password.

▼ How to Display Login Information

The login commands operate on the local `/etc/passwd` files and also on the NIS and NIS+ password databases.

To display login status for a user:

♦ **Type** `logins -x -l` *username* **and press Return.**
The information includes the login, UID, GID, user name, home directory, login shell, and password aging information—the date the password was last changed, the number of days required between changes, the number of days allowed before a change is required, and the warning period.

If you need background information about login access, turn to "Restricting Login Access" on page 8.

Example of Displaying Login Information

This example displays information about login `charlie`.

```
# logins -x -l charlie
charlie        200                                         200
charlie
                                /home/charlie
                                /bin/sh
                                PS 062392 1 90 7
```

To show logins that have no passwords:

♦ **Type** `logins -p` **and press Return.**
Use the output of this command to make sure that all users on the system have a valid password.

▼ How to Enable Login Logging

If you need background information about logging logins, turn to “Maintaining a Log of Unsuccessful Login Attempts” on page 15.

To enable login logging, create the log file `loginlog`, with read and write permission for root only, as follows:

1. **As root, type** `touch /var/adm/loginlog` **and press Return.**
   This creates the log file `loginlog`.
2. **Type** `chmod 600 /var/adm/loginlog` **and press Return.**
   This sets read and write permissions for root on the file.
3. **Type** `chgrp sys /var/adm/loginlog` **and press Return.**
   This sets the group to `sys`.

The `loginlog` file may grow quickly. To use the information in this file and to prevent the file from getting too large, you must check and clear its contents occasionally. If this file shows a lot of activity, it may suggest an attempt to break into the computer system. For more information about this file, see `loginlog(4)`.

▼ How to Set Up Automatic Account Expiration

The `usermod` and `useradd` commands work only on the local machine. Use `admintool` to add a user to the network. See *User Accounts, Printers, and Mail Administration*.

To set up account expiration for a new account:

- **As root, type** `useradd -e` *mm/dd/yy username* **and press Return.**
  mm/dd/yy is the date the account will expire.

Example of Setting Up Automatic Account Expiration

The following example sets up a new login, `shortterm`, which expires on July 31, 1993.

```
# useradd -e 07/31/94 shortterm
```

To extend a login's expiration date:

♦ **As root, type** `usermod -e` *newdate username* **and press Return.**
newdate is a new expiration date.

```
# usermod -e 09/10/94 shortterm
```

▼ How to Disable and Re-Enable Inactive Accounts

To disable an inactive account:

♦ **As root, type** `usermod -f` *n username* **and press Return.**
The `-f` *n* option to `usermod` sets the "inactive" field. A login is considered inactive if a user has not logged in for *n* number of days. Once the account has been disabled, the user cannot log in until the administrator resets the login account.

Example of Disabling a User Account

This example disables the user account for `charlie` if it is inactive for 30 days.

```
# usermod -f 30 charlie
```

To re-enable a disabled account:

1. **As root, type** `usermod -f 0` *username* **and press Return.**
2. **Have the affected user log in again so that the** *lastlogin* **date will be updated.**
3. **Type** `usermod -f` *n username* **and press Return.**
This resets the time the account can be inactive.

▼ How to Create a Dial-Up Password

If you need more information about dial-up passwords, turn to "Password Protection Using Dial-Up Passwords" on page 11.

Caution – When you first establish a dial-up password, be sure to remain logged in on at least one terminal while testing the password on a different terminal. If you make a mistake while installing the extra password and log off to test the new password, you might not be able to log back on. If you are still logged in on another terminal, you can go back and fix your mistake.

You must be root to perform these tasks.

1. **Create an** `/etc/dialups` **file.**
   This file should contain a list of terminal devices. List all the ports that will require dial-up password protection. The `/etc/dialups` file should look like this:

```
/dev/term/a
/dev/term/b
/dev/term/c
```

2. **Create an** `/etc/d_passwd` **file.**
   This file should contain the login programs that will require a dial-up password, and the encrypted dial-up password. List all shell programs that a user could be running when he logs in, for example, `uucico`, `sh`, `ksh`, and `csh`. The `/etc/d_passwd` file should look like this:

```
/usr/lib/uucp/uucico:encrypted_password:
/usr/bin/csh:encrypted_password:
/usr/bin/ksh:encrypted_password:
/usr/bin/sh:encrypted_password
```

3. **Type** `chown root /etc/dialups /etc/d_passwd` **and press Return.**
   This command sets the owner and group to root.
4. **Type** `chgrp root /etc/dialups /etc/d_passwd` **and press Return.**
5. **Type** `chmod 600 /etc/dialups /etc/d_passwd` **and press Return.**
   This sets the modes of these files to read and write for the owner, root.
6. **Create the encrypted passwords.**

a. Type `useradd dummy` **and press Return.**
This creates a dummy user.

b. Type `passwd dummy` **and press Return.**
This creates a password for the dummy user.

c. Type `grep dummy /etc/shadow > dummy.temp` **and press Return.**
This captures the encrypted password.

d. Edit the file `dummy.temp`**.**
Open `dummy.temp` for editing, and delete all fields except the encrypted password (the second field).

For example, in the following line, the encrypted password is `U9gp9SyA/JlSk`.

```
dummy:U9gp9SyA/JlSk:7967:::::7988:
```

e. Type `userdel dummy` **and press Return.**
This deletes the dummy user.

7. Edit the file `/etc/d_passwd`**.**
Open `d_passwd` and copy the encrypted password from your `dummy.temp` file into the password file. You can create a different password for each login shell, or use the same one for each.

♦ **To temporarily disable dial-up logins, put the following entry by itself into the** `/etc/d_passwd` **file:**

```
/usr/bin/sh:*:
```

▼ How to Monitor and Control `su` Use

If you need more information about monitoring `su` use, turn to "Restricting Root Access" on page 14.

To monitor `su` ***use:***

1. Open the file `/etc/default/su` **for editing.**

2. Uncomment the line: `SULOG=/var/adm/sulog`
Whenever the `su` command is run from a remote system, a message is logged to the file `/var/adm/sulog`.

To check the superuser log:

♦ **Type** `more /var/adm/sulog` **and press Return.**
The log file lists each root access.

Example of `/var/adm/sulog` *File*

The `/var/adm/sulog` file lists all uses of the `su` command, not only those used to switch user to `root`. The entries show the date and time the command was entered, whether or not it was successful (+ or –), the port from which the command was issued, and finally, the name of the user and the switched identity.

Note that user `rar` first switched user to `charlie` and then from `charlie` to `root`.

```
# more /var/adm/sulog
SU 10/23 16:41 - pts/3 rar-root
SU 10/23 16:41 + pts/3 jjones-root
SU 10/23 19:07 + pts/0 rar-charlie
SU 10/23 19:08 + pts/0 charlie-root
SU 10/23 19:16 + console root-root
```

To display each instance of `su root` ***to the console:***

1. **Open the file** `/etc/default/su` **for editing.**
2. **Uncomment the line:** `CONSOLE=/dev/console`
Whenever the `su root` command is run from a remote system, a message is printed on the system console.

▼ How To Restrict Root Login to the Console:

1. **Open the file** `/etc/default/login` **for editing.**
2. **Uncomment the line:** `CONSOLE=/dev/console`
This restricts root access to the console. Any user who tries to remotely log in to a system must first log in as himself, and then use the `su` command to become root.

Securing Files and Data 3

The SunOS operating system is a multiuser system, which means that all the users logged in to a machine can read and use files belonging to one another, as long as they have permission to do so.

This chapter describes how to secure the files and directories in a file system. If you are familiar with the concepts of file protection, use the following table to find instructions for the task that you want to perform.

3

About File Access

Every file and directory can be assigned three basic file permissions:

- read (`r`)
- write (`w`)
- execute (`x`)

This group of three permissions, called *triplets*, can be assigned to the three classes of users:

- The file or directory owner—usually the user who created the file. The owner of a file can decide who has the right to read it, to write to it (make changes to it), or, if it is a command, to execute it.
- Members of a group
- All others who are not the file or group owner

Only the owner of the file or the superuser can assign or modify file permissions.

File Permissions

Table 3-1 lists and describes the file permissions.

Table 3-1 File Permissions

Symbol	Permission	Means Designated Users...
`r`	Read	Can open and read the contents of a file
`w`	Write	Can write to the file (modify its contents), add to it, or delete it
`x`	Execute	Can execute the file (if it is a program or shell script), or run it with one of the `exec(2)` system calls
`–`	Denied	Cannot read, write, or execute the file

These file permissions apply to special files such as devices, sockets, and named pipes (FIFOs), as they do to regular files.

For a symbolic link, the permissions that apply are those of the file the link points to.

Directory Permissions

Table 3-2 lists and describes the directory permissions.

Table 3-2 Directory Permissions

Symbol	Permission	Means Designated Users...
r	Read	Can list files in the directory.
w	Write	Can add or remove files or links in the directory.
x	Execute	Can open or execute files in the directory. Also can make the directory and to the directories beneath it current.

You can protect the files in a directory (and in its subdirectories) by disallowing access to that directory. Note, however, that root (superuser) has access to all files and directories on the system.

File Types

A file can be one of six types. Table 3-3 lists the possible file types.

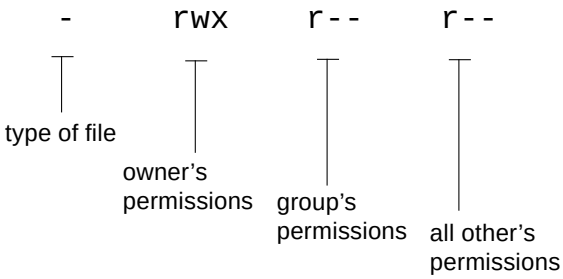
Table 3-3 File Types

Symbol	Type
–	Text or program
d	Directory
b	Block special file
c	Character special file
p	Named pipe (FIFO)
l	Symbolic link

Viewing Permissions

You can view permissions on the files and subdirectories within the current directory by typing `ls -l`. The first column of output describes the *mode* of the file. This information tells you what type of file it is, and who has permission to access it.

For example, a file with a listing of:



indicates that this is a text file with read, write, and execute permissions for its owner, read-only permission for groups and others.

Changing Permissions With `chmod`

You must be root or the owner of a file or directory to assign or change its permissions.

The `chmod` command sets or changes the permissions on a file. You can use this command to set permissions in either of two modes: *symbolic* or *absolute*. Symbolic mode uses combinations of letters and symbols to add or remove permissions. Absolute mode uses numbers to represent file permissions.

For instructions on how to assign permissions in symbolic mode, turn to "How to Change Permissions in Symbolic Mode" on page 39.

Symbolic Mode

Symbolic mode uses letters to specify whose privileges are being modified, and in what way. Table 3-4 lists the symbols that specify whose permissions are to be set or changed, the operation to be performed, and the permissions being assigned or changed.

Table 3-4 Permission Symbols and Their Meanings

Symbol	Function	Meaning
`u`	Who	User (owner)
`g`	Who	Group
`o`	Who	Others
`a`	Who	All
`=`	Operation	Assign
`+`	Operation	Add
`–`	Operation	Remove
`r`	Permission	Read
`w`	Permission	Write
`x`	Permission	Execute
`l`	Permission	Mandatory locking, `setgid` bit is on, group execution bit is off
`s`	Permission	`setuid` or `setgid` bit is on
`S`	Permission	`suid` bit is on, user execution bit is off
`t`	Permission	Sticky bit is on, execution bit for others is on
`T`	Permission	Sticky bit is on, execution bit for others is off

The read, write, and execute permissions are explained in “File Permissions” on page 26. The `setuid`, `setgid`, and sticky bits are explained in “Special Permissions (setuid, setgid and Sticky Bit)” on page 31.

Absolute Mode

For instructions on how to assign permissions in absolute mode, turn to “How to Set Permissions in Absolute Mode” on page 38.

The absolute mode is the method most commonly used to set permissions. This mode uses the octal numeric value representing owner, group and others, as illustrated below:

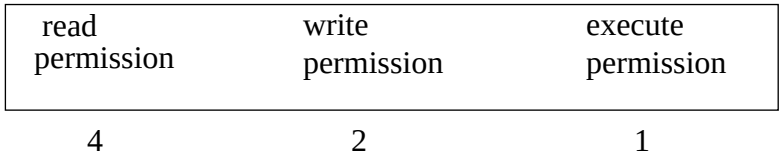


Table 3-5 lists the octal values and their meanings.

Table 3-5 Octal Values and Permissions Granted

Octal Value	Permission Assigned
0	No permissions
1	Execute permission only
2	Write permission only
3	Write and execute permissions
4	Read permission only
5	Read and execute permissions
6	Read and write permissions
7	Read, write, and execute permissions

When you change permissions by using the absolute mode, represent permissions for each triplet by an octal mode number. Thus, the following command sets read, write, and execute permissions for owner; read and execute permissions for group and others:

`chmod 755` *filename*

Special Permissions (`setuid`*,* `setgid` *and Sticky Bit)*

Three special types of permissions are available for executable files and public directories. When these permissions are set, any user who runs that executable file assumes the permissions of the owner (or group) of the executable file.

`setuid` *Permission*

When set-user identification (`setuid`) permission is set on an executable file, a process that runs this file is granted access based on the owner of the file (usually root), rather than the user who created the process. This allows a user to access files and directories that are normally only available to the owner. For example, the `setuid` permission on the `passwd` command makes it possible for a user to change passwords, assuming the permissions of the root ID:

```
-r-sr-sr-x   1 root     sys        10332 May  3 08:23 /usr/bin/passwd
```

This presents a security risk, because some determined users can find a way to maintain the permissions granted to them by the `setuid` process even after the process has finished executing.

`setgid` *Permission*

The set-group identification (`setgid`) permission is similar to `setuid`, except that the process's effective group ID (GID) is changed to the group owner of the file, and a user is granted access based on permissions granted to that group. The `/usr/bin/mail` program has setgid permissions:

```
-r-x--s--x   1 bin      mail       62504 May  3 07:58 /usr/bin/mail
```

When `setgid` permission is applied to a directory, files created in this directory belong to the group the directory belongs to, not the group the creating process belongs to. Any user who has write permission in the directory can create a file there—however, the file will not belong to the group of the user, but will belong to the group of the directory.

Sticky Bit

The *sticky bit* is a permission bit that protects the files within a directory. If the directory has the sticky bit set, a file can be deleted only by the owner of the file, the owner of the directory, or by root. This prevents a user from deleting other users' files from public directories such as `uucppublic`:

```
drwxrwxrwt  2 uucp    uucp    512 May 24 09:48 /var/spool/uucppublic
```

Be sure to set the sticky bit manually when you set up a public directory on a `tmpfs` filesystem.

Setting Special Permissions

You must be extremely careful when assigning these permissions, since setting these permissions constitutes a security risk. For example, a user can gain root permission by executing a program that sets the user ID to root.

For instructions on how to keep track of files with `setuid` permissions, turn to "How to Find Files With setuid Permissions Set" on page 36.

You should monitor your system to stay aware of any unauthorized use of the `setuid` and `setgid` permissions to gain root privileges. You can use the `find` command or the `ncheck` command to search the file systems and print out a list of all programs using these permissions. A suspicious listing would be one that grants ownership of such a program to a user rather than to `bin` or `sys`. Only the superuser can set these permissions. They are set using the extreme left position in the permission triplet:

special permission	read permission	write permission	execute permission
4=setuid 2=setgid 1=sticky bit	4	2	1

Examples

This `chmod` command sets `setuid` permission:

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x   1 db     staff        12095 May  6 09:29 dbprog
```

This `chmod` command sets `setgid` permission:

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x   1 db     dbstaff       24576 May  6 09:30 dbprog
```

This `chmod` command sets sticky bit permission:

```
# chmod 1777 pubdir
# ls -ld pubdir
-rwxrwxrwt   1 root   root           512 May  8 09:45 pubdir
```

Setting a Default `umask`

When you create a file or directory, it has a default set of permissions. These default permissions are determined by the value of `umask` in the system file `/etc/profile`, or in your `.cshrc` or `.login` file. By default, the system sets the permissions on a text file to `666`, granting read and write permission to user, group, and others, and to `777` on a directory or executable.

The value assigned by `umask` is subtracted from the default. This has the effect of denying permissions in the same way that `chmod` grants them. For example, while the command `chmod 022` grants write permission to group and others, `umask 022` denies write permission for group and others.

Table 3-6 shows some typical `umask` settings, and the effect on an executable file.

Table 3-6 `umask` Settings for Different Security Levels

Level of Security	umask	Disallows
Permissive (744)	022	w for group and others
Moderate (740)	027	w for group, rwx for others
Moderate (741)	026	w for group, rw for others
Severe (700)	077	rwx for group and others

Encrypting Files

Placing a sensitive file into an inaccessible directory (`700` mode) and making the file unreadable by others (`600` mode) will keep it secure in most cases. However, someone who guesses your password or the root password can read and write to that file. Also, the sensitive file is preserved on backup tapes every time you back up the system files to tape.

Fortunately, an additional layer of security is available to all SunOS system software users in the United States—the optional file encryption kit. The encryption kit includes the `crypt` command which scrambles the data to disguise the text.

Instructions for Securing Your Files

This section outlines the tasks that you perform to restrict access to the files and directories on your system.

▼ How to Display File Permissions and Ownership

♦ **Type** `ls -lg` **and press Return.**
The screen displays a long list of files in the directory. The `-l` option specifies a long listing, and the `-g` option lists the group. Each line in the display has the following information:

- Type of file, and its permissions
- Number of hard links
- Owner of the file
- Group of the file
- Size of the file, in bytes
- Date the file was created, or the last date it was changed
- Name of the file

Example of a Display of File Permissions and Ownership

The following example shows the result of the `ls -lg` command on the `/sbin` directory.

```
example% ls -lg
-r-xr-xr-x 1 bin      bin   122488 Dec 13 13:38    autopush*
-rwxr--r-- 1 root     sys     6916 Jan 1 1994      bcheckrc*
-rwxr-xr-x 1 bin      bin   523256 Dec 13 13:15    bpgetfile*
-r-xr-xr-x 1 bin      bin   483372 Dec 13 10:56    hostconfig*
-r-xr-xr-x 1 bin      bin   373700 Dec 13 14:15    ifconfig
-r-xr-xr-x 1 root     sys   187884 Dec 13 15:45    init*
-r-xr-xr-x 1 bin      staff 144808 Dec 13 15:38    mount*
-r-xr-xr-x 1 root     sys     5696 Jan 1 1994      mountall*
-rwxr--r-- 3 root     sys     2265 Jan 1 1994      rc0
-rwxr--r-- 1 root     sys     1018 Jan 1 197       rc1*
-rwxr--r-- 1 root     sys     1374 Jan 1 1994      rc2*
-rwxr--r-- 1 root     sys      713 Jan 1 1994      rc3*
-rwxr--r-- 3 root     sys     2265 Jan 1 1994      rc5*
-rwxr--r-- 3 root     sys     2265 Jan 1 1994      rc6*
-r-xr-xr-x 2 bin      root  192016 Sep 5 18:02     sh*
```

▼ How to Find Files With `setuid` Permissions Set

1. Type the following command:

```
find / -user root -perm -4000 -exec ls -ldb {}\; >/tmp/filename
```

This `find` command lists all `setuid` programs (those with a permission of `4000`) owned by root. The results are saved in a file in `/tmp`. All mounted paths are checked by this command starting at the root directory (`/`). Any surprises in the output should be investigated. Search time depends on the number of entries in the directory to be searched. This program can be run for `/sys`, `/bin`, and `/mail`, as well as `/`.

If you need background information about `setuid` and `setgid` permissions, turn to "Special Permissions (setuid, setgid and Sticky Bit)" on page 31.

2. Type `cat /tmp/`*filename* to see the results of the `find` command.

Example of Finding Files With `setuid` *Permissions Set*

In this example, an unauthorized user (`rar`) has made a personal copy of `/usr/bin/sh`, and has set the permissions as `setuid` to `root`. This means that `rar` can execute `/usr/rar/bin/sh` and become the privileged user.

```
# find / –user root –perm –4000 –exec ls –ldb { }\; > /tmp/ckprm
# cat /tmp/ckprm
-r-sr-xr-x 1 root bin 38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x 1 root bin 19812 Aug 10 16:16 /usr/bin/crontab
---s--x--x 1 root sys 46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x 1 root sys 12092 Aug 11 01:29 /usr/lib/mv_dir
-r-sr-sr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x 1 root bin 38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root rar 45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x 1 root bin 12524 Aug 11 01:27 /usr/bin/df
-rwsr-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-sr-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-sr-xr-x 1 root sys 23824 Aug 11 01:27 /usr/bin/su
#
```

If you want to save this output for future reference, move the file out of the `/tmp` directory.

▼ How to Create a Group for Users

1. **As root, type** `groupadd -g` *gid groupname* **and press Return**.
   This creates a new group called *groupname*.
2. **Type** `vi /etc/group` **and press Return.**
3. **Add the users to the group you have just created.**

Example of Creating a Group for Users

In the following example, users `charlie`, `msmith` and `jjones` are added to the `projects` group as their secondary group membership.

```
# groupadd -g 200 projects
# vi /etc/group
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
uucp::5:root,uucp
mail::6:root
tty::7:root,tty,adm
lp::8:root,lp,adm
nuucp::9:root,nuucp
staff::10:
daemon::12:root,daemon
sysadmin::14:
nobody::60001:
noaccess::60002:
users::100:
projects::200:charlie,msmith,jjones
```

▼ How to Change the Owner of a File or Directory

♦ **As root, type** chown *newowner filename* **and press Return.**
Only the current owner or root can change the owner of a file or directory.

```
example% chown msmith myfile
example% ls –l myfile
-rwxr-xr-x msmith 12985 Nov 12 16:28 myfile
```

▼ How to Change the Group of a File

♦ **Type** chgrp *group filename* **and press Return.**
Use the –g option to the ls command to list the group.

```
example% chgrp languages myfile
example% ls –lg myfile
-rwxrw–rw– 1 msmith languages 12985 Nov 12 16:28 myfile
```

▼ How to Set Permissions in Absolute Mode

When using chmod in absolute mode, the octal value assigned to the permission bits sets permissions for the owner, group, and others.

♦ **Type** chmod *nnn filename* **and press Return.**
The value of *n* has the following meanings:

If you need background information about setting permissions in absolute mode, turn to “Absolute Mode” on page 30.

Octal Value	File Permissions
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

Examples

To set `rwxr-xr-x` ***permissions:***

```
example% chmod 755 myfile
```

To set `rwsr-xr-x` ***permissions:***

```
example% chmod 4755 myfile
```

To set the sticky bit on a directory:

```
example% chmod 1777 pubdir
```

▼ How to Change Permissions in Symbolic Mode

When using `chmod` in symbolic mode, the symbols add, remove, or assign permissions to the owner, group, or all others.

For a description of all the symbols, turn to "Symbolic Mode" on page 29.

♦ **Type** `chmod` *who operator permissions filename*

Table 3-4 on page 29 lists the symbols that designate whose permissions are being set (*who*), the action being taken (*operator*), and the permissions being assigned (*permissions*).

Examples

To take away `read` ***permission from others:***

```
example% chmod o-r filename
```

To add `read` ***and*** `execute` ***permissions for user, group, and others:***

```
example% chmod a+rx filename
```

To assign `read`***,*** `write`***, and*** `execute` ***permissions to group:***

```
example% chmod g=rwx filename
```

Securing the Network 4

The first section of this chapter provides background information about the security issues associated with sending and receiving data across a network. Topics include firewall machines, remote logins, distributed file services, Secure NFS®, Kerberos, and Administration Tool.

If you want to go directly to the instructions, use the following table to find the page where instructions for a specific task begin.

4

About Network Security

The more available access is across a network, the more advantageous it is for the networked machines. However, free access and sharing of data and resources create security problems.

This chapter discusses network security, including ways to protect your network against unauthorized use, how to control access, and how distributed applications implement security on the network.

Protecting the Network With Firewall Machines

You can set up a firewall machine to protect the resources in your network from outside access.

A *firewall* machine is a secure host that acts as a barrier between your internal network and outside networks.

The firewall has two functions. It acts as a gateway which passes data between the networks, and it acts as a barrier which blocks the free passage of data to and from the network. The firewall requires a user on the internal network to log in to the firewall machine to access hosts on remote networks. Similarly, a user on an outside network must log in to the firewall machine before being granted access to a host on the internal network.

In addition, all electronic mail sent from the internal network is sent to the firewall machine for transfer to a host on an external network. The firewall machine receives all incoming electronic mail, and distributes it to the hosts on the internal network.

Caution – A firewall prevents unauthorized users from accessing hosts on your network. You must maintain strict and rigidly enforced security on the firewall, but since the other hosts on the network are protected, security on those systems can be more relaxed. However, an intruder who can break into your firewall machine can then gain access to all the other hosts on the internal network.

For information about trusted hosts, turn to "Trusted Hosts and Users" on page 44.

A firewall machine should not have any *trusted hosts*. (A trusted host is one from which a user can log in without being required to type in a password.) It should not share any of its file systems, or mount any file systems from other servers.

The Automated Security Enhancement Tool (ASET) can be used to make a machine into a firewall, and to enforce high security on a firewall machine. Chapter 5, “Monitoring and Controlling Security Using ASET,” describes the ASET utility.

Packet Smashing

Most local-area networks transmit data between computers in blocks called packets. Through a procedure called *packet smashing*, unauthorized users can harm or destroy data. Packet smashing involves capturing packets before they reach their destination, injecting arbitrary data into the contents, then sending the packets back on their original course. On a local-area network, packet smashing is impossible because packets reach all machines, including the server, at the same time. Packet smashing is possible on a gateway, however, so make sure all gateways on the network are protected.

The most dangerous attacks are those that affect the integrity of the data. Such attacks involve changing the contents of the packets or impersonating a user. Attacks that involve eavesdropping—recording conversations and replaying them later without impersonating a user—do not compromise data integrity. These attacks do affect privacy, however. You can protect the privacy of sensitive information by encrypting data that goes over the network.

Remote Logins

The `telnet` and `rlogin` programs enable users to log in to a remote machine over the network, and use the resources of the remote machine. The `rlogin` programs include `rsh` and `rcp`, which enable you to create a remote shell, and copy files to and from a remote machine.

`telnet` security problems arise because the packets sent across the network are broadcast to every computer physically connected to the Ethernet cable. Although a machine should be programmed to listen only to the packets addressed to that machine, a computer can be programmed to listen to and record all the packets sent across the network. It is possible to intercept the part of the login message that contains a user’s name and password. Since the password is sent in ASCII text, an intruder can use that information to assume the identity and permissions of that user.

The `rlogin` program automatically sends the user's name at the start of the transmission. If the remote login request is coming from a trusted host or a trusted user, the receiving machine allows the user to log in without requiring a password.

Trusted Hosts and Users

The administrator of a local system on the network can designate one or more remote hosts as "trusted" by listing them in the system file `/etc/hosts.equiv`. A user can then log in to and access the resources of a host on the network without being required to enter a password. In the same way, an administrator can designate trusted users on the `.rhosts` file.

The `/etc/hosts.equiv` ***File***

The `/etc/hosts.equiv` file contains a list of trusted hosts, one per line. If a user attempts to log in remotely (using `rlogin`) or to remotely execute a command (using `rsh`) from one of the hosts listed in this file, and if that user has an account on the local system with the same login name, the system allows the user to log in without a password.

A typical `hosts.equiv` file has the following structure:

```
host1
host2 user_a
+@group1
-@group2
```

When a simple entry for a host is made in `hosts.equiv`, such as the entry above for `host1`, it means that the host is trusted, and so is any user at that machine.

If the user name is also mentioned, as in the second entry in the example, then the host is trusted only if the mentioned user is attempting access.

A group name preceded by a plus sign (+) means that all the machines in that netgroup are considered trusted.

A group name preceded by a minus sign (–) means that none of the machines in that netgroup are considered trusted.

Caution – The `/etc/hosts/.equiv` file presents a security risk. If you maintain a `/etc/hosts.equiv` file on your system, you should include only trusted hosts in your network. The file should not include any host that belongs to a different network, or any machines that are in public areas. (For example, do not include a host that is located in a terminal room.)

Caution – A single line of `+` in the `hosts.equiv` file indicates that every known host is trusted.

This can create a serious security problem. Either replace the `/etc/hosts.equiv` file with a correctly configured one, or remove the file altogether.

The `.rhosts` *File*

The `.rhosts` file is the user equivalent of the `/etc/hosts.equiv` file. It contains a list of host-user combinations, rather than hosts in general. If a host-user combination is listed in this file, that user is granted permission to log in remotely from that host without having to supply a password.

Users can create `.rhosts` files in their home directories. Using the `.rhosts` file is another way to allow trusted access between their own accounts on different systems without using the `hosts.equiv` file.

Caution – Unfortunately, the `.rhosts` file presents a major security problem. While the `hosts.equiv` file is under the system administrator's control and can be managed effectively, any user may create an `.rhosts` file granting access to whomever the user chooses without the system administrator's knowledge.

For instructions, see "How to Search for and Remove .rhosts Files" on page 58.

The only secure way to manage `.rhosts` files is to completely disallow them. As system administrator, you can check the system often for violations of this policy. One possible exception to this policy is for the root account—you may need to have a `.rhosts` file to perform network backups and other remote services.

4

NFS Distributed Computing File System

The NFS system enables several hosts to share files over the network. Under the NFS system, a server holds the data and resources for several clients. The clients have access to the file systems that the server exports to the clients. Users logged in to the client machine can access the file systems by mounting them from the server. To the user on the client machine, it appears as if the files were local to the client. One of the most common uses of the NFS system is to allow systems to be installed in offices, while keeping all user files in a central location. Some features of the NFS system, such as the `nosuid` mount option, can be used to prohibit the opening of devices as well as file systems by unauthorized users.

The NFS system uses Secure RPC to authenticate users who make requests over the network. The authentication mechanism, `AUTH_DES`, use DES encryption to ensure authorized access.

Secure RPC

Secure RPC is a method of authentication that authenticates the host making a request, and the user. Secure RPC uses either DES or Kerberos authentication.

DES Encryption

The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt a secret key. If two credential users (or principals) know the same DES key, they can communicate in private, using the key to encipher and decipher text. DES is relatively fast encryption mechanism. A DES chip makes the encryption even faster; but if the chip is not present, a software implementation is substituted.

The risk of using just the DES key is that, with enough time, an intruder can collect enough cipher-text messages encrypted with the same key to be able to discover the key and decipher the messages. For this reason, security systems such as Secure NFS change the keys frequently.

The client and the server each has its own private key which they use together with the public key to devise a common key. They use the common key to communicate with each other, using an agreed-upon encryption / decryption function (such as DES).

DES Authentication

The DES method of authenticating a user that is very difficult for an intruder to crack. DES uses encryption and decryption functions that involve two keys, a public key and a private key (sometimes called a secret key).

Authentication is based on the ability of the sending system to use the common key to encrypt the current time, which the receiving system can decrypt and check against its current time. Make sure you synchronize the time on the client and the server.

See "How to Set Up an NIS+ Client to Use DES Security" on page 61 for instructions.

The public and private keys are stored in an NIS or NIS+ database. NIS stores the keys in the `publickey` map, and NIS+ stores the keys in the `cred` table. These files contain the public key and the private key for all potential users.

The system administrator is responsible for setting up NIS or NIS+ tables and generating a public key and a private key for each user. The private key is stored encrypted with the user's password. This makes the private key known only to the user.

See "How to Set Up NIS+ Security for a User or a Client" on page 58 for instructions.

The *NFS Administration Guide* manual describes how to set up and administer Secure NFS. Setting up the NIS+ tables and entering names in the `cred` table are discussed in *Name Services Administration Guide.* Turn to "Implementation of Secure RPC" on page 69 for an outline of the steps involved in RPC authentication.

Kerberos

See "How to Share and Mount Files With Kerberos Authentication" on page 62 for instructions.

Kerberos is an authentication system that was developed at the Massachusetts Institute of Technology. Kerberos uses DES encryption to authenticate a user when logging in to the system.

Kerberos works by authenticating the user's login password. A user enters the `kinit` command, which acquires a ticket that is valid for the time of the session (or eight hours, the default session time) from the Kerberos authentication server. When the user logs out, the ticket can be destroyed (using the `kdestroy` command).

See “How to Log In to Kerberos Service” on page 64 for instructions.

The Kerberos software is available from MIT project Athena, and is not part of the SunOS software. SunOS software provides:

- Routines used by the client to create, acquire, and verify tickets
- An authentication option to Secure RPC
- A client-side daemon, `kerbd`(1M)

“Implementation of Kerberos Authentication” on page 71 gives an overview of how the Kerberos authentication procedure works.

Note – Solaris provides the ability to connect to the Kerberos functionality. It does not provide the Kerberos package. However, you can `ftp` Kerberos 4 source from `athena-dist.mit.edu` using `anonymous` as a username and your email address as a password. The source is located in `pub/kerberos`.

Alternative to Secure RPC

If you do not want to run Secure RPC, a possible substitute is the Solaris “privileged port” mechanism. A privileged port is built up by the root with a port number of less than 1024. After a client system has authenticated the client’s credential, it builds a connection to the server via the privileged port. The server then verifies the client credential by examining the connection’s port number.

Non-Solaris clients however may not be able to communicate via the privileged port. If they cannot, you will see error messages such as these:

```
“Weak Authentication
 NFS request from unprivileged port”
```

For information on how to turn on the privileged port mechanism, see *NFS Administration Guide.*

Access Control

See “How to Share and Mount Files With DES Authentication” on page 62, and “How to Share and Mount Files With Kerberos Authentication” on page 62, for instructions.

A network file server can control which files are available for sharing. It can also control which clients have access to the files, and what type of access is permitted to those clients. In general, the file server can grant read / write or read-only access either to all clients or to specific clients. Access control is specified when resources are made available with the `share` command.

A server can use the /etc/dfs/dfstab file to list the file systems it makes available to clients on the network.

The /etc/dfs/dfstab *File*

The commands in the dfstab file are run by rc scripts when the system enters multiuser mode. You can put a share command into this file that limits the access of a client to a particular file system. This is an example of an /etc/dfs/dfstab file.

```
# place share(1M) commands here for automatic execution
# on entering init state 3.
#
# share [-F fstype][ -o options][-d "<text>"] <pathname> <resource>
# .e.g,
# share -F rfs -d "/var/news" /var/news NEWS
share -F nfs -o ro /usr/share/man
```

Restricting Root Access

In general, a superuser is not allowed root access to file systems shared across the network. Unless the server specifically grants superuser privileges, a user who is logged in as root on a client cannot gain root access to files that are remotely mounted on the client. The NFS system implements this by changing the user ID of the requester to the user ID of the user name, nobody; this is generally 60001. The access rights of user nobody are the same as those given to the public (or a user without credentials) for a particular file. For example, if the public has only execute permission for a file, then user nobody can only execute that file.

An NFS server can grant superuser privileges on a shared file system on a per-host basis, using the root=*hostname* option to the share command.

Administration Tool

Administration Tool (accessed with the admintool command) enables a user to perform system administration tasks across the network. For example, an administrator can use the applications in this tool to add a new system to the network, add and set up a printer, or set up a new user account.

Administration Tool uses the underlying support of the *distributed administrative framework*, which enables a user to execute processes on remote systems.

Caution – It is important to control access to Administration Tool, since anyone with the right access can use Administration Tool to change the configuration of the systems on the network. It is also important to be sure that the network security is set up in a logical and consistent manner so that access is available to users who require it and unavailable to users who should not have it.

Administration Tool is designed to work in a distributed system. In a distributed system, some parts of the application run on the user's system and some parts run on a remote system, which could be a system server or another user's system. It is even possible for these two systems to be the same, as for example, if a user is logged on to the server. The Administration Tool application determines what modifications are taking place and which system is being modified.

Distributed Administration Framework Security

The security mechanism used by Administration Tool is the same underlying mechanism supported by the distributed administration framework.

Security is administered in three stages: authentication, authorization, and setting user and group identities.

Authentication

Authentication means that the security mechanism verifies the identity of the user making the request. When executing a request, the `admind` daemon must authenticate the client identity to the server.

Authorization

Authorization means that the security mechanism checks if the authenticated user has permission to execute the Administration Tool function on the server. Once the client identity is verified, `admind` uses this identity to perform authorization checks.

An additional level of authorization is used by NIS+ to allow or deny access to the NIS+ tables. You may have permission to use Administration Tool, but you also need to have create, delete, or modify permission to change an NIS+ database. See *Name Services Administration Guide* for a description of NIS+ security.

User and group identities are used for authorization checking as follows:

- *Root identity* – The root identity is allowed root privileges only on the local system (to access and update data on local files and databases). On the server, the security mechanism changes the UID of all requests that originate as root from a remote client to the user `nobody`. If the server is the local system (in other words, if the user has logged in as root on the server), the user will be allowed to perform Administration Tool functions on the server under the root identity.
- *User ID* – An ordinary user can only retrieve information, but cannot use Administration Tool to perform any tasks that modify administration data.
- *User ID who is a member of sysadmin group* (`GID=14`) – Administration Tool permissions are granted to users who are members of the sysadmin group (`GID=14`). This means that a user performing a task that modifies administration data on a system—using Administration Tool to add a new user account, for example—must be a member of the sysadmin group on the system where the task is being executed.

How Administration Tool Sets User and Group Identities

The security mechanisms are responsible for making sure that the user and group identities are set correctly for the task that needs to be executed. If the security mechanism has authenticated the client making the request, and has verified that the client has the authority to perform the administrative task, Administration Tool will set the UIDs and GIDs so that the client can perform the task. In some cases the root identity is set in order to perform the task; at other times, the client's identity is set.

Security Levels

Administration Tool uses the distributed administration framework daemon (`admind`) to carry out the security tasks. The `admind` daemon process executes the request on the server on behalf of the client process.

Each request contains a set of credentials with a user ID (UID) and a set of group IDs (GIDs) to which the UID belongs. The server uses these credentials to perform identity and permission checks. Three levels of authentication security are available:

- *Level 0* (`AUTH_NONE`) – No identity checking is done. All user IDs are set to the `nobody` identity. This level is used mostly for testing.
- *Level 1* (`AUTH_SYS`) – The server accepts the original user and group identities directly from the client system and uses them as the identities for the authorization checks. The server does not check that the UID of the client represents the same user on the server system. It is assumed the administrator has made the user IDs and group IDs consistent on all systems in the network. Checks are made to see if the client has permission to execute the request.
- *Level 2* (`AUTH_DES`) – Credentials are validated using DES authentication, and the server checks that the client has permission to execute the request. The user and group identities are obtained from databases on the server system by mapping the user's DES network identity (the DES entry in the NIS+ Cred table, for example) to a local UID and set of GIDs. The database used depends on which name service is selected on the server system. This level provides the most secure environment for performing administrative tasks and requires that a `publickey` entry exists for all server systems where the `admind` daemon is running, and for all users accessing the tools.

For instructions on how to edit the `/etc/inet.conf` file to change the `admind` security level, turn to "How to Set Up Security for Administration Tool" on page 66.

Administration Tool Security uses the Level 1 authentication (`AUTH_SYS`) as the default. You can tighten security to require Level 2 security checks by editing the `/etc/inetd.conf` file on each system, and adding the `-S 2` option to the `admind` entry. Make sure that the servers on the domain are set up to use DES security.

You do not need to maintain the same level of security on all systems in the network. You can run some systems, such as file servers requiring strict security, at security Level 2, while running other systems, such as systems, at

the default Level 1 security. The distributed administration framework automatically determines the right authentication mechanism to use for accessing each system based on its security level.

See “How to Set Up NIS+ Security for a User or a Client” on page 58, and “How to Set Up an NIS+ Client to Use DES Security” on page 61. See also the description of how to set security for NIS+ in *Name Services Administration Guide*.

Name Service Information

Administration Tool `admind` uses information held by the name service. The three sources of information are:

- Files in the `/etc` directory such as `passwd`, `group`, and `shadow`, referred to by the keyword `files`
- The NIS name service referred to by the keyword `nis`
- The NIS+ name service referred to by the keyword `nisplus`

On each system, the `/etc/nsswitch.conf` file lists administrative databases, followed by a list of one or more name services that will be searched for information. If more than one name service is listed, they are searched in the order given. For example, the following entry

```
group:   files nisplus
```

indicates that the `admind` looks first in the local `/etc/group` file for an entry. If the entry exists, it uses the information in this entry. If it doesn’t exist, the NIS+ `group` table is searched.

In the case of Administration Tool, the `/etc/group` file is searched for an entry for sysadmin group (`GID=14`). If the entry exists, it uses the information listed there, and does not check the NIS+ `group` table. If you want to set up your system to use network-wide information, remove the sysadmin group from the local system.

The example below is a `/etc/nsswitch.conf` file that lists `files` and `nisplus` as the two name services to search.

```
#
# /etc/nsswitch.conf:
#
# “hosts:” and “services:” in this file are used only if the
/etc/netconfig # file contains “switch.so” as a nametoaddr library for
“inet” transports.

# the following two lines obviate the “+” entry in /etc/passwd and
/etc/group.
passwd:      files nisplus
group:       files nisplus

# consult /etc “files” only if nisplus is down.
hosts:       nisplus [NOTFOUND=return] files

services:    nisplus [NOTFOUND=return] files
networks:    nisplus [NOTFOUND=return] files
protocols:   nisplus [NOTFOUND=return] files
rpc:         nisplus [NOTFOUND=return] files
ethers:      nisplus [NOTFOUND=return] files
netmasks:    nisplus [NOTFOUND=return] files
bootparams   nisplus [NOTFOUND=return] files
:

publickey:   nisplus

netgroup:    nisplus

automount:   files nisplus
aliases:     files nisplus
```

The following entry indicates that the `/etc` files should be searched only if the NIS+ network name service cannot be reached:

```
hosts:      nisplus [NOTFOUND=return] files
```

If the NIS+ databases are available, and if the entry is not found, then the `/etc` files are not searched.

This entry lists only one name service for public and private key information:

```
publickey:  nisplus
```

For more information about public and private keys, turn to “Secure RPC” on page 46.

Note – When running under Level 2 security, the `admind` uses the public/private key information. Make sure that the entry for `publickey` is followed by either `nis` or `nisplus` (depending on which name service you are using), and remove the `files` designation.

Creating a Security Policy for Administration Tool

You can use the security mechanisms discussed in this section to create a security policy for using Administration Tool in the network. This security policy should be consistent with the policy chosen for the name service being used.

Determine How Much Trust Is Needed

If your network is secure and you do not need to use elaborate authentication security, you can use the Administration Tool applications using the default Level 1 security.

For instructions on how to edit your `/etc/inet.conf` file to change the security level, turn to "How to Set Up Security for Administration Tool" on page 66.

If you need to enforce a higher level of security, you can set the security level of `admind` to Level 2.

For Level 1 security, the client system uses the client's ID and GID as established when the user logged in (or when an `su` command was issued). The distributed administration framework passes the client's UID and GID identities to the `admind` daemon on the server system, which accepts these identities without checking that they define the same user on the server system. You must align the `passwd` and `group` files on each system in the network to ensure users have the same identities on all systems. This is done most easily when using a name service like NIS+.

For Level 2 security, the client system maps the user's identity as established when the user logged in to a globally unique network identity, the `netid`. The distributed administration framework uses Secure RPC's DES authentication mechanism to do this mapping, and to pass the `netid` identity to the `admind` daemon on the server system within an encrypted credential. The `admind` daemon uses the DES authentication mechanism to decrypt the credential and map the `netid` identity to a UID and list of GIDs which represent the same user on the server system. The databases used on each system are determined

by which name service is being used on that system. Because of the way the maps are established and the public keys generated, Level 2 security is primarily used with NIS+ name services.

Determine Which Name Service Will Be Used

The name services determine where the security methods get information about user and group identities. The name services are designated in the `/etc/nsswitch.conf` file (see "Name Service Information" on page 53).

Decide Which Users Have Access to Administration Tool

Decide which users will use Administration Tool to perform administrative functions over the network. List these users as members of the sysadmin group accessed by the server system, as defined in the `/etc/nsswitch.conf` configuration file. The sysadmin group must be accessible from each system in the network upon which administration data will be updated by Administration Tool. The sysadmin group can be established locally on each system or can be used globally within a name service domain, depending upon the policy established by the administrator. By default, the Solaris system software creates the `sysadmin` group with a group ID of 14. Adding users to this group automatically gives them access to Administration Tool.

Determine Global and Local Policies

The *global policy* affects all hosts in the network. For example, if you add users to the sysadmin group NIS or NIS+ `group` files, members of this group can perform administrative tasks on all systems that list the network name service as the primary source of information. The name services are listed in the `/etc/nsswitch.conf` file.

For information about the `nsswitch.conf` file, turn to "Name Service Information" on page 53.

For example, a domain may have an NIS+ server that administers a domain using the NIS+ name service. The NIS+ `group` table contains an entry for the sysadmin group listing `allison`, `beverly`, and `charles` as members of this group. These three administrators have permissions on all systems in the network that list `nisplus` as the name service.

A user can choose to establish a *local policy* that is different from the global policy by adding users to the sysadmin entry to the local `/etc/group` file. The members of this group will have permission to use Administration Tool applications on the user's local system.

For example, to allow access only to beverly and david:

```
sysadmin:14:beverly,david
```

Set Up Permissions for NIS+ Databases

You need the proper permissions when using Administration Tool to modify or update the NIS+ databases. In addition to the permissions required by Administration Tool, the NIS+ security mechanisms impose their own set of access permissions. The NIS+ security mechanisms are described in *Name Services Administration Guide.*

Set Up Initial Security

By default, Solaris system software assigns the group ID 14 to the sysadmin group. Add users to this group to give them access to Administration Tool applications.

Use Database Manager tool to add users to the sysadmin group in the NIS+ group table. The appropriate global policy will allow members of this group to perform administration tasks on all systems in the NIS+ domain.

See *Administration Application Reference Manual* for information on using the Administration Tool applications.

`ttyhstmgr` *Security*

`ttyhstmgr` is a tool that allows an administrator to add or remove a host information when they do not have access to a window system without using Administration Tool. It uses the same security mechanisms as Administration Tool. That is, only members of sysadmin group (GID=14) have permission to manage the associations between servers and clients on a network with `ttyhstmgr`.

4

Instructions for Administering Network Security

A system administrator can implement policies that help secure the network. The level of security required will differ with each site. This section provides instructions for some tasks associated with network security.

▼ How to Search for and Remove `.rhosts` Files

As system administrator, you may want to search for and remove `.rhosts` files on a system, as they can pose a security risk.

1. **Type** `find` */directory* `-name .rhosts -print` **and press Return.**
   The `find` command starts at the designated directory and searches for any file named `.rhosts`. If it finds any, it prints the path name on the screen.
2. **Type** `rm` */directory/username/*`.rhosts` **and press Return.**
   The most secure option is to remove these files.

For background information, see "The .rhosts File" on page 45.

Example of Finding and Removing `.rhosts` *Files*

The following example searches for `.rhost` files starting from the `/usr/users` directory, printing any files it finds on the screen.

```
# find /export/home -name .rhosts -print
/export/home/jjones/.rhosts
# rm /export/home/jjones/.rhosts
#
```

▼ How to Set Up NIS+ Security for a User or a Client

For detailed description of NIS+ security, see *Name Services Administration Guide.*

To set up secure NIS+ for root on a client:

1. **As root, edit the** `/etc/nsswitch.conf` **file and add the following line:**
   `publickey: nisplus`
2. **Type** `nisinit -cH` *hostname* **and press Return.**
   hostname is the name of a trusted NIS+ server that contains an entry in its tables for the client machine.

For background information, see "NFS Distributed Computing File System" on page 46.

3. **Add the client to the** `cred` **table. Type the following commands:**
   `nisaddcred local`
   `nisaddcred des`

4. **To verify the setup, type** `keylogin` **and press Return.**
   If you are prompted for a password, the procedure has succeeded.

Example of Setting Up an NIS+ Client With DES Security

The following example uses the host `pluto` to set up `earth` as an NIS+ client. You can ignore the warnings. The `keylogin` command is accepted, verifying that `earth` is correctly set up as a secure NIS+ client.

```
earth% su
# nisinit -cH pluto
NIS Server/Client setup utility.
This machine is in the North.Abc.COM. directory.
Setting up NIS+ client ...
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@North.Abc.COM
Adding new key for unix.earth@North.Abc.Com (earth.North.Abc.COM.)

Network password: xxx <Press Return>
Warning, password differs from login password.
Retype password: xxx <Press Return>

# keylogin
Password:
#
```

To set up secure NIS+ for a user:

1. **Add the user to the** `cred` **table on the root master server. Type the following command:**
   `nisaddcred –p unix.`*UID@domainname* `–P` *username.domainname.* `des`

   Note that, in this case, the *username-domainname* must end with a dot (.)

2. **To verify the setup, login as the client and type** `keylogin` **and press Return.**

Example of Adding a User to Secure NIS+

The following example gives DES security authorization to user `george`.

```
# nisaddcred -p unix.1234@North.Abc.com -P george.North.Abc.COM. des
DES principal name : unix.1234@North.Abc.COM
Adding new key for unix.1234@North.Abc.COM (george.North.Abc.COM.)

Password:
Retype password:

# rlogin rootmaster -l george
# keylogin
Password:
#
```

▼ How to Set Up an NIS+ Client to Use DES Security

For reference information, turn to "Implementation of Secure RPC" on page 69.

To create a new key for root on a client:

1. **Become root on the client.**
2. **Type** `newkey -h` *hostname* **and press Return.**
   hostname is the name of the client.

Example of Setting Up an NIS+ Client to Use DES Security

The following example sets up `earth` as a secure NIS client.

```
earth% su
# newkey -h earth
Adding new key for unix.earth@North.Abc.COM
New Password:
Retype password:
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

To create a new key for a user:

1. **Log in to the server as root.**
   Only the system administrator, logged in to the NIS+ server, can generate a new key for a user.
2. **Type** `newkey -u` *username* **and press Return.**
   username is the name of the user. The system prompts for a password. The system administrator can type a generic password. The private key is stored encrypted with the generic password.

```
example% su
# newkey -u george
Adding new key for unix.12345@Abc.North.Acme.COM
New Password:
Retype password:
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

3. **Tell the user to log in and type the** `chkey` **command.**
   This allows the user to re-encrypt their private key with a password known only to the user.

```
earth% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@Abc.North.Acme.COM
Please enter the Secure-RPC password for usr1:
Please enter the login password for usr1:
Sending key change request to pluto...
#
```

Note – The `chkey` command can be used to create a new key-pair for a user.

▼ How to Share and Mount Files With DES Authentication

Prerequisite

The DES `publickey` authentication must be enabled on the network. See "How to Set Up NIS+ Security for a User or a Client" on page 58, and "How to Set Up an NIS+ Client to Use DES Security" on page 61.

To share a file system with DES authentication:

♦ **Type** `share -F nfs -o secure` */filesystem* **and press Return.**

For background information about DES authentication, turn to "Secure RPC" on page 46.

To mount a file system with DES authentication:

♦ **Type** `mount –F nfs -o secure` *server:resource mountpoint* **and press Return.**
The `–o secure` option mounts the file system with `AUTH_DES` authentication.

▼ How to Share and Mount Files With Kerberos Authentication

Prerequisite

Kerberos authentication must be enabled on the network.

To share a file system with Kerberos authentication:

♦ **Type** `share -F nfs -o kerberos` */filesystem* **and press Return.**

For background information about Kerberos authentication, turn to “Kerberos” on page 47.

To mount a file system with Kerberos authentication:

♦ **Type the following command:**
`mount –F nfs -o kerberos` *server:resource mountpoint*

The `–o kerberos` option mounts the file system with `AUTH_KERB` authentication.

▼ How to Acquire a Kerberos Ticket for Root on a Client

If the NFS file system that you need to access has not been mounted, you need to acquire a ticket for root on the client before mounting it.

To acquire a ticket for a not-yet-mounted file system:

1. **Type** `su` **and press Return.**
2. **Type** `kinit root.`*hostname* **and press Return.**
   hostname is the name of the client system.

```
earth% su
# kinit root.earth
Password:
#
```

To acquire a ticket for a mounted file system:

If the entry `root.`*hostname* for the client has been entered into the `/etc/srvtab` configuration file, you can use the `ksrvtgt` command to get a ticket for root. In this case, you are not required to give a root password. Consult the MIT documentation for information about initializing the `/etc/srvtab` file.

♦ **As root, type** `ksrvtgt root.`*hostname* **and press Return.**

```
earth% su
#ksrvtgt root.earth
#
```

▼ How to Log In to Kerberos Service

♦ **Type** `kinit -l` *username* **and press Return.**
`kinit` prompts you for the ticket lifetime (`-l` option), and your password. It prints out ticket status using the verbose mode (`-v` option).

Example of Logging In to Kerberos Service

```
earth% kinit -l jjones
SunOS (earth)
Kerberos Initialization for "jjones"
Kerberos ticket lifetime (minutes): 480
Password:
earth%
```

▼ How to List Kerberos Tickets

♦ **Type** `klist` **and press Return.**

Example of Listing Kerberos Tickets

```
earth% klist
Ticket file: /tmp/tkt8516
Principal: jjones@North.Abc.COM
  Issued            Expires          Principal
  Jan 14 20:40:54   Jan
15:04:40:54  krbtgt.North.Abc.COM@North.Abc.COM
```

▼ How to Access a Directory With Kerberos Authentication

♦ **Type** `cd` */mountpoint* **and press Return.**
Access the mounted directory, just as you would any other mounted directory. You can list the files in the directory with the `ls` command, or list the Kerberos tickets with the `klist` command.

Example of Accessing a Directory With Kerberos Authentication

In the following example, user `jjones` can change to the mounted directory `mntkrb`, and list the files in this directory.

The `kerbd` daemon has automatically secured a ticket on the user's behalf for the NFS server exporting the file system. At this point there are two valid tickets—the original ticket-granting ticket, and the server ticket. These two tickets are listed by `klist`.

```
earth% cd /mntkrb
earth% ls -l /mntkrb
-rw-r--r-- 1 marks staff 29 Jul 1994 sports
drwxr-xr-x 3 jjones staff 512 Sep 13 13:44 market

earth% klist
Ticket file: /tmp/tkt8516
Principal: jjones@North.Abc.COM
  Issued            Expires          Principal
  Jan 14 20:40:54   Jan
15:04:40:54  krbtgt.North.Abc.COM@North.Abc.COM
  Jan 14 20:43:21   Jan 15:04:43:21  nfs.pluto@North.Abc.COM
```

▼ How to Destroy a Kerberos Ticket

♦ **Type** `kdestroy` **and press Return.**
Destroy Kerberos tickets when the session is over, so that an unauthorized user cannot to gain access to it. If you want to reinitiate Kerberos authentication, use the `kinit` command.

Example of Destroying a Kerberos Ticket

The following example shows how to destroy the Kerberos ticket. If the user then tries to change to or list a Kerberos-protected directory, the ticket server denies access.

```
earth% kdestroy
Tickets destroyed
earth% ls /mntkrb
Can't get Kerberos key: No ticket file (tf_util)
NFS getattr failed for server pluto: RPC: Authentication error
can not access directory /mntkrb.
```

▼ How to Set Up Security for Administration Tool

To use Administration Tool to add or update administration data on a remote system, a user must be a member of the sysadmin group with a group ID of 14 (GID=14). Use the following procedure to add users to this group.

For information about the sysadmin group, turn to "Name Service Information" on page 53.

To set up security using NIS+ name service:

1. **Log on the NIS+ master server as root.**
2. **Start the Administration Tool Database Manager and open the** `group` **table.**
3. **Select the group table.**

4. **Add a comma-separated list of users to the Members List:**
   Group Name: `sysadmin`
   Group ID: `14`
   Members List: *usera,userb,userc*

   usera, *userb*, and *userc* will have permission to update data with Administration Tool.

   See *Administration Application Reference Manual* for details about how to set up Administration Tool using the NIS+ name service.

To set up security using NIS name service:

1. **Log in to the NIS master server as root.**
2. **Edit the `/etc/group` file, adding users to the sysadmin group:**
   `sysadmin:*:14:`*usera,userb,userc*

   usera, *userb*, and *userc* will have permission to update data with Administration Tool.
3. **Change directory to `/var/yp`.**
4. **Type `make group` and press Return.**
   This updates the group map on the NIS master server and sends the updated map to the NIS clients on the network.

▼ How to Set Up DES Authentication for Administration Tool

For instructions on how to set up a client with DES authentication, turn to "How to Set Up NIS+ Security for a User or a Client" on page 58, or "How to Set Up an NIS+ Client to Use DES Security" on page 61.

You can increase security for administration tasks by requiring DES authentication for Administration Tool. If you require DES authentication for Administration Tool, you need to set up the systems in the network with `publickey` security mechanisms.

Prerequisite

All NIS+ principals who will use Administration Tool must have DES credentials in the `cred` table. This includes all hosts that are Administration Tool server systems and all users who are members of the sysadmin group.

For background information about Level 2 security, turn to "Administration Tool" on page 49.

1. **As root, edit the** `/etc/inet/inetd.conf` **file.**
   Add the `-S 2` option to the `admind` entry:

```
100087/10 tli rpc/udp wait root /usr/sbin/admind admind -S 2
```

   This entry changes the security level to Level 2, which requires DES authentication.

The `/etc/nsswitch.conf` file is described in "Name Service Information" on page 53.

2. **Check the entry for** `publickey` **in the file** `/etc/nsswitch.conf`**.**
   Make sure that the source of the information for `publickey` is the network name service `NIS+`. The entry should be:

```
publickey: nisplus
```

Note – You will need to re-establish credentials for users who are added to the sysadmin group after their `publickey` entries were established in NIS+. This is because the Cred table contains a copy of the user's groups from the group database. The DES authentication mechanism uses the information in the `cred` table to set group identities on the server. Use the `nisaddcred` command to get the new group information in the `cred` table. (See "How to Set Up NIS+ Security for a User or a Client" on page 58.)

Reference Material for Administering Network Security

This section describes how Secure RPC and Kerberos provide authentication for users and clients accessing files on a network.

Implementation of Secure RPC

The *NFS Administration Guide* manual describes the procedure involved in implementing Secure RPC on a system. The process is outlined briefly here.

1. The system administrator runs a program that generates a public key and a secret key (or private key) for each user on the system.

   The keys are stored in a public database, `/etc/publickey`. This database contains the public key and an encrypted secret key for all principals. A principal is a client user or a client system who credentials have been stored in the NIS+ domain. A user's secret key is stored encrypted with the user's password.

2. The keyserver program (`keyserv`) must be running on both the client machine and the server.

   The keyserver is an RPC service that runs locally on every machine. The keyserver saves the decrypted secret key, and waits for the user to initiate a transfer with a server.

3. The `keylogin` program runs whenever a user logs in.

   The `keylogin` program runs automatically at login time, if it is included in the `/etc/profile` file. The login process gets the secret key from `/etc/publickey`, decrypts it using the user's password, and passes the decrypted secret key to the keyserver. This scheme relies on the user's choice of an appropriate password.

4. The user starts an initial transaction with a server.

   The keyserver uses the client's secret key and the server's public key to create a common key. In addition, the keyserver generates a random conversation key, which is encrypted using public key cryptography.

5. The client sends the transmission, including a time stamp and the conversation key, to the server.

The initial transmission includes a credential and a verifier. The credential contains three parts: the client user's `netname`, the conversation key, and a window. The conversation key is encrypted with the common key, and the window is encrypted with the conversation key.

The window is the difference that the client allows between the server's clock and the client's time stamp. If the difference between the server's clock and the time stamp is greater than the window, the server should reject the client's request. (For secure NFS, the window currently defaults to 30 minutes.)

The client's verifier contains the encrypted time stamp, and an encrypted verifier of the specified window, incremented by one. Using a window verifier makes it more difficult for an intruder to guess the credential.

6. The server receives the transmission from the client.

   The keyserver uses the client's public key and the server's secret key to generate the common key—the same common key computed by the client. (Only the server and the client can calculate the common key, since doing so requires knowing one secret key or the other.)

   The server decrypts the client's time stamp with the decrypted conversation key. Authentication is based on the sender's ability to encrypt the current time, which the server decrypts and checks against its current time. If it's close, then the sender must have encrypted it correctly. The server also checks if the time stamp is greater than the one previously seen from the client.

   The sender must base its time stamp on the correct time of the receiver, so that the receiver can check for replayed messages. It is important to synchronize the time on the server and the client.

7. The server stores four items in a credential table:
   - The client user's `netname`
   - The conversation key
   - The window
   - The client's time stamp

   The server stores the first three items for future use. It stores the time stamp to protect against replays. The server will accept only time stamps that are chronologically greater than the last one seen, so that any replayed transactions are automatically rejected.

8. The server returns a verifier to the client.

   The verifier includes the index ID, which the server records in its credential table, and the client's time stamp minus one, encrypted by the conversation key. One is subtracted from the time stamp to ensure that it is invalid, and cannot be reused as a client verifier.

9. The client receives the verifier and authenticates the server.

   The server doesn't need to send a credential for verification, since the client called the server in the first place. However, the client needs to authenticate that the reply is really from the server. Only the server can send the correct time stamp, since only the server knows what time stamp the client sent.

10. The client returns the index ID to the server in its second transaction and sends another encrypted time stamp.

11. The server sends back the client's time stamp minus one, encrypted by the conversation key.

    With each transaction after the first, the client sends back its index ID and another encrypted time stamp, and the server returns the time stamp minus one.

Implementation of Kerberos Authentication

For additional information about Kerberos authentication, turn to "Kerberos" on page 47.

The following procedure assumes that the Kerberos key distribution center (KDC) is already installed on the network, using publicly available sources from MIT project Athena.

The `/usr/sbin/kerbd` daemon must be running on the NFS client and server.

This daemon is normally started automatically at system boot time by the `/etc/init.d/rpc` startup script. `kerbd` is the user-mode daemon. It interfaces with the kernel RPC and the KDC. It generates and validates authentication tickets.

You can use the `ps` command to see if the daemon is running.

```
earth% ps -ef |grep kerbd
```

1. The system administrator sets up the NFS server to use Kerberos authentication.

   The MIT Kerberos software is used to register the principal names in the Kerberos key distribution center (KDC) on the Kerberos server. The following entries are required:
   - `root.`*hostname* (required for each NFS client)
   - `nfs.`*hostname* (required for each NFS server)

2. The user mounts the shared file system.

   The user on the client must get a ticket for root on the client to mount the shared file system.

3. The user logs in to the Kerberos service, using the `kinit` command.

   The Kerberos authentication server authenticates the request, and grants a ticket for the ticket-granting service.

4. The user accesses the mounted directory.

   The `kerbd` daemon automatically secures a ticket on behalf of the client for the NFS server exporting the file system. At this point, there are two valid tickets, the original ticket-granting ticket and one for the server.

5. The user destroys the tickets at the end of the session to prevent them from being compromised.

   The `kdestroy` command destroys the user's active Kerberos authorization tickets by writing zeros to the file that contains the tickets. You can put the `kdestroy` command in your `.logout` file, so that all Kerberos tickets are automatically destroyed when you log out of the system.

6. If tickets have been destroyed before the session has finished, the user must request a new ticket with the `kinit` command.

Monitoring and Controlling Security Using ASET 5

SunOS system software includes the Automated Security Enhancement Tool (ASET). ASET helps you monitor and control system security by automatically performing tasks that you would otherwise do manually.

This chapter describes how ASET works and tells you how to customize the ASET functions to suit your environment. If you are familiar with ASET, use the following table to go directly to the task you want to perform.

5

About ASET

The ASET security package provides automated administration tools that enable you to control and monitor your system's security. You specify a security level—low, medium, or high—at which ASET will run. At each higher level, ASET's file-control functions increase to reduce file access and tighten your system security.

There are seven tasks involved with ASET, each performing specific checks and adjustments to system files. The ASET tasks tighten file permissions, check the contents of critical system files for security weaknesses, and monitor crucial areas. ASET can safeguard a network by applying the basic requirements of a firewall machine to a system that serves as a gateway machine. (See "Firewall Setup" on page 78.)

ASET uses master files for configuration. Master files, reports, and other ASET files are in the directory `/usr/aset`. These files can be changed to suit the particular requirements of your site.

If you need information about ASET files, see "ASET Reports" on page 78, and "Master Files" on page 83, or consult the reference page for `aset(1M)`.

Each task generates a report noting detected security weaknesses and changes the task has made to the system files. When run at the highest security level, ASET will attempt to modify all system security weaknesses. If it cannot correct a potential security problem, ASET reports the existence of the problem.

You can initiate an ASET session by typing the following command in a command shell:

```
% aset
```

You can also set up ASET to run periodically by putting an entry into the `crontab` file.

If you need information about running ASET periodically, see "Schedule ASET Execution: PERIODIC_SCHEDULE" on page 86.

ASET tasks are disk-intensive and can interfere with regular activities. To minimize the impact on system performance, schedule ASET to run when system activity level is lowest, for example, once every 24 or 48 hours at midnight.

These topics are discussed in more detail on the following pages:

- ASET security levels
- ASET tasks
- ASET reports

- ASET files
- Configuring ASET
- Restoring files modified by ASET
- Network operation using the NFS system

ASET Security Levels

ASET can be set to operate at one of three security levels: low, medium, or high. At each higher level, ASET's file-control functions increase to reduce file access and heighten system security. These functions range from monitoring system security without limiting users' file access, to increasingly tightening access permissions until the system is fully secured.

The three levels are outlined below:

- *Low security* – This level ensures that attributes of system files are set to standard release values. ASET performs several checks and reports potential security weaknesses. At this level, ASET takes no action and does not affect system services.
- *Medium security* – This level provides adequate security control for most environments. ASET modifies some of the settings of system files and parameters, restricting system access to reduce the risks from security attacks. ASET reports security weaknesses and any modifications it makes to restrict access. At this level, ASET does not affect system services.
- *High security* – This level renders a highly secure system. ASET adjusts many system files and parameter settings to minimum access permissions. Most system applications and commands continue to function normally, but at this level, security considerations take precedence over other system behavior.

Note – ASET does not change the permissions of a file to make it less secure, unless you downgrade the security level or intentionally revert the system to the settings that existed prior to running ASET.

ASET Tasks

This section discusses what ASET does. You should understand each ASET task—what its objectives are, what operations it performs, and what system components it affects—to interpret and use the reports effectively.

If you need information about report files, turn to “ASET Reports” on page 78.

ASET report files contain messages that describe as specifically as possible any problems discovered by each ASET task. These messages can help you diagnose and correct these problems. However, successful use of ASET assumes that you possess a general understanding of system administration and system components. If you are a new administrator, you can refer to other SunOS system administration documentation and related manual pages to prepare yourself for ASET administration.

The `taskstat` utility identifies the tasks that have been completed and the ones that are still running. Each completed task produces a report file. For a complete description of the `taskstat` utility, refer to the `taskstat(1M)` manual page.

System Files Permissions Verification

“Tune Files” on page 98 shows an example of the files that ASET consults when setting permissions.

This task sets the permissions on system files to the security level you designate. It is run when the system is installed. If you decide later to alter the previously established levels, run this task again. At low security, the permissions are set to values that are appropriate for an open information-sharing environment. At medium security, the permissions are tightened to produce adequate security for most environments. At high security, they are tightened to severely restrict access.

Any modifications that this task makes to system files permissions or parameter settings are reported in the `tune.rpt` file.

System Files Checks

This task examines system files and compares each one with a description of that file listed in a master file. The master file is created the first time ASET runs this task. The master file contains the system file settings enforced by `checklist` for the specified security level.

A list of directories whose files are to be checked is defined for each security level. You can use the default list, or you can modify it, specifying different directories for each level.

For each file, the following criteria are checked:

- Owner and group
- Permission bits
- Size and checksum

- Number of links
- Last modification time

Any discrepancies found are reported in the `cklist.rpt` file. This file contains the results of comparing system file size, permission, and checksum values to the master file.

User/Group Checks

This task checks the consistency and integrity of user accounts and groups as defined in the `passwd` and `group` files. It checks the local, and NIS or NIS+ password files. NIS+ password file problems are reported but not corrected. This task checks for the following violations:

- Duplicate names or IDs
- Entries in incorrect format
- Accounts without a password
- Invalid login directories
- An account `nobody`
- Null group password
- A plus sign (+) in the `/etc/passwd` file on an NIS (or NIS+) server

Discrepancies are reported in the `usrgrp.rpt` file.

System Configuration Files Check

During this task, ASET checks various system tables, most of which are in the `/etc` directory. These files are:

- `/etc/default/login`
- `/etc/hosts.equiv`
- `/etc/inetd.conf`
- `/etc/aliases`
- `/var/adm/utmp`
- `/var/adm/utmpx`
- `/.rhosts`
- `/etc/vfstab`
- `/etc/dfs/dfstab`
- `/etc/ftpusers`

ASET performs various checks and modifications on these files, and reports all problems in the `sysconf.rpt` file.

Environment Check

This task checks how the `PATH` and `UMASK` environment variables are set for `root`, and other users, in the `/.profile`, `/.login`, and `/.cshrc` files.

The results of checking the environment for security are reported in the `env.rpt` file.

`eeprom` *Check*

This task checks the value of the `eeprom` security parameter to ensure that it is set to the appropriate security level. You can set the `eeprom` security parameter to `none`, `command`, or `full`.

ASET does not change this setting, but reports its recommendations in the `eeprom.rpt` file.

Firewall Setup

Chapter 4, “Securing the Network,” discusses how to use a firewall machine.

This task ensures that the system can be safely used as a network relay. It protects an internal network from external public networks by setting up a dedicated machine as a firewall. The firewall machine separates two networks, each of which approaches the other as untrusted. The firewall setup task disables the forwarding of Internet Protocol (IP) packets and hides routing information from the external network.

“Choose Which Tasks to Run: TASKS” on page 84 discusses how to edit the `asetenv` file.

The firewall task runs at all security levels, but takes action only at the highest level. If you want to run ASET at high security, but find that your system does not require firewall protection, you can eliminate the firewall task by editing the `asetenv` file.

Any changes made are reported in the `firewall.rpt` file.

ASET Reports

All report files generated from ASET tasks are found in subdirectories under the directory `/usr/aset/reports`.

ASET Execution Log

ASET generates an execution log whether it runs interactively or in the background. By default, ASET generates the log file on standard output. The execution log confirms that ASET ran at the designated time, and also contains any execution error messages. The `-n` option of the `aset` command directs the log to be delivered by electronic mail to a designated user. For a complete list of ASET options, refer to the `aset(1M)` reference page.

Example of an Execution Log File

```
ASET running at security level low

Machine=example; Current time = 0325_08:00


aset: Using /usr/aset as working directory

Executing task list...
        firewall
        env
        sysconfig
        usrgrp
        tune
        cklist
        eeprom
All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
     %/usr/aset/util/taskstat     aset_dir
Where aset_dir is ASET's operating directory, currently=/usr/aset

When the tasks complete, the reports can be found in:
     /usr/aset/reports/latest/*.rpt
You can view them by:
more /usr/aset/reports/latest/*.rpt
```

The log first shows the machine and time that ASET was run. Then it lists each task as it is started.

For a discussion of each of the tasks in the execution log file, turn to “ASET Tasks” on page 75.

ASET invokes a background process for each of these tasks. The task is listed in the execution log when it starts; this does not indicate that it has been completed. To check the status of the background tasks, use the `taskstat` utility.

ASET `reports` *Directory Structure*

This section describes the structure of the `reports` directory, and provides some guidelines on managing the report files.

For instructions on how to review the data gathered in the ASET reports, turn to “How to Manage the ASET Reports” on page 93.

ASET places the report files in subdirectories that are named to reflect the time and date when the reports are generated. This enables you to keep an orderly trail of records documenting the system status as it varies between ASET executions. You can monitor and compare these reports to determine the soundness of your system’s security. Figure 5-1 shows an example of the `reports` directory structure.

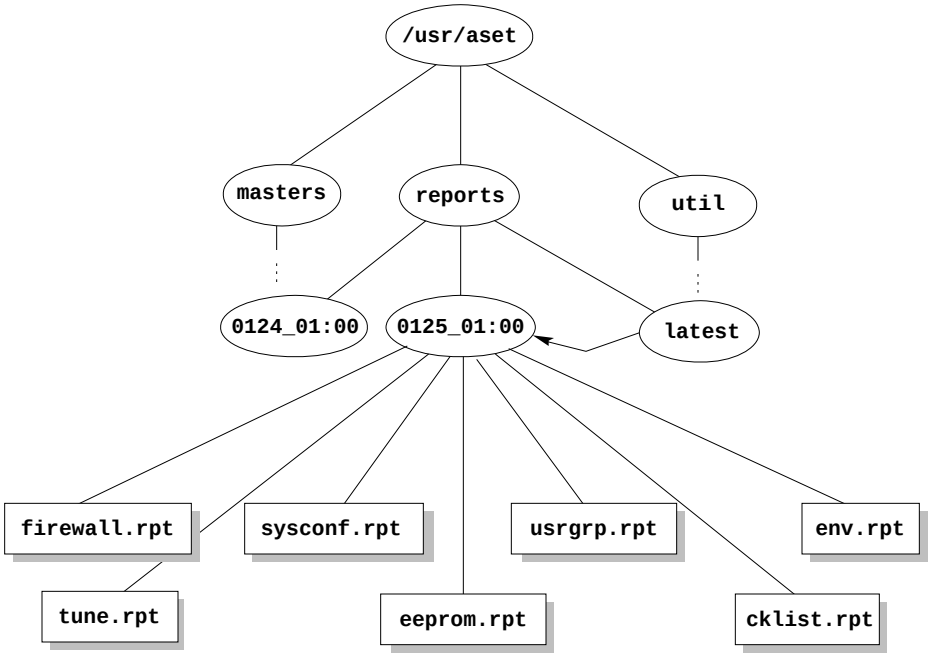


Figure 5-1 `reports` Directory Structure

Two report subdirectories are shown in this example:

- `0124_01:00`
- `0125_01:00`

The subdirectory names indicate the date and time the reports were generated. Each report subdirectory name has the following format:

monthdate_hour:minute

where *month*, *date*, *hour*, and *minute* are all two-digit numbers. For example, `0125_01:00` represents January 25, at 1 a.m.

Each of the two report subdirectories contains a collection of reports generated from one execution of ASET.

The directory `latest` is a symbolic link that always points to the subdirectory that contains the latest reports. Therefore, to look at the latest reports that ASET has generated, you can go to the `/usr/aset/reports/latest` directory. There is a report file in this directory for each task that ASET performed during its most recent execution.

Report Files Format

Each report file is named after the task that generates it. See Table 5-1 for a list of tasks and their reports.

Table 5-1 ASET Tasks and Resulting Reports

Tasks	**Report**
System files permissions Tuning (`tune`)	`tune.rpt`
System files checklist (`cklist`)	`cklist.rpt`
User / group checks (`usrgrp`)	`usrgrp.rpt`
System configuration files check (`sysconf`)	`sysconf.rpt`
Environment check (`env`)	`env.rpt`
`eeprom` check (`eeprom`)	`eeprom.rpt`
Firewall setup (`firewall`)	`firewall.rpt`

Within each report file, messages are bracketed by a beginning and an ending banner line. Sometimes a task terminates prematurely; for example, when a component of ASET is accidently removed or damaged. In most cases, the report file will contain a message near the end that indicates the reason for the premature exit.

The following is a sample report file, `usrgrp.rpt`.

```
*** Begin User and Group Checking ***

Checking /etc/passwd ...
Warning! Password file, line 10, no passwd
:sync::1:1::/:/bin/sync
..end user check; starting group check ...
Checking /etc/group...
*** End User And group Checking ***
```

Examining Report Files

After initially running or reconfiguring ASET, you should examine the report files closely. (Reconfiguration includes modifying the `asetenv` file or the master files in the `masters` subdirectory, or changing the security level at which ASET operates.) The reports record any errors introduced when you reconfigured. By watching the reports closely, you can react to, and solve, problems as they arise.

Comparing Reports

After you monitor the report files for a period during which there are no configuration changes or system updates, you may find that the content of the reports begins to stabilize and that it contains little, if any, unexpected information. You can use the `diff` utility to compare reports.

ASET Files

The files that ASET uses to define security levels and establish parameters under which it runs are the master files and the environment file.

Master Files

ASET's master files, `tune.high`, `tune.low`, `tune.med` and `uid_aliases`, are located in the `/usr/aset/masters` directory.

Tune Files

The `tune.low`, `tune.med`, and `tune.high` master files define the available ASET security levels. They specify the attributes of system files at each level and are used for comparison and reference purposes.

The `uid_aliases` *File*

The `uid_aliases` file contains a list of multiple user accounts sharing the same ID. Normally, ASET warns about such multiple user accounts because this practice lessens accountability. You can allow for exceptions to this rule by listing the exceptions in the `uid_aliases` file. ASET does not report entries in the `passwd` file with duplicate user IDs if these entries are specified in the `uid_aliases` file.

Avoid having multiple user accounts (password entries) share the same user ID. You should consider other methods of achieving your objective. For example, if you intend for several users to share a set of permissions, you could create a group account. Sharing user IDs should be your last resort, used only when absolutely necessary and when other methods will not accomplish your objectives.

You can use the `UID_ALIASES` environment variable to specify an alternate aliases file. The default is `/usr/aset/masters/uid_aliases`.

The Checklist Files

The master files used by the systems files checklist are generated when you first execute ASET, or when you run ASET after you change the security level.

The files checked by this task are defined by the environment variables: `CKLISTPATH_LOW`, `CKLISTPATH_MED`, and `CKLISTPATH_HIGH`.

Environment File, `asetenv`

The environment file, `asetenv`, contains a list of variables that affect ASET tasks. These variables can be changed to modify ASET operation.

5

Configuring ASET

This section discusses how ASET is configured and the environment under which it operates.

ASET requires minimum administration and configuration, and in most cases, you can run it with the default values. You can, however, fine-tune some of the parameters that affect the operation and behavior of ASET to maximize its benefit. Before changing the default values, you should understand how ASET works, and how it affects the components of your system.

ASET relies on four configuration files to control behavior of its tasks:

- `/usr/aset/asetenv`
- `/usr/aset/masters/tune.low`
- `/usr/aset/masters/tune.med`
- `/usr/aset/masters/tune.high`

Modifying the Environment File, `asetenv`

The `/usr/aset/asetenv` file has two main sections:

- A user-configurable parameters section
- An internal environment variables section

You can alter the user-configurable parameters section. However, the settings in the internal environment variables section are for internal use only and should not be modified.

You can edit the entries in the user-configurable parameters section to:

- Choose which tasks to run
- Specify directories for checklist task
- Schedule ASET execution
- Specify an aliases file
- Extend checks to NIS+ tables

Choose Which Tasks to Run: TASKS

Each of the tasks ASET performs monitors a particular area of system security. In most system environments, all the tasks are necessary to provide balanced security coverage. However, you may decide to eliminate one or more of the tasks.

For example, the firewall task runs at all security levels, but takes action only at the high security level. You may want to run ASET at the high-security level, but do not require firewall protection.

It's possible to set up ASET to run at the high level without the firewall feature by editing the TASKS list of environment variables in the asetenv file. By default, the TASKS list contains all of the ASET tasks. (An example is shown below). To delete a task, remove the task setting from the file. In this case, you would delete the firewall environment variable from the list. The next time ASET runs, the excluded task will not be performed.

```
TASKS="env sysconfig usrgrp tune cklist eeprom firewall"
```

Specify Directories for Checklist Task: CKLISTPATH

The system files check checks attributes of files in selected system directories. You define which directories to check by using these checklist path environment variables:

- CKLISTPATH_LOW
- CKLISTPATH_MED
- CKLISTPATH_HIGH

The CKLISTPATH_LOW variable defines the directories to be checked at the low security level.

CKLISTPATH_MED and CKLISTPATH_HIGH environment variables function similarly for the medium and high security levels.

For additional information, see "CKLISTPATH_level Variable" on page 98.

The directory list defined by a variable at a lower security level should be a subset of the directory list defined at the next higher level. For example, all directories specified for CKLISTPATH_LOW should be included in CKLISTPATH_MED, and all the directories specified for CKLISTPATH_MED should be included in CKLISTPATH_HIGH.

Checks performed on these directories are not recursive; ASET only checks those directories explicitly listed in the variable. It does not check their subdirectories.

You can edit these variable definitions to add or delete directories that you want ASET to check. Note that these checklists are useful only for system files that do not normally change from day to day. A user's home directory, for example, is generally too dynamic to be a candidate for a checklist.

Schedule ASET Execution: PERIODIC_SCHEDULE

When you start ASET, you can start it interactively, or use the `-p` option to request that the ASET tasks run at a scheduled time and period. You can run ASET periodically, at a time when system demand is light. For example, ASET consults `PERIODIC_SCHEDULE` to determine how frequently to execute the ASET tasks, and at what time to run them. The format of `PERIODIC_SCHEDULE` follows the format of `crontab` entries. See the `crontab(1)` reference page entry for complete information.

Specify an Aliases File: UID_ALIASES

The `UID_ALIASES` variable specifies an aliases file that lists shared user IDs. The default is `/usr/aset/masters/uid_aliases`.

For instructions on how to set up ASET to run periodically, turn to "How to Set Up ASET to Run Periodically" on page 91.

Extend Checks to NIS+ Tables: YPCHECK

The `YPCHECK` environment variable specifies whether ASET should also check system configuration file tables. `YPCHECK` is a Boolean variable; you can specify only true or false for it. The default value is false, disabling NIS+ table checking.

To understand how this variable works, consider its effect on the `passwd` file. When this variable is set to false, ASET checks the local `passwd` file. When it is set to true, the task also checks the NIS+ `passwd` file for the domain of the machine.

Note – Although ASET automatically repairs the local tables, it only reports potential problems in the NIS+ tables; it does not change them.

Modifying the Tune Files

If you need additional information, see “Tune Files” on page 98.

ASET uses the three master tune files, `tune.low`, `tune.med`, and `tune.high`, are used by ASET to ease or tighten access to critical system files. These master files are located in the `/usr/aset/masters` directory. They can be modified to suit your environment.

The `tune.low` file sets permissions to values appropriate for default system settings. The `tune.med` file further restricts these permissions and includes entries not present in `tune.low`. The `tune.high` file restricts permissions even further.

Note – Modify settings in the tune file modified by adding or deleting file entries. Setting a permission to a less restrictive value than the current setting has no effect; the ASET tasks do not relax permissions unless you downgrade your system security to a lower level.

Restoring System Files Modified by ASET

When ASET is executed for the first time, it saves and archives the original system files. The `aset.restore` utility reinstates these files. It also deschedules ASET, if it is currently scheduled for periodic execution. The `aset.restore` utility is located in the ASET operating directory, `/usr/aset`.

Changes made to system files are lost when you run `aset.restore`.

You should use `aset.restore`:

- When you want to remove ASET changes and restore the original system. If you want to deactivate ASET permanently, you can remove it from `cron` scheduling if the `aset` command had been added to root’s `crontab` previously. For directions on how to use cron to remove automatic execution, see “To remove the crontab entry:” on page 92.
- After a brief period of experimenting with ASET, to restore the original system state.
- When some major system functionality is not working properly and you suspect that ASET is causing the problem.

5

Network Operation Using the NFS System

Generally, ASET is used in standalone mode, even on a machine that is part of a network. As system administrator for your standalone system, you are responsible for the security of your system and for running and managing ASET to protect your system.

You can also use ASET in the NFS distributed environment. As a network administrator, you are responsible for installing, running, and managing various administrative tasks for all of your clients. To facilitate ASET management across several client systems, you can make configuration changes that are applied globally to all clients, eliminating the need for you to log in to each system to repeat the process.

When deciding how to set up ASET on your networked systems, you should consider how much you want users to control security on their own systems, and how much you want to centralize responsibility for security control.

Providing a Global Configuration for Each Security Level

A case might arise where you want to set up more than one network configuration. For example, you may want to set up one configuration for clients that are designated with low security level, another configuration for those with medium level, and yet another one with high level.

If you need to create a separate ASET network configuration for each security level, you can create three ASET configurations on the server—one for each level. You would export each configuration to the clients with the appropriate security level. Some ASET components that are common to all three configurations could be shared using links.

Collecting ASET Reports

For instructions on how to set up a collection mechanism, see "How to Collect Reports on a Server" on page 93.

Just as you can centralize the ASET components on a server to be accessed by clients with or without root privilege, so you can collect all reports produced by tasks running on various clients in a central directory on the server.

Instructions for Using ASET

This section describes the steps to run ASET. It describes how to start ASET interactively using the default settings, and how to change the ASET default configuration to suit the requirements of your environment. It also tells you how to manage the ASET report files, and how to use ASET on a network.

▼ How to Run ASET Interactively

You must be root to run these commands.

To set the ASET security level:

♦ **Type** `/usr/aset/aset -l` *level* **and press Return.**
Use the `-l` option to specify a security level of low (`low`), medium (`med`), or high (`high`). The default setting is `low`.

The ASET tasks start running. The execution log message is displayed on the screen, telling you which tasks have started.

To name an ASET working directory:

♦ **Type** `/usr/aset/aset -d` *pathname* **and press Return.**
Use the `-d` option to specify a working directory for ASET. The default is `/usr/aset`.

Example of Initiating an ASET Session

The following example initiates an ASET session at medium-security level, specifying a working directory of `/usr/etc/asetdir`. The execution log file is displayed on the screen, listing the tasks that have been started.

```
example# /usr/aset/aset -l med -d /usr/etc/aset
ASET running at security level med
machine =example; Current time = 1007_14:13

aset: Using /usr/etc/aset as working directory
Executing task list ...
    env
    sysconf
    usrgrp
    tune
    cklist
    eeprom
    firewall

All tasks executed. Some background tasks may still be running.

Run /usr/etc/aset/util/taskstat to check their status:
    /usr/etc/aset/util/taskstat [aset_dir]

where aset_dir is ASET's operating directory, currently=/usr/etc/aset.

When the tasks complete, the reports can be found in:
    usr/etc/aset/reports/latest/*.rpt
You can view them by:
     more /usr/etc/aset/reports/latest/*.rpt
```

▼ How to Use Environment Variables to Set Options

When you run ASET interactively, you can use the environment variables, `ASETDIR` and `ASETSECLEVEL`, to specify the ASET working directory and the security level at which to run the tasks.

To set variables from a C shell:

♦ **Type** `setenv` *VARIABLE value* **and press Return.**

To set variables from a Bourne shell or a Korn shell:

♦ **Type the following commands**:

VARIABLE=*value*
`export` *VARIABLE*

Example of Setting Environment Variables

The following example specifies a working directory for ASET, and starts it running at a security level of medium.

```
example# setenv ASETDIR /usr/etc/asetdir
example# setenv ASETSECLEVEL med
example# aset
```

▼ How to Set Up ASET to Run Periodically

You can schedule ASET to run periodically when system demand is light. The ASET schedule is determined by an entry in the `crontab` file.

To set up ASET for running periodically:

♦ **Type** `aset -p` **and press Return.**
The `-p` option inserts a line in the `crontab` file that starts the ASET tasks running at the default time of 12:00 midnight every 24 hours.

If you need background information about running ASET automatically, turn to“Schedule ASET Execution: PERIODIC_SCHEDULE” on page 86.

To confirm the schedule:

- **Type** `crontab -l root` **and press Return.**
  This command displays the `crontab` entry, allowing you to confirm the schedule.

To remove the `crontab` ***entry:***

1. **Type** `crontab -e root` **and press Return.**
   This command invokes the editor on the `crontab` file.
2. **Delete the ASET entry.**

To change the `PERIODIC_SCHEDULE` ***variable default setting:***

1. **Open the file** `/usr/aset/asetenv` **for editing.**
2. **Edit the line with the** `PERIOD_SCHEDULE` **environment variable, inserting a new schedule.**

See “PERIODIC_SCHEDULE Variable” on page 96 of the Reference section for the format of the `asetenv` entry.

3. **Type** `/usr/aset/aset -p` **and press Return.**
   The `-p` option puts an entry in the `crontab` file. If you have previously scheduled ASET to run periodically at a different time, both entries are in the file and you must remove the unwanted entry.

To remove the unwanted entry:

1. **Type** `crontab -e root` **and press Return.**
2. **Delete the unwanted entry.**

You can also change the automatic execution entry by altering the `crontab` file directly.

The following entry in `asetenv` sets a schedule for ASET to be run at 1:00 a.m. every Monday, Wednesday, and Friday.

```
PERIOD_SCHEDULE=0 1 * * 1,3,5
```

▼ How to Manage the ASET Reports

If you need information about the reports, turn to "ASET reports Directory Structure" on page 80.

To view the ASET reports:

1. **Type** `cd /usr/aset/reports/latest` **and press Return.**
   This changes to the directory where the latest reports files reside.
2. **Type** `more *.rpt` **and press Return.**
   The reports most recently generated are displayed on the screen.

▼ How to Collect Reports on a Server

This section describes how to create a directory on the server to store all ASET reports.

1. **Set up a directory on the server:**
   a. **Type** `cd /usr/aset` **and press Return.**
   b. **Type** `mkdir` *rptdir* **and press Return.**
   These two commands create a directory (*rptdir*) on the server for report collection.
   c. **Type** `cd` *rptdir* **and press Return.**
   d. **Type** `mkdir` *client_rpt* **and press Return.**
   This creates a subdirectory (*client_rpt*) for a client. Repeat this step for each client whose reports you need to collect.

   The following example creates the directory `all_reports`, and the subdirectories `pluto_rpt` and `neptune_rpt`.

```
mars# cd /usr/aset
mars# mkdir all_reports
mars# cd all_reports
mars# mkdir pluto_rpt
mars# mkdir neptune_rpt
```

2. **Share the client subdirectories.**
   Add the *client_rpt* directories to the `/etc/dfs/dfstab` file. The directories should have read/write options.

For example, the following entries in `dfstab` are shared with read/write permissions.

```
share -F nfs -o rw=pluto /usr/aset/all_reports/pluto_rpt
share -F nfs -o rw=neptune /usr/aset/all_reports/neptune_rpt
```

3. **Type `shareall` and press Return.**
   This makes the resources in the `dfstab` file available to the clients.

4. **Type the following command on each client**:
   `mount server:/usr/aset/`*client_rpt* `/usr/aset/masters/reports`

   This mounts the client subdirectory (`/usr/aset/`*client_rpt*) from the server to the client, at the mount point, `/usr/aset/masters/reports`.

5. **Edit the `/etc/vfstab` file to mount the directory automatically at boot time.**
   The following sample entry in `/etc/vfstab` on `neptune` lists the directory to be mounted from `mars`, `/usr/aset/all_reports/neptune_rpt`, and the mount point on `neptune`, `/usr/aset/reports`. At boot time, the directories listed in `vfstab` are automatically mounted.

```
mars:/usr/aset/all_reports/neptune.rpt /usr/aset/reports nfs - yes hard
```

Setting up the collection of reports in this manner allows you to review reports for all clients on the server. You can use this method whether a client has `root` privilege or not. Alternatively, you can leave the reports directory on the local system when you want users to monitor their own ASET reports.

Reference Material for Using ASET

This section shows examples of the formats of environment variables and report files. See Appendix C, "Error Messages," for an explanation of the ASET error messages.

Environment Variables

Table 5-2 lists the ASET environment variables and the values that they specify.

Table 5-2 Environment Variables and Their Meanings

Environment Variable	Specifies
ASETDIR (See below)	ASET working directory
ASETSECLEVEL (See below)	Security level
PERIOD_SCHEDULE	Periodic schedule
TASKS	Tasks to run
UID_ALIAS	Aliases file
YPCHECK	Extends check to NIS and NIS+
CKLISTPATH_LOW	Directory lists for low security
CKLISTPATH_MED	Directory list for medium security
CKLISTPATH_HIGH	Directory list for high security

The environment variables listed below are found in the file `/usr/aset/asetenv`. The `ASETDIR` and `ASETSECLEVEL` variables are optional and can be set only through the shell using the `aset` command. The other environment variables can be set by editing the file. The variables are described below.

ASETDIR *Variable*

`ASETDIR` specifies an ASET working directory.

From the C shell, type:
`setenv ASETDIR` *pathname*

From the Bourne shell or the Korn shell, type:
`ASETDIR=`*pathname*
`export ASETDIR`

Set *pathname* to the full path name of the ASET working directory.

ASETSECLEVEL *Variable*

`ASETSECLEVEL` specifies a security level at which ASET tasks are executed.

From the C shell, type:
`setenv ASETSECLEVEL` *level*

From the Bourne shell or the Korn shell, type:
`ASETSECLEVEL=`*level*
`export ASETSECLEVEL`

In the above commands, *level* can be set to one of the following:

- `low` low security level
- `med` medium security level
- `high` high security level

PERIODIC_SCHEDULE *Variable*

The value of `PERIODIC_SCHEDULE` follows the same format as the `crontab` file. Specify the variable value as a string of five fields enclosed in double quotation marks, each field separated by a space:

"*minutes hours day-of-month month day-of-week*"

- *minutes hours* Specifies start time in number of minutes after the hour (0-59) and the hour (0-23)
- *day-of-month* Specifies the day of the month when ASET should be run, using values from 1 through 31
- *month* Specifies the month of the year when ASET should be run, using values from 1 through 12
- *day-of-week* Specifies the day of the week when ASET should be run, using values from 0 through 6; Sunday is day 0 in this scheme

The following rules apply:

- You can specify a list of values, each delimited by a comma, for any field.
- You can specify a value as a number, or you can specify it as a range; that is, a pair of numbers joined by a hyphen. A range states that the ASET tasks should be executed for every time included in the range.
- You can specify an asterisk (*) as the value of any field. An asterisk specifies all possible values of the field, inclusive.

The default entry for `PERIODIC_SCHEDULE` variable causes ASET to execute at 12:00 midnight every day:

```
PERIODIC_SCHEDULE="0 0 * * *"
```

`TASKS` *Variable*

The `TASKS` variable lists the tasks that ASET performs. The default is to list all seven tasks:

```
TASKS="env sysconfig usrgrp tune cklist eeprom firewall"
```

`UID_ALIASES` *Variable*

The `UID_ALIASES` variable specifies an aliases file. If present, ASET consults this file for a list of permitted multiple aliases. The format is `UID_ALIASES=`*pathname*. *pathname* is the full path name of the aliases file.

The default is:

```
UID_ALIASES=${ASETDIR}/masters/uid_aliases
```

YPCHECK Variable

The `YPCHECK` variable extends the task of checking system tables to include NIS or NIS+ tables. It is a Boolean variable, which can be set to either true or false.

The default is false, confining checking to local system tables:

```
YPCHECK=false
```

CKLISTPATH_*level Variable*

The three checklist path variables list the directories to be checked by the checklist task. The following definitions of the variables are set by default; they illustrate the relationship between the variables at different levels:

```
CKLISTPATH_LOW=${ASETDIR}/tasks:${ASETDIR}/util:${ASETDIR}/masters:/etc
CKLISTPATH_MED=${CKLISTPATH_LOW}:/usr/bin:/usr/ucb
CKLISTPATH_HIGH=${CKLISTPATH_MED}:/usr/lib:/sbin:/usr/sbin:/usr/ucblib
```

The values for the checklist path environment variables are similar to those of the shell path variables, in that they are lists of directory names separated by colons (:). You use an equal sign (=) to connect the variable name to its value.

ASET File Examples

This section has examples of some of the ASET files, including the tune files and the aliases file.

Tune Files

ASET maintains three tune files. Entries in all three tune files have the following format:

pathname	The full path name of the file
mode	A five-digit number that represents the permission setting
owner	The owner of the file
group	The group of the file
type	The type of the file

The following rules apply:

- You can use regular shell wildcard characters, such as an asterisk (*) and a question mark (?), in the path name for multiple references. See the reference page for the shell command `sh(1)`.

- *mode* represents the least restrictive value. If the current setting is already more restrictive than the specified value, ASET does not loosen the permission settings. For example, if the specified value is 00777, the permission will remain unchanged, because 00777 is always less restrictive than whatever the current setting is.

  This is how ASET handles mode setting, unless the security level is being downgraded or you are removing ASET. When you decrease the security level from what it was for the previous execution, or when you want to restore the system files to the state they were in before ASET was first executed, ASET recognizes what you are doing and decreases the protection level.

- You must use names for *owner* and *group* instead of numeric IDs.
- You can use a question mark (?) in place of *owner*, *group*, and *type* to prevent ASET from changing the existing values of these parameters.
- *type* can be `symlink` (symbolic link), `directory`, or `file` (everything else).
- Higher security level tune files reset file permissions to be at least as restrictive as they are at lower levels. Also, at higher levels, additional files are added to the list.
- A file can match more than one tune file entry. For example, `etc/passwd` matches `etc/pass*` and `/etc/*` entries.
- Where two entries have different permissions, the file permission is set to the most restrictive value. In the following example, the permission of `/etc/passwd` will be set to 00755, which is the more restrictive of 00755 and 00770.

```
/etc/pass*    00755    ?    ?  file
   /etc/*     00770    ?    ?  file
```

- If two entries have different *owner* or *group* designations, the last entry takes precedence.

  The following example shows the first few lines of the `tune.low` file.

```
/           02755  root   staff  directory
/bin        00777  root   staff  symlink
/etc        02755  root   staff  directory
/etc/chroot 00777  root   staff  symlink
/etc/clri   00777  root   staff  symlink
```

Aliases File

An aliases file contains a list of aliases that share the same user ID.

Each entry is in this form:

uid=alias1=alias2=alias3=`...`

Where:

uid is the shared user ID.

aliasn is the user account sharing the user ID.

For example, the following entry lists the user ID `0` being shared by `sysadm` and `root`:

```
0=root=sysadm
```

Part 2— Performance and Accounting

Part 2 contains the following chapters:

- “Introduction to Performance”
- “Managing Processes”
- “Monitoring Performance”
- “A Guide to Network Performance”
- “Setting Up and Maintaining Accounting”

Introduction to Performance 6

Getting good performance from a computer or network is an important part of system administration. This chapter is an overview of some of the factors that contribute to maintaining and managing the performance of the computer systems in your care.

Note – The chapters on system performance in this book contain information about SunOS 5.3 system software with specific references to SunOS 5.4 system software noted.

About Performance

The performance of a computer system depends on how the system uses and allocates its resources. It is important to monitor your system's performance on a regular basis so that you know how it behaves under normal conditions. You should have a good idea of what to expect, and be able to recognize a problem when it occurs.

Several tools are available that gather and display important data about your system. The `sadc` and `sar` utilities sample and report on various system-wide activities. Chapter 8, "Monitoring Performance," describes how to use these commands and explains the information they provide.

The performance of a network depends not only on the individual systems that make up the network, and on the interrelations among them. Any member of the network, or the physical components of the network itself, including the

cables and connectors, can have an impact on the performance of the network as a whole. Discovering and isolating problems can sometimes involve a great deal of detective work. Chapter 9, "A Guide to Network Performance," describes some of the tools that you can use to track down and investigate the cause of a network slowdown.

Managing System Resources

The performance of a computer system depends upon how the system uses and allocates its resources. System resources include:

- *Central processing unit (CPU)* – The CPU processes instructions, fetching instructions from memory and executing them.
- *Input/output (I/O) devices* – I/O devices transfer information into and out of the computer. Such a device could be a terminal and keyboard, a disk drive, or a printer.
- *Memory* – Physical (or main) memory is the amount of memory (RAM) on the system.

Chapter 8, "Monitoring Performance," describes the tools that display statistics about the activity and the performance of the computer system.

Managing Processes

You can use the `timex` command to report the amount of time it takes for a command to be executed, and which resources the command uses during its execution. For example, using `timex` on the `sleep` command produces:

```
% timex sleep 20
real      20.60
user       0.04
sys        0.37
```

The `sleep` command suspends execution for a specified number of seconds. The example shows that it took a total of 20.60 seconds to execute the command. However, the actual CPU time was a total of 0.41 seconds—0.04 seconds devoted to executing the code, and 0.37 seconds used by the operating system on behalf of the program. The rest of the time was spent idle (sleeping).

The `timex` command can also report the operating system activities that occurred while the command was being executed. If no other processes are running, `timex` can give you a good idea of which resources a specific command uses during its execution. You can collect a record of CPU consumption for each application program and then use this information to tune the heavily loaded resources.

The kernel is responsible for allocating the CPU time, which it does according to a set of policies. The policies determine the order in which processes are executed and the amount of CPU time they are allotted. A large number of processes contending for the CPU can slow down the system to unacceptable performance.

One way to help an overloaded CPU is to spread out the workload. You can, for example, run a CPU-intensive program at night, or at an hour when the system is not busy. Use the `at` command to submit a command or application for execution at a later time. See the reference page for `at(1)` for a description of this command.

Using `at`, you can specify an hour, such as 0100 (which means 1 a.m.), when you want the commands to be executed.

Use the `-f` option to specify a script that contains a list of commands to be executed. Or, you can list the commands, ending with Control-d.

Use the `-r` option to cancel jobs that you previously scheduled with `at`. The `-r` option takes the job number as an argument. Type `at -l` to find the job number of a previously scheduled job.

The following example submits three commands to be executed at 5:05 p.m.

```
mysys% at 1705
at> command1
at> command2
at> command3
at> Control-D
warning: commands will be executed using /usr/bin/sh
job 763689900.a at Mon Mar 14 17:05:00 1994
mysys%
```

6

Managing Disk I/O

The disk is used to store data and instructions used by your computer system. You can examine how efficiently the system is accessing data on the disk by looking at the disk access activity and terminal activity. See Chapter 8, "Monitoring Performance," for a discussion of the `iostat` and `sar` commands, which report statistics on disk activity. Managing and allocating disk space and partitioning your disk, are discussed in *File System Administration*.

Improving and Controlling File-System Use

One of the biggest problems you can encounter with your system is running out of disk space. To avoid file-system space problems, you should try to monitor the amount free disk space at the start of each week to determine whether the system has enough disk space for what your users and applications normally generate in a week.

See *File System Administration* for a discussion of planning `ufs` file systems.

Checking for Disk Slowdowns

If the CPU spends much of its time waiting for I/O completions, there is a problem with disk slowdown. Some ways to prevent disk slowdowns are:

- Keep disk space with 10% free so file systems are not full. If a disk becomes full, back up and restore the file systems to prevent disk fragmentation. Consider purchasing products that resolve disk fragmentation.
- Organize the file system to minimize disk activity. If you have two disks, distribute the file system for a more balanced load. Using Sun's Online: DiskSuite™ product provides more efficient disk usage.
- Add more memory. Additional memory reduces swapping and paging traffic, and allows an expanded buffer pool (reducing the number of user-level reads and writes that need to go out to disk).
- Add a disk and balance the most active file systems across the disks.

Managing Memory

Performance suffers when the programs running on the system require more physical memory than is available. When this happens, the operating system begins paging and swapping.

Paging involves moving pages that have not been recently referenced to a free list of available memory pages. Most of the kernel resides in main memory and is not pageable.

Swapping occurs if the page daemon cannot keep up with the demand for memory. The swapper will attempt to swap out sleeping or stopped lightweight processes (LWPs). If there are no sleeping or stopped LWPs, the swapper will swap out a runnable process. The swapper will swap LWPs back in based on their priority. It will attempt to swap in processes that are runnable.

Paging and swapping of pages is costly in both disk and CPU overhead.

See Chapter 7, “Managing Processes,” for a discussion of process terminology.

Adding Swap Space

Swap areas are really file systems used for swapping. Swap areas should be sized based on the requirements of your applications. Check with your vendor to identify application requirements.

Table 6-1 describes the formula used to size default swap areas by the Solaris 2.x installation program. These default swap sizes are a good place to start if you are not sure how to size your swap areas.

Table 6-1 Default Swap Sizes

If Your Physical Memory Size Is...	Your Default Swap Size Is...
16–64 Mbytes	32 Mbytes
64-128 Mbytes	64 Mbytes
128-512 Mbytes	128 Mbytes
greater than 512 Mbytes	256 Mbytes

See *File System Administration* for information about managing swap space.

Buffer Resources

The buffer cache for `read(2)` and `write(2)` system calls uses a range of virtual addresses in the kernel address space. A page of data is mapped into the kernel address space and the amount of data requested by the process is then physically copied to the process' address space. The page is then unmapped in the kernel. The physical page will remain in memory until the page is freed up by the page daemon.

This means a few I/O-intensive processes can monopolize or force other processes out of main memory. To prevent monolopization of main memory, balance the running of I/O-intensive processes serially in a script or with the `at(1)` command. Programmers can use `mmap(2)` and `madvise(3)` to ensure that their programs free memory when they are not using it.

Monitoring Tools

The Solaris 2.x system software provides several tools to help you keep track of how your system is performing. These include:

- The `sar` and `sadc` utilities, which collect and report on many aspects of system activity. Chapter 8, "Monitoring Performance," describes these utilities and the information that they provide.
- The `ps` command, which provides information about the active processes; Chapter 7, "Managing Processes," describes the `ps` command.
- The performance meter, which provides a graphical representation of the status of your system and other hosts on the network; Chapter 8, "Monitoring Performance," describes the performance meter.
- The `vmstat` and `iostat` commands, which summarize system activity, providing information about virtual memory activity, disk usage, and CPU activity; Chapter 8, "Monitoring Performance," describes these tools.
- The `swap` command, which can be used to display information about available swap space on your system. See *File System Administration* for information on using the `swap` command.
- The `netstat` and `nfsstat` commands, which display information about network performance. Chapter 9, "A Guide to Network Performance," describes these commands.

Kernel Parameters

Many basic parameters (or tables) within the kernel are calculated from the value of the `maxusers` parameter. Tables are allocated space dynamically. However, you can set maximums for these tables to ensure that applications won't take up large amounts of memory.

By default, `maxusers` is approximately set to the number of Mbytes of physical memory on the system. However, the system will never set `maxusers` higher than 1048.

See Appendix A, "Tuning Kernel Parameters," and the reference page for `system(4)` for details on kernel parameters.

In addition to `maxusers`, a number of kernel parameters are allocated dynamically based on the amount of physical memory on the system, as shown in Table 6-2 below.

Table 6-2 Kernel Parameters

Kernel Parameter	Description
`ufs_ninode`	The maximum size of the inode table
`ncsize`	The size of the directory name lookup cache
`max_nprocs`	The maximum size of the process
`ndquot`	The number of disk quota structures
`maxuprc`	The maximum number of user processes per user-id

Table 6-3 lists the default settings for kernel parameters affected by the value assigned to `maxusers`.

Table 6-3 Default Settings for Kernel Parameters

Kernel Table	Variable	Default Setting
Inode	`ufs_ninode`	*max_nprocs* + 16 + *maxusers* + 64
Name cache	`ncsize`	*max_nprocs* + 16 + *maxusers* + 64
Process	`max_nprocs`	10 + 16 * *maxusers*
Quota table	`ndquot`	(*maxusers* * *NMOUNT*) / 4 + *max_nprocs*
User process	`maxuprc`	*max_nprocs* – 5

See Appendix A, "Tuning Kernel Parameters," for a description of the kernel parameters and how to change the default values.

Sources of Information

Performance is a broad subject that can't be adequately covered in these chapters. There are several books available that cover various aspects of improving performance and tuning your system or network. Three useful books are:

- *Sun Performance and Tuning: SPARC and Solaris*, by Adrian Cockcroft, SunSoft Press / PRT Prentice Hall, ISBN 0-13-149642-3
- *System Performance Tuning*, by Mike Loukides, O'Reilly & Associates, Inc.
- *Managing NFS and NIS*, by Hal Stern, O'Reilly & Associates, Inc.

Managing Processes 7

The processes running on a system consume the system's resources. You may be able to influence the performance of your system by controlling and monitoring the processes that the CPU is executing.

This chapter describes how to monitor and manage processes running on your system. If you are familiar with the concepts of process control, use the following table to go directly to the step-by-step instructions.

7

Process Terminology

The new structures and terms related to processes are described in Table 7-1.

Table 7-1 Process Terminology

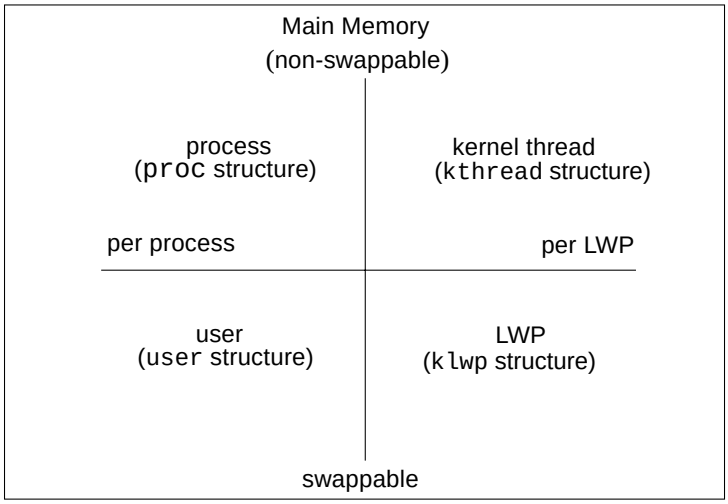
Term	Description
process	An instance of program in execution.
lightweight process	Is a virtual CPU or execution resource. LWPs are scheduled by the kernel to use available CPU resources based on their scheduling class and priority. LWPs include a kernel thread, which contains information that has to be in memory all the time and an LWP, which contains information that is swappable.
application thread	A series of instructions with a separate stack that can execute independently in a user's address space. They can be multiplexed on top of LWPs.

A process can consist of multiple lightweight processes (LWPs), and multiple application threads. The kernel schedules a kernel-thread structure, which is the scheduling entity in the SunOS 5.x environment. The different process structures are described below.

Structure	Description
`proc`	Contains information that pertains to the whole process and has to be in main memory all the time.
`kthread`	Contains information that pertains to one LWP and has to be in main memory all the time.
`user`	Contains the per process information that is swappable.
`klwp`	Contains the per LWP process information that is swappable.

Figure 7-1 illustrates the relationship of these structures.

Figure 7-1 Process Structures



Most process resources are accessible to all the threads in the process. Almost all process virtual memory is shared. A change in shared data by one thread is available to the other threads in the process.

About Monitoring Processes

The ps *Command*

The ps command enables you to monitor the execution status (or process status) of the active processes on a system. Depending on which options you use, the command reports the following information:

- Current status of the process
- Process ID
- Parent process ID
- User ID
- Scheduling class
- Priority
- Address of the process
- Memory used
- CPU time used

Table 7-2 lists and describes the fields reported by `ps`. The fields displayed depend on which option you choose. See the reference page for `ps(1)` for a description of all the available options.

Table 7-2 Summary of Fields in `ps` Reports

Field	Description
`F`	Hexadecimal flags, which, added together, indicate the process's current state as follows: `00` The process has terminated. `01` The process is a system process and is always in memory. `02` The process is being traced by its parent. `04` The process is stopped. `08` The process cannot be awakened by a signal. `10` The process is currently in memory and is locked. `20` The process cannot be swapped.
`S`	The current state of the process, as indicated by one of the following letters: `O` The process is currently running. `S` The process is sleeping; waiting for an I/O event to be completed. `R` The process is ready to run. `I` The process is idle; it is being created. `Z` This is a zombie process; the process has terminated and the parent has not reaped its status. `T` The process is stopped. `X` The process is waiting for more memory.
`UID`	The user ID of the process's owner.
`PID`	The process identification number.
`PPID`	The parent process's identification number.
`C`	The processor utilization for scheduling; this field is not displayed when the `-c` option is used.
`CLS`	The scheduling class to which the process belongs: real-time, system, or timesharing. This field is included only with the `-c` option.
`PRI`	The kernel thread's scheduling priority; higher numbers mean higher priority.
`NI`	The process's `nice` number, which contributes to its scheduling priority. Making a process nicer means lowering its priority.

Table 7-2 Summary of Fields in `ps` Reports (Continued)

Field	Description
ADDR	The address of the `proc` structure.
SZ	The virtual address size of process.
WCHAN	The address of an event or lock for which the process is sleeping.
STIME	The starting time of the process (in hours, minutes, and seconds).
TTY	The terminal from which the process (or its parent) was started. A question mark indicates there is no controlling terminal.
TIME	The total amount of CPU time used by the process since it began.
COMD	The command that generated the process.

Typing `ps` with no arguments shows you the processes associated with your login session:

```
mysys% /usr/bin/ps
  PID   TTY    TIME COMD
 1664  pts/4  0:06 csh
 2081  pts/4  0:00 ps
```

Typing `ps -ef` displays a full (`-f`) listing of all the processes being executed on the system (`-e`). The UID displays the login name of the user.

```
mysys% ps   -ef
UID        PID     PPID     C        STIME    TTY      TIME     CMD
root       0       0         80      Jan 30   ?         0:02    sched
root       1       0         80      Jan 30   ?         7:17    /etc/init
root       2       0         67      Jan 30   ?         0:01    pageout
root       3       0         80      Jan 30   ?        69.46    fsflush
root       123     1         80      Jan 30   ?         0:12    /usr/lib/saf/sac -t 300
root       122     1         80      Jan 30   ?         1:31    /usr/sbin/rpcbind
root       124     1         80      Feb 04   ?         0.37    /usr/sbin/keyserv
gsmith     1117    1115      80      Feb 04   pts/0     0:04    -csh
ebrown     1664    1662      80      Feb 04   pts/4     0:06    -csh
ebrown     2128    1664      80      Feb 04   pts/4     0:00    /usr/bin/ps -ef
```

This `ps` output shows that the first process executed when the system boots is the swapper (`sched`) followed by the `init` process, `pageout`, and so on.

Looking for Problems

Here are some tips on obvious problems you may find:

- Look for several identical jobs owned by the same user. This may come as a result of running a script that starts a lot of background jobs without waiting for any of the jobs to finish.
- Look for a process that has accumulated a large amount of CPU time. You'll see this by looking at the `TIME` field. Possibly, the process is in an endless loop.
- Look for a process running with a priority that is too high. Type `ps -c` to see the `CLS` field, which displays the scheduler class of each process. A process executing as a real-time (`RT`) process can monopolize the CPU. Or look for a timeshare (`TS`) process with a high nice value. A user with superuser privileges may have bumped up the priorities of this process. The system administrator can lower the priority by using the `nice` command.

For a description of the `nice` command, turn to "Changing the Priority of a Timesharing Process With nice" on page 118.

- Look for a runaway process—one that progressively uses more and more CPU time. You can see it happening by looking at the time when the process started (`STIME`) and by watching the cumulation of CPU time (`TIME`) for awhile.

Process Priority Levels

A process is allocated CPU time according to its scheduling class and its priority level. By default, the SunOS 5.x operating system has four process scheduling classes: *real-time*, *system*, *timesharing* and *interactive*.

- Real-time processes have the highest priority. This class includes processes that must respond to external events as they happen. For example, a process that collects data from a sensing device may need to process the data and respond immediately. In most cases, a real-time process requires a dedicated system. No other processes can be serviced while a real-time process has control of the CPU. By default, the range of priorities is 100-159.

For instructions on how to determine the scheduling class of a process, turn to "How to Get Basic Information About Process Classes" on page 120.

- System processes have the middle priorities. This class is made up of those processes that are automatically run by the kernel, such as the swapper and the paging daemon. By default, the range of priorities is 60-99.

- Timesharing processes have the lowest priority. This class includes the standard UNIX processes. Normally, all user processes are timesharing processes. They are subject to a scheduling policy that attempts to distribute processing time fairly, giving interactive applications quick response time and maintaining good throughput for computations. By default, the range of priorities is 0-59.
- Interactive processes are introduced in the SunOS 5.4 environment. The priorities range from 0-59. All processes started under OpenWindows are placed in the interactive class and those processes with keyboard focus get higher priorities.

The scheduling priority determines the order in which processes will be run.

Real-time processes have fixed priorities. If a real-time process is ready to run, no system process or timesharing process can run.

System processes have fixed priorities that are established by the kernel when they are started. The processes in the system class are controlled by the kernel, and cannot be changed.

Timesharing and interactive processes are controlled by the scheduler, which dynamically assigns their priorities. You can manipulate the priorities of the processes within this class.

Changing the Scheduling Priority of Processes With `priocntl`

For instructions on how to assign the priority of a process, turn to "How to Designate Priority With priocntl" on page 121.

The scheduling priority of a process is the priority it is assigned by the process scheduler. These priorities are assigned according to the scheduling policies of the scheduler. The `dispadmin` command lists the default scheduling policies. See Appendix B, "The Scheduler," for information on the `dispadmin` command.

The `priocntl` command can be used to assign processes to a priority class and to manage process priorities. See the section called "Instructions for Managing Processes" on page 120 for instructions on using the `priocntl` command to manage processes.

Changing the Priority of a Timesharing Process With `nice`

The `nice` command is only supported for backward compatibility to previous SunOS releases. The `priocntl` command provides more flexibility in managing processes.

For an example of using the `nice` command, turn to "How to Change the Priority of a Process with the nice Command" on page 123.

The priority of a process is determined by the policies of its scheduling class, and by its *nice number*. Each timesharing process has a global priority which is calculated by adding the user-supplied priority, which can be influenced by the nice or priocntl command, and the system-calculated priority.

The execution priority number of a process is assigned by the operating system, and is determined by several factors, including its schedule class, how much CPU time it has used, and (in the case of a timesharing process) its nice number.

Each timesharing process starts with a default nice number, which it inherits from its parent process. The nice number is shown in the `NI` column of the `ps` report.

A user can lower the priority of a process by increasing its user-supplied priority. But only the system administrator (or root) can lower a nice number to increase the priority of a process. This is to prevent users from increasing the priorities of their own processes, thereby monopolizing a greater share of the CPU.

Nice numbers range between 0 and +40, with 0 conferring the highest priority. The default value is 20. Two versions of the command are available, the standard version, `/usr/bin/nice`, and a version that is part of the C shell.

Killing a Process

Sometimes it is necessary to stop (kill) a process. The process may be in an endless loop, or you may have started a large job that you want to stop before it is completed. You can kill any process that you own, and the superuser can kill any processes in the system except for those with `PID`s `0`, `1`, `2`, `3`, and `4`.

You can destroy a process using the `kill` command:

```
kill PID
```

The default action of this command is to send signal 15 to the process identified by the PID. You should use the `ps` command to be sure that the process is truly gone. If the process is still alive after the `kill` command, you can ensure its termination by sending the `-9` signal:

```
kill –9 PID
```

Note – Some sleeping processes may be not wakeable by signals.

You may find that although you have killed a process, the children have been inherited by `init`, and are still active. You can give multiple PIDs as arguments to the `kill` command.

7

Instructions for Managing Processes

This section provides step-by-step instructions on how to control the processes starting up or being executed on your system.

▼ How to Get Basic Information About Process Classes

To display process class and scheduling parameters:

♦ **Type** `priocntl -l` **and press Return.**

For background information about process classes, turn to "Process Priority Levels" on page 116.

Example of Displaying Process Classes

The listing below shows which classes are configured on your system, and the user priority range for the timesharing class. The possible classes are:

- System (SYS)
- Interactive (IA)
- Real-time (RT)
- Timesharing (TS)
  - The user-supplied priority ranges from –20 to +20.
  - The priority of a process is inherited from the parent process. This is referred to as the *user-mode* priority.
  - The system looks up the user-mode priority in the timesharing dispatch parameter table and adds in any `nice` or `priocntl` (user-supplied) priority to and ensures a 0-59 range to create a *global* priority.

```
# priocntl -l
CONFIGURED CLASSES
==================

SYS (System Class)

TS (Time Sharing)
    Configured TS User Priority Range: -20 through 20
#
```

To display the global priority of a process:

♦ **Type** `ps -ecl` **and press Return.**

▼ How to Designate Priority With `priocntl`

To start a process with a designated priority:

♦ **Type** `priocntl -e -c` *class* `-m` *userlimit* `-p` *pri command_name* **and press Return.**

- The `-e` option means execute the command.
- The `-c` option specifies the class within which to run the process. The default classes are `TS` or `RT`.
- The `-m` option is the maximum amount you can raise or lower your priority, when using `-p` option.
- For a real-time kernel thread, the `-p` option lets you specify the relative priority in the real RT class. For a timesharing process, the `-p` option lets you specify the user-supplied priority which ranges from-20 to +20.

Example of Using `priocntl` *to Designate Priority*

The following example starts the `find` command running with the highest possible user-supplied priority:

```
% su root
# priocntl -e -c TS -m 20 -p 20 find . -name core -print
```

To change the scheduling parameters of a running timeshare process:

♦ **Type the following command**:
`priocntl -s -m` *userlimit* `[-p` *userpriority*`] -i` *idtype idlist*

- The `-s` option lets you set the upper limit on the user priority range and change the current priority.
- The `-m` option is the maximum amount you can raise or lower your priority, when using `-p` option
- The `-p` option (optional) allows you to designate a priority.
- The `-i` option uses a combination of *idtype* and *idlist* to identify the process. The *idtype* specifies the type of `ID`, such as `PID` or `UID`.

Example of Changing the Scheduling Parameters of a Process

The following example executes a command with a 500-millisecond time slice, a priority of 20 in the RT class, and a global priority of 120.

```
% su root
# priocntl -e -c RT -t 500 -p 20 myprog
```

▼ How to Change the Class of a Process

1. **(Optional) Become superuser.**

Note – You must be superuser or working in a real-time shell to change processes from, or to, real-time processes.

---

2. **Type the following command**:
   `priocntl -s -c` *class* `-i` *idtype idlist*

   The `-c` option specifies the class, `TS` or `RT`, to which you are changing the process.

Example of Changing the Class of a Process

The following example changes all the processes belonging to user `15249` to real-time processes:

```
% su
# priocntl –s –c RT –i uid 15249
```

Note – If, as superuser, you change a user process to the real-time class, the user cannot subsequently change the real-time scheduling parameters (using `priocntl -s`).

▼ How to Change the Priority of a Process with the `nice` Command

To lower the priority of a process:

♦ **Type one of the following commands**:

```
/usr/bin/nice command_name
/usr/bin/nice +4 command_name
/usr/bin/nice –10 command_name
```

These examples lower the priority of the command, *command_name*, by raising the nice number. The first two commands increase the nice number by four units (the default); and the second two commands increase the nice by ten units (the default), lowering the priority of the process.

To raise the priority of a process:

♦ **As root, type one of the following commands:**

```
/usr/bin/nice -10 command_name
/usr/bin/nice --10 command_name
```

Both these examples raise the priority of the command, *command_name*, by lowering the nice number. Note that in the second case, the two minus signs are required. The first minus sign is the option sign, and the second minus sign indicates a negative number.

Monitoring Performance 8

This chapter describes the tools that enable you to monitor system performance. These include reporting tools such as:

- `sar` and `sadc`, which collect and report information about system activity
- `vmstat` and `iostat`, which display statistics on system resources
- `df`, which monitors disk usage; and the performance meter, which provides a dynamic display of various system activities.

If you want to go directly to the instructions, use the following table to find the page where instructions for the specific tasks begin.

8

About Monitoring Performance

While your computer is running, counters in the operating system are incremented to keep track of various system activities. System activities that are tracked are:

- Central processing unit (CPU) utilization
- Buffer usage
- Disk and tape input/output (I/O) activity
- Terminal device activity
- System call activity
- Context switching
- File access
- Queue activity
- Kernel tables
- Interprocess communication
- Paging
- Free memory and swap space
- Kernel Memory Allocation (KMA)

The following sections describe tools and commands that help you monitor performance.

The `sar` Command

Use the `sar` command to:

- Organize and view data about system activity
- Access system activity data on a special request basis
- Generate automatic reports to measure and monitor system performance, and special request reports to pinpoint specific performance problems. "Automatic Collection of System Activity Data" on page 133 describes these tools.

The `vmstat` *Command*

The `vmstat` command reports virtual memory statistics and shows CPU load, paging, number of context switches, device interrupts, and system calls.

The following example shows the `vmstat` display of statistics gathered at five-second intervals.

```
example% vmstat 5
 procs     memory            page                  disk       faults      cpu
 r b w  swap  free  re  mf  pi  po  fr  de sr f0 s3 -- --  in  sy  cs us sy id
 0 0 8 28312   668   0   9   2   0   1   0  0  0  1  0  0  10  61  82  1  2 97
 0 0 3 31940   248   0  10  20   0  26   0 27  0  4  0  0  53 189 191  6  6 88
 0 0 3 32080   288   3  19  49   6  26   0 15  0  9  0  0  75 415 277  6 15 79
 0 0 3 32080   256   0  26  20   6  21   0 12  1  6  0  0 163 110 138  1  3 96
 0 1 3 32060   256   3  45  52  28  61   0 27  5 12  0  0 195 191 223  7 11 82
 0 0 3 32056   260   0   1   0   0   0   0  0  0  0  0  0   4  52  84  0  1 99
```

The fields in the `vmstat` report have the following meanings:

`procs` reports the following states:

- `r` The number of kernel threads in the dispatch queue
- `b` Blocked kernel threads waiting for resources
- `w` Swapped out LWPs waiting for processing resources to finish

`memory` reports on usage of real and virtual memory:

- `swap` Available swap space
- `free` Size of the free list

`page` reports on page faults and paging activity, in units per second:

- `re` Pages reclaimed
- `mf` Minor and major faults
- `pi` Kbytes paged in
- `po` Kbytes paged out
- `fr` Kbytes freed
- `de` Anticipated memory needed by recently swapped-in processes
- `sr` Pages scanned by page daemon (not currently used)

If `sr` does not equal zero, the page daemon has been running.

`disk` reports the number of disk operations per second. This field can show data on up to four disks.

`faults` reports the trap/interrupt rates (per second):

- `in` Interrupts per second
- `sy` System calls per second
- `cs` CPU context switch rate

`cpu` reports on the use of CPU time:

- `us` User time
- `sy` System time
- `id` Idle time

The `vmstat` command can also display statistics on swapping, cache flushing, and interrupts.

System Events

Run `vmstat -s` to show the total of various system events that have taken place since the system was last booted.

Swapping

Run `vmstat -S` to show swapping statistics in addition to paging statistics. The additional fields are:

- `si` Average number of LWPs swapped in per second
- `so` Number of whole processes swapped out

Note – The `vmstat` command truncates the output of both of these fields. Use the `sar` command to display a more accurate accounting of swap statistics.

Cache Flushing

Run `vmstat -c` to show cache flushing statistics for a virtual cache. It shows the total number of cache flushes since the last boot. The cache types are:

- `usr` User
- `ctx` Context
- `rgn` Region
- `seg` Segment
- `pag` Page
- `par` Partial-page

Interrupts

Run `vmstat -i` to show interrupts per device.

```
example% vmstat -i
interrupt         total     rate
--------------------------------
clock         104638405      100
esp0            2895003        2
fdc0                  0        0
--------------------------------
Total         107533408      102
```

8

The `iostat` *Command*

The `iostat` command reports statistics about disk input and output, and produces measures of throughput, utilization, queue lengths, transaction rates, and service time.

The following example shows disk statistics gathered every five seconds:

```
example% iostat 5
      tty          fd0           sd3       cpu
 tin tout bps tps serv  bps tps serv  us sy wt  id
   0    1   0   0    0    1   0 5640   0  1  0  98
   0   10   0   0    0    0   0    0   0  1  0  99
   0   10   0   0    0    0   0    0   0  1  0  99
   0   10   0   0    0   27   3  319   0  4  9  88
   0   10   0   0    0    2   0 5061   0  0  0  99
   0   10   0   0    0    0   0    0   0  0  0 100
   0   10   0   0    0    0   0    0   0  0  0 100
   0   10   0   0    0    0   0    0   0  0  0 100
   0   10   0   0    0    0   0    0   0  0  0 100
```

The first line of output shows the statistics since the last boot. Each subsequent line shows the interval statistics. The default is to show statistics for the terminal (`tty`), disks (`fd` and `sd`), and CPU (`cpu`).

For each terminal, `iostat` displays:

- `tin` Number of characters in the terminal input queue
- `tout` Number of characters in the terminal output queue

For each disk, `iostat` displays the following information:

- `bps` Blocks per second
- `tps` Transactions per second
- `serv` Average service time, in milliseconds

For the CPU, `iostat` displays the CPU time spent in the following modes:

- `us` In user mode
- `sy` In system mode
- `wt` Waiting for I/O
- `id` Idle

Run `iostat -xtc` to get extended disk statistics.

```
example% iostat -xtc
                            extended disk statistics        tty       cpu
disk    r/s w/s   Kr/s Kw/s   wait actv   svc_t       %w %b   tin tout us sy  wt  id
sd0     0.2 1.7    1.0  9.7   0.0  0.1     39.8        0  3     0   9   1  6  9   85
sd1     0.5 2.5   10.6 21.0   0.0  0.1     26.6        0  5
sd2     0.0 0.2    0.1  0.0   0.0  0.0    157.7        0  0
```

Each disk has a line of output:

- `r/s` Reads per second
- `w/s` Writes per second
- `Kr/s` Kbytes read per second
- `Kw/s` Kbytes written per second
- `wait` Average number of transactions waiting for service (queue length)
- `actv` Average number of transactions actively being serviced
- `svc_t` Average service time, in milliseconds
- `%w` Percentage of time the queue is not empty
- `%b` Percentage of time the disk is busy

The `df` *Command*

The `df` command shows the amount of free disk space on each mounted disk. The *usable* disk space reported by `df` reflects only 90% of full capacity, as the reporting statistics leave a 10% head room above the total available space. This head room normally stays empty for better performance. The percentage of disk space actually reported by `df` is used space divided by usable space. If the file system is above 90% capacity, transfer files to a disk that is not as full by using `cp`, or to a tape by using `tar` or `cpio`; or remove the files.

Use the `df -k` command to display file system information in Kbytes. The following information is given:

- `kbytes` Total size of usable space in the file system
- `used` Amount of space used
- `avail` Amount of space available for use
- `capacity` Amount of space used, as a percent of the total capacity
- `mounted on` Mount point

```
example%  df -k
filesystem            kbytes    used     avail    capacity     mounted on
/dev/dsk/c0t3d0s0      17269    11099    4450     71%          /
/dev/dsk/c0t3d0s6     136045    79818    42627    65%          /usr
/proc                      0    0        0        0            /proc
swap                   40424    0        40416    0            /tmp
```

The `profil` *Command*

`profil` uses CPU statistics to show the amount of time that a program uses. You can analyze a program and identify the functions that consume a high percentage of CPU time. See the reference page for `profil(2)` for more information.

Performance Meter

The Performance Meter (`perfmeter`) offers a way to view and monitor various performance parameters for your system, or for other systems on the network. The performance meters graphically display the status of the following system parameters:

- CPU
- Ethernet packets per second
- Paging activity in pages per second
- Jobs swapped per second
- Number of device interrupts per second
- Disk traffic in transfers per second
- Number of context switches per second
- Average number of runnable processes over the last minute
- Collisions per second detected on the Ethernet
- Errors per second on receiving packets

The main advantage of the performance meters is that they give you an almost immediate feedback on system performance. You can execute a command or a program, and watch the graph to see if CPU usage makes a big jump. Or you can display the load of several hosts on the network, and choose the one with the smallest load on which to run a CPU-intensive program.

Figure 8-1 shows a performance meter displaying CPU activity, in percent. The curve moves from right to left, showing the CPU load on the system.

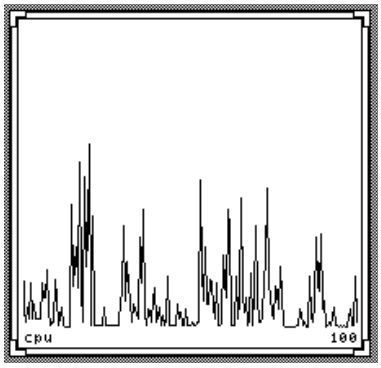


Figure 8-1 Performance Meter Display of CPU Activity

Automatic Collection of System Activity Data

Three commands are involved in automatic system activity data collection: `sadc`, `sa1`, and `sa2`.

For instructions on how to set up automatic monitoring, turn to "How to Set Up Automatic Data Collection" on page 154.

The `sadc` data collection utility periodically collects data on system activity and saves it in a file in binary format—one file for each 24-hour period. You can set up `sadc` to run periodically (usually once each hour), and whenever the system boots to multiuser mode. The data files are placed in the directory `/usr/adm/sa`. Each file is named `sa`*dd*, where *dd* is the current date. The format of the command is as follows:

`/usr/lib/sa/sadc [`*t n*`] [`*ofile*`]`

The command samples *n* times with an interval of *t* seconds (*t* should be greater than 5 seconds) between samples. It then writes, in binary format, to the file *ofile*, or to standard output. If *t* and *n* are omitted, a special file is written once.

Running sadc *When Booting*

The sadc command should be run at system boot time in order to record the statistics from when the counters are reset to zero. To make sure that sadc is run at boot time, the /etc/init.d/perf file must contain a command line that writes a record to the daily data file.

The command entry has the following format:

```
su sys -c "/usr/lib/sa/sadc /usr/adm/sa/sa‘date +5d‘"
```

Running sadc *Periodically with* sa1

To generate periodic records, you need to run sadc regularly. The simplest way to do this is by putting a line into the /var/spool/cron/sys file, which calls the shell script, sa1. This script invokes sadc and writes to the daily data files, /var/adm/sa/sa*dd*. It has the following format:

/usr/lib/sa/sa1 [*t n*]

The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds. If these arguments are omitted, the records are written only one time.

Producing Reports With sa2

There is another shell script, sa2, which produces reports rather than binary data files. The sa2 command invokes the sar command and writes the ASCII output to a report file.

Collecting System Activity Data With `sar`

The `sar` command can be used either to gather system activity data itself or to report what has been collected in the daily activity files created by `sadc`.

The `sar` command has the following formats:

`sar [-aAbcdgkmpqruvwy] [-o` *file*`] t [`*n*`]`

`sar [-aAbcdgkmpqruvwy] [-s` *time*`] [-e` *time*`] [-i` *sec*`] [-f` *file*`]`

The `sar` command below samples cumulative activity counters in the operating system every *t* seconds, *n* times. (*t* should be 5 seconds or greater; otherwise, the command itself may affect the sample.) You must specify a time interval between which to take the samples; otherwise, the command operates according to the second format. The default value of *n* is 1. The following example takes two samples separated by 10 seconds. If the `-o` option is specified, samples are saved in *file* in binary format.

```
example% sar -u 10 2
```

Other important information about the `sar` command:

- With no sampling interval or number of samples specified, `sar` extracts data from a previously recorded file, either the one specified by the `-f` option or, by default, the standard daily activity file, `/var/adm/sa/sa`*dd*, for the most recent day.
- The `-s` and `-e` options define the starting and ending times for the report. Starting and ending times are of the form *hh*[*:mm*[*:ss*]] (where *h*, *m*, and *s* represent hours, minutes, and seconds).
- The `-i` option specifies, in seconds, the intervals between record selection. If the `-i` option is not included, all intervals found in the daily activity file are reported.

Table 8-1 lists the `sar` options and their actions.

Table 8-1 `sar` Options

Option	Actions
`-a`	Checks file access operations
`-b`	Checks buffer activity
`-c`	Checks system calls
`-d`	Checks activity for each block device
`-g`	Checks page-out and memory freeing
`-k`	Checks kernel memory allocation
`-m`	Checks interprocess communication
`-p`	Checks swap and dispatch activity
`-q`	Checks queue activity
`-r`	Checks unused memory
`-u`	Checks CPU utilization
`-v`	Checks system table status
`-w`	Checks swapping and switching volume
`-y`	Checks terminal activity
`-A`	Reports overall system performance (same as entering all options)

If no option is used, it is equivalent to calling the command with the `-u` option.

Checking File Access With `sar -a`

The `sar -a` option reports on the use of file-access operations. The operating system routines reported are as follows:

`iget/s`	The number of requests made for inodes that were not in the directory name lookup cache (`dnlc`).
`namei/s`	This is the number of file-system path searches per second. If `namei` does not find a directory name in the `dnlc`, it calls `iget` to get the inode for either a file or directory. Hence, most `igets` are the result of `dnlc` misses.
`dirbk/s`	This is the number of directory block reads issued per second.

The following is an example of `sar -a` output. It illustrates a one-minute sampling interval.

```
Solaris mysys Solaris 2.3 sun4c 08/22/93

14:28:12    iget/s namei/s dirbk/s
14:29:12         0       2       1
14:30:12         0       4       1
14:31:12         0       3       1

Average          0       3       1
```

The larger the values reported, the more time the kernel is spending to access user files. The amount of time reflects how heavily programs and applications are using the file systems. The `-a` option is helpful for viewing how disk-dependent an application is.

Checking Buffer Activity with `sar -b`

The buffer cache is used to cache metadata, which includes inodes, cylinder group blocks, and indirect blocks. The `-b` option reports on the following buffer activities:

`bread/s`	Average number of reads per second submitted to the buffer cache from the disk.
`lread/s`	Average number of logical reads per second from the buffer cache.
`%rcache`	Fraction of logical reads found in the buffer cache (100% minus the ratio of `bread/s` to `lread/s`).
`bwrit/s`	Average number of physical blocks (512 blocks) written from the buffer cache to disk, per second.
`lwrite/s`	Average number of logical writes to the buffer cache, per second.
`%wcache`	Fraction of logical writes found in the buffer cache (100% minus the ratio of `bwrit/s` to `lwrit/s`).
`pread/s`	Average number of physical reads, per second, using character device interfaces.
`pwrit/s`	Average number of physical write requests, per second, using character device interfaces.

The most important entries are the cache hit ratios `%rcache` and `%wcache`, which measure the effectiveness of system buffering. If `%rcache` falls below 90, or if `%wcache` falls below 65, it may be possible to improve performance by increasing the buffer space.

The following is an example of `sar -b` output:

```
14:28:12 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
14:29:12       0      14     100       6      17      67       0       0
14:30:12       0      12      99       6      16      65       0       0
14:31:12       0      12     100       6      16      65       0       0

Average        0      12     100       6      16      66       0       0
```

This example shows that the buffers are not causing any slowdowns, because all the data is within acceptable limits.

Checking System Calls With `sar -c`

The `-c` option reports on system calls in the following categories:

`scall/s`	All types of system calls per second (generally about 30 per second on a busy four- to six-user system).
`sread/s`	`read` system calls per second.
`swrit/s`	`write` system calls per second.
`fork/s`	`fork` system calls per second (about 0.5 per second on a four- to six-user system); this number will increase if shell scripts are running.
`exec/d`	`exec` system calls per second; if `exec/s` divided by `fork/s` is greater than three, look for inefficient `PATH` variables.
`rchar/s`	Characters (bytes) transferred by `read` system calls per second.
`wchar/s`	Characters (bytes) transferred by `write` system calls per second.

Typically, `reads` and `writes` account for about half of the total system calls, although the percentage varies greatly with the activities that are being performed by the system.

The following is an example of `sar -c` output:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
14:29:12      17       2       2    0.28    0.28    2527    1542
14:30:12      25       2       1    0.50    0.47    1624     295
14:31:12      21       2       2    0.35    0.35    1812     703

Average       21       2       2    0.38    0.37    1987     847
```

Checking Disk Activity With `sar -d`

The `sar -d` option reports the activities of disk devices.

`device`	Name of the disk device being monitored.
`%busy`	Percentage of time the device spent servicing a transfer request.
`avque`	The sum of the average wait time plus the average service time.
`r+w/s`	Number of read and write transfers to the device per second.
`blks/s`	Number of 512-byte blocks transferred to the device per second.
`avwait`	Average time, in milliseconds, that transfer requests wait idly in the queue (measured only when the queue is occupied).
`avserv`	Average time, in milliseconds, for a transfer request to be completed by the device (for disks, this includes seek, rotational latency, and data transfer times).

The following two examples illustrate the `sar -d` output. The first example is from a computer with a non-SCSI (Small Computer System Interface, pronounced “scuzzy”) integral disk; that is, a disk that does not use a SCSI interface. This example illustrates data being transferred from a hard disk (`hdsk-0`) to the floppy disk (`fdsk-0`):

```
Solaris mysys Solaris 2.3 sun4c    8/11/93
13:46:28  device %busy avque r+w/s blks/s  avwait  avserv
13:46:58  hdsk-0    6   1.6     3      5    13.8    23.7
          fdsk-0   93   2.1     2      4   467.8   444.0
13:47:28  hdsk-0   13   1.3     4      8    10.8    32.3
          fdsk-0  100   3.1     2      5   857.4   404.1
13:47:58  hdsk-0   17    .7     2     41      .6    48.1
          fdsk-0  100   4.4     2      6  1451.9   406.5
Average   hdsk-0   12   1.2     3     18     8.4    34.7
          fdsk-0   98   3.2     2      5   925.7   418.2
```

The second example is from a computer with SCSI integral disks; that is, disks that use a SCSI interface. The example illustrates data being transferred from one SCSI hard disk (`sd00-0`) to another SCSI integral disk (`sd00-1`):

```
Solaris mysys Solaris 2.3 sun4c    8/11/93
14:16:24  device %busy avque r+w/s blks/s  avwait  avserv
14:16:52 sd00-0      2   1.0     1      3     0.0    17.9
         sd00-1      6   1.1     3      5     2.0    23.9
14:17:21 sd00-0      2   1.0     1      2     0.0    19.6
         sd00-1      6   1.1     3      5     0.2    24.3
14:17:48 sd00-0      3   1.0     1      3     0.3    18.3
         sd00-1      7   1.1     3      5     1.3    25.4
14:18:15 sd00-0      3   1.0     1      3     0.0    17.2
         sd00-1      5   1.0     2      5     0.0    21.6
Average  sd00-0      2   1.0     1      3     0.1    18.2
         sd00-1      6   1.0     3      5     0.9    23.0
```

Note that queue lengths and wait times are measured when there is something in the queue. If `%busy` is small, large queues and service times probably represent the periodic efforts by the system to ensure that altered blocks are written to the disk in a timely fashion.

Checking Page-out and Memory With `sar -g`

The `sar -g` option reports page-out and memory freeing activities (in averages) as follows:

`pgout/s`	The number of page-out requests per second.
`ppgout/s`	The actual number of pages that are paged-out, per second. (A single page-out request may involve paging-out multiple pages.)
`pgfree/s`	The number of pages, per second, that are placed on the free list.
`pgscan/s`	The number of pages, per second, scanned by the page daemon. If this value is high, the page daemon is spending a lot of time checking for free memory. This implies that more memory may be needed.
`%ufs_ipf`	The percentage of `ufs` inodes taken off the free list by `iget` that had reusable pages associated with them. These pages are flushed and cannot be reclaimed by processes. Thus, this is the percentage of `igets` with page flushes. A high value indicates that the free list of inodes is page-bound and the number of `ufs` inodes may need to be increased.

The following is an example of `sar -g` output:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12  pgout/s ppgout/s pgfree/s pgscan/s %ufs_ipf
15:29:13     0.00     0.00     0.35     8.18   0.00
16:29:12     1.20     2.20     3.35     3.40   0.00
```

`sar -g` is a good indicator of whether more memory may be needed. Use the `ps -elf` command to show the number of cycles used by the page daemon. A high number of cycles, combined with high values for `pgfree/s` and `pgscan/s` indicates a memory shortage.

`sar -g` also shows whether inodes are being recycled too quickly, causing a loss of reusable pages.

Checking Kernel Memory Allocation With `sar -k`

The `sar -k` option reports on the following activities of the Kernel Memory Allocator (KMA):

`sml_mem`	The amount of memory, in bytes, that the KMA has available in the small memory request pool (a small request is less than 256 bytes).
`alloc`	The amount of memory, in bytes, that the KMA has allocated from its small memory request pool to small memory requests.
`fail`	The number of requests for small amounts of memory that failed.
`lg_mem`	The amount of memory, in bytes, that the KMA has available in the large memory request pool (a large request is from 512 bytes to 4 Kbytes).
`alloc`	The amount of memory, in bytes, that the KMA has allocated from its large memory request pool to large memory requests.
`fail`	The number of failed requests for large amounts of memory.
`ovsz_alloc`	The amount of memory allocated for oversized requests (those greater than 4 Kbytes); these requests are satisfied by the page allocator—thus, there is no pool.
`fail`	The number of failed requests for oversized amounts of memory.

The KMA allows a kernel subsystem to allocate and free memory as needed. Rather than statically allocating the maximum amount of memory it is expected to require under peak load, the KMA divides requests for memory into three categories: small (less than 256 bytes), large (512 to 4 Kbytes), and oversized (greater than 4 Kbytes). It keeps two pools of memory to satisfy small and large requests. The oversized requests are satisfied by allocating memory from the system page allocator.

If you are investigating a system that is being used to write drivers or STREAMS that use KMA resources, then `sar -k` will likely prove useful. Otherwise, you will probably not need the information it provides. Any driver or module that uses KMA resources, but does not specifically return the resources before it exits, can create a memory leak. A memory leak causes the

amount of memory allocated by KMA to increase over time. Thus, if the `alloc` fields of `sar -k` increase steadily over time, there may be a memory leak. Another indication of a memory leak is failed requests. If this occurs, then it is likely that a memory leak has caused KMA to be unable to reserve and allocate memory.

If it appears that a memory leak has occurred, you should check any drivers or STREAMS that may have requested memory from KMA and not returned it.

The following is an example of `sar -k` output:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12  sml_mem  alloc  fail  lg_mem  alloc  fail ovsz_alloc fail
14:29:12   95232   73472     0  311296  198656    0     180224    0
14:30:12   95232   75120     0  311296  198656    0     180224    0
14:31:12   95232   73600     0  311296  197632    0     180224    0

Average    95232   74064     0  311296  198314    0     180224    0
```

Checking Interprocess Communication With `sar -m`

The `sar -m` option reports interprocess communication activities.

`msg/s`	The number of message operations (sends and receives) per second.
`sema/s`	The number of semaphore operations per second.

An example of `sar -m` output follows:

```
Solaris mysys 2.0 sun4c    08/22/93

14:28:12   msg/s  sema/s
14:29:12    0.00    0.00
14:30:12    0.00    0.00
14:31:12    0.00    0.00

Average     0.00    0.00
```

These figures will usually be zero (0.00), unless you are running applications that use messages or semaphores.

Checking Page-in Activity With `sar -p`

The `sar -p` option reports page-in activity which includes protection and translation faults. It reports the following statistics:

`atch/s`	The number of page faults, per second, that are satisfied by reclaiming a page currently in memory (attaches per second). Instances of this include reclaiming an invalid page from the free list and sharing a page of text currently being used by another process (for example, two or more processes accessing the same program text).
`pgin/s`	The number of times, per second, that file systems receive page-in requests.
`ppgin/s`	The number of pages paged in, per second. A single page-in request, such as a soft-lock request (see `slock/s`), or a large block size, may involve paging-in multiple pages.
`pflt/s`	The number of page faults from protection errors. Instances of protection faults are illegal access to a page and "copy-on-writes." Generally, this number consists primarily of "copy-on-writes."
`vflt/s`	The number of address translation page faults, per second. These are known as validity faults, and occur when a valid process table entry does not exist for a given virtual address.
`slock/s`	The number of faults, per second, caused by software lock requests requiring physical I/O. An example of the occurrence of a soft-lock request is the transfer of data from a disk to memory. The system locks the page that is to receive the data, so that it cannot be claimed and used by another process.

The following is an example of `sar -p` output:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12  atch/s  pgin/s ppgin/s  pflt/s  vflt/s slock/s
14:29:12    1.17   12.87   12.87    5.67   11.28    1.15
14:30:12    1.67    7.08    7.08    9.12    6.33    0.67
14:31:12    1.37   12.48   12.48    6.83   10.78    1.03

Average     1.40   10.81   10.81    7.21    9.46    0.95
```

Checking Queue Activity With `sar -q`

The `sar -q` option reports the average queue length while the queue is occupied, and the percentage of time that the queue is occupied.

`runq-sz`	The number of kernel threads in memory waiting for a CPU to run. Typically, this value should be less than 2. Consistently higher values mean that the system may be CPU-bound.
`%runocc`	The percentage of time the dispatch queues are occupied.
`swpq-sz`	The average number of swapped out LWPs.
`%swpocc`	The percentage of time LWPs are swapped out.

An example of `sar -q` output follows:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12 runq-sz %runocc swpq-sz %swpocc
14:29:12     1.2      53    1       100
14:30:12     1.3      38
14:31:12     1.1      37

Average      1.2      43
```

Note – The number of LWPs swapped out may greater than zero even if the system has an abundance of free memory. This happens when a sleeping LWP is swapped out and has not been awakened (for example, a process or LWP sleeping, waiting for the keyboard or mouse input).

If `%runocc` is high (greater than 90 percent) and `runq-sz` is greater than 2, the CPU is heavily loaded and response is degraded. In this case, additional CPU capacity may be required to obtain acceptable system response.

Checking Unused Memory With `sar -r`

The `-r` option records the number of memory pages and swap-file disk blocks that are currently unused.

`freemem` This shows the average number of memory pages available to user processes over the intervals sampled by the command. Page size is machine-dependent.

`freeswap` This shows the number of 512-byte disk blocks available for page swapping.

An example of `sar -r` output follows:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12 freemem freeswap
14:29:12     268    3034
14:30:12     351    3009
14:31:12     297    3033

Average      306    3025
```

Checking CPU Utilization With `sar -u`

CPU utilization is listed by `sar -u`. (`sar` without any options is equivalent to `sar -u`.) At any given moment, the processor is either busy or idle. When busy, the processor is in either user or system mode. When idle, the processor is either waiting for I/O completion or "sitting still" with no work to do.

The `sar -u` command displays the following information:

- `%sys` lists the percentage of time that the processor is in system mode
- `%user` lists the percentage of time that the processor is in user mode
- `%wio` lists the percentage of time the processor is idle and waiting for I/O completion
- `%idle` lists the percentage of time the processor is idle and is not waiting for I/O

A high `%wio` generally means a disk slowdown has occurred.

The following is an example of `sar -u` output:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12    %usr    %sys    %wio   %idle
14:29:12      22      27      18      32
14:30:12       6      24      13      57
14:31:12       8      28      19      45

Average       12      27      17      45
```

Checking System Table Status With `sar -v`

The `-v` option reports the status of the process table, inode table, file table, and shared memory record table.

`proc-sz`	The number of process entries (`proc` structs) currently being used, or allocated in the kernel.
`inod-sz`	The total number of inodes in memory verses the maximum number of inodes allocated in the kernel. This is not a strict high water mark; it can overflow.
`file-sz`	The size of the open system file table. The `sz` is given as 0, since space is allocated dynamically for the file table.
`ov`	This is the number of times a table has overflowed (reported for the three tables listed above).
`lock-sz`	The number of shared memory record table entries currently being used or allocated in the kernel. The `sz` is given as 0 because space is allocated dynamically for the shared memory record table.

An example of `sar -v` output follows:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12 proc-sz ov inod-sz ov file-sz ov lock-sz
14:29:12  28/200  0 297/300  0  63/0    0  6/0
14:30:12  30/200  0 297/300  0  65/0    0  6/0
14:31:12  28/200  0 296/300  0  63/0    0  6/0
```

This example shows that all tables are large enough to have no overflows. These tables are all dynamically allocated based on the amount of physical memory.

Checking Swap Activity With `sar -w`

The `-w` option reports swapping and switching activity. The following are some target values and observations:

`swpin/s` The number of LWP transfers into memory per second.

`bswin/s` The number of 512-byte blocks transferred for swap-ins per second.

Note – All process swap-ins include process initialization.

`swpot/s` The average number of processes swapped out of memory, per second. If the number is greater than 1, you may need to increase memory.

`bswot/s` The number of blocks transferred for swap-outs per second.

`pswch/s` The number of kernel thread switches per second.

An example of `sar -w` output follows:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12 swpin/s pswin/s swpot/s pswot/s pswch/s
14:29:12    0.00     0.0    0.00     0.0      22
14:30:12    0.00     0.0    0.00     0.0      12
14:31:12    0.00     0.0    0.00     0.0      18

Average     0.00     0.0    0.00     0.0      18
```

Checking Terminal Activity with `sar -y`

The `-y` option monitors terminal device activities. If you have a lot of terminal I/O, you can use this report to determine if there are any bad lines. The activities recorded are defined as follows:

`rawch/s`	Input characters (raw queue), per second.
`canch/s`	Input characters processed by canon (canonical queue) per second.
`outch/s`	Output characters (output queue) per second.
`rcvin/s`	Receiver hardware interrupts per second.
`xmtin/s`	Transmitter hardware interrupts per second.
`mdmin/s`	Modem interrupts per second.

The number of modem interrupts per second (`mdmin/s`) should be close to zero, and the receive and transmit interrupts per second (`xmtin/s` and `rcvin/s`) should be less than or equal to the number of incoming or outgoing characters, respectively. If this is not the case, check for bad lines.

An example of `sar -y` output follows:

```
Solaris mysys Solaris 2.3 sun4c    08/22/93

14:28:12 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
14:29:12       0       1     157       1       3       0
14:30:12       0       2      34       2       2       0
14:31:12       0       1      11       1       2       0

Average        0       1      67       1       2       0
```

Checking Overall System Performance With `sar -A`

The `-A` option provides a view of overall system performance. Use it to get a more global perspective. If data from more than one time segment is shown, the report includes averages.

8

Instructions for Monitoring Performance

This section describes how to set up the `sadc` and `sar` utilities to monitor the performance of your system.

▼ How to Set Up Automatic Data Collection

If you need background information about the `sadc` command, turn to "Automatic Collection of System Activity Data" on page 133.

1. **As root, open the file** `/etc/init.d/perf` **for editing.**

2. **Verify that the following line is uncommented:**

   ```
   # su sys -c "/usr/lib/sa/sadc /var/adm/sa/sa‘date +%d‘"
   ```

   This version of the `sadc` command writes a special record that marks the time when the counters are reset to zero (boot time). The output of `sadc` is put into the file `sa`*dd* (where *dd* is the current date), which acts as the daily system activity record.

3. **Open the file** `/var/spool/cron/crontabs/sys` **for editing.**

4. **Uncomment the following lines:**

   ```
   # 0 * * * 0-6 /usr/lib/sa/sa1
   # 20,40 8-17 * * 1-5 /usr/lib/sa/sa1
   ```

   The first entry writes a record to `/var/adm/sa/sa`*dd* on the hour, every hour, seven days a week.

   The second entry writes a record to `/var/adm/sa/sa`*dd* twice each hour during peak working hours: at 20 minutes and 40 minutes past the hour, from 8 a.m. to 5 p.m., Monday through Friday.

   Thus, these two `crontab` entries cause a record to be written to `/var/adm/sa/sa`*dd* every 20 minutes from 8 a.m. to 5 p.m., Monday through Friday, and every hour on the hour otherwise. You can change these defaults to meet your needs.

▼ How to Display Statistics With `vmstat`

For background information about the fields that `vmstat` displays, turn to "The vmstat Command" on page 127.

To display statistics gathered at five-second intervals:

♦ **Type `vmstat 5` and press Return.**

To display statistics since the last boot:

♦ **Type `vmstat -s` and press Return.**

To display swapping activity:

♦ **Type `vmstat -S` and press Return.**

To display cache flushing statistics:

♦ **Type `vmstat -c` and press Return.**

▼ How to Display I/O Statistics With `iostat`

For background information about the fields that `iostat` displays, turn to "The iostat Command" on page 130.

To display disk statistics gathered at five-second intervals:

♦ **Type `iostat 5` and press Return.**

8

Reference Material for Monitoring Performance

Table 8-2 lists the `sar` command options according to the type of information that they provide:

Table 8-2 List of Activities Reported by `sar`

Resource	Option	Activity Reported
I/O	`-a`	File access
	`-b`	Buffers
	`-d`	Disk transfers
	`-y`	Terminal activity
CPU	`-c`	System calls
	`-m`	Interprocess communication
	`-q`	Queue activity
	`-u`	CPU mode
Memory	`-g`	Page-out activity
	`-k`	Memory allocation
	`-p`	Page-in activity
	`-r`	Unused memory
	`-v`	System tables
	`-w`	Swapping

A Guide to Network Performance 9

This chapter describes the tools that can help you identify problems in physical connections, network traffic, NFS performance, and server slowdown. These tools include:

- `ping`
- `spray`
- `snoop`
- `netstat`
- `nfsstat`

`ping` *Command*

Use `ping` to look at the response of hosts on the network. The simplest version of `ping` sends a single packet to a host on the network. If it receives the correct response, it prints the message *host* `is alive`.

```
% ping elvis
elvis is alive.
```

With the –s option, ping sends one datagram per second to a host. It then prints each response and the time it took for the round trip. For example:

```
% ping –s pluto
64 bytes from pluto (123.456.78.90): icmp_seq=0. time=10. ms
64 bytes from pluto (123.456.78.90): icmp_seq=5. time=0. ms
64 bytes from pluto (123.456.78.90): icmp_seq=6. time=0. ms
^C
----pluto PING Statistics----
8 packets transmitted, 8 packets received, 0% packet loss
round-trip (ms) min/avg/max = 0/2/10
%
```

If you suspect a physical problem, you can use ping to find the response time of several hosts on the network. If the response from one host is not what you would expect, you can investigate that host.

spray *Command*

Use spray to test the reliability of your packet sizes. It can tell you whether packets are being delayed, or dropped.

The following example sends 100 packets to a host (–c 100) with each packet having a size of 2048 bytes (–l 2048). The packets are sent with a delay time of 20 microseconds between each burst (–d 20). If you don't use a delay, you may run out of buffers.

```
% spray -c 100 -d 20 0 –l 2048 pluto
spray -c 100 -d 20 -l 2048 iris
sending 100 packets of length 2048 to pluto ...
no packets dropped by pluto
279 packets/sec, 573043 bytes/sec
```

Physical problems could be caused by:

- Loose cables or connectors
- Improper grounding
- Missing termination
- Signal reflection

snoop *Command*

Use `snoop` to capture packets from the network and trace the calls from each client to each server. It provides accurate time stamps that allow some network performance problems to be isolated quickly. For more information, see the reference page for `snoop(1M)`.

Dropped packets could be caused by insufficient buffer space, or an overloaded CPU.

netstat *Command*

The `netstat` command displays statistics about the network interfaces. With the `–i` option, `netstat` displays the state of the interfaces used for TCP/IP traffic.

```
pluto% netstat -i
Name  Mtu   Net/Dest  Address    Ipkts     Ierrs  Opkts     Oerrs  Collis    Queue
ie0   1500  software  pluto      28926802  16235  39590055  22109  13690055  0
lo0   1536  loopback  localhost  850886    0      850886    0      0         0
```

This display shows how many packets a machine has transmitted and received on each interface. A machine with active network traffic should show both `Ipkts` and `Opkts` continually increasing.

Calculate the network collisions rate by dividing the number of collision counts (`Collis`) by the number of out packets (`Opkts`). In the above example, the collision rate is 3.5 percent. A network-wide collision rate greater than 5 to 10 percent can indicate a problem.

Calculate the input packet error rate by dividing the number of input errors by the total number of input packets (`Ierrs/Ipkts`). The output packet error rate is the number of output errors divided by the total number of output packets (`Oerrs/Opkts`). If the input error rate is high (over 0.25 percent), the host may be dropping packets.

With the `-s` option, `netstat` displays the per-protocol statistics for the `UDP`, `TCP`, `ICMP`, and `IGMP` protocols. This example shows part of the `netstat -s` display:

```
UDP                                                                    0
       udpInDatagrams       =   61321    udpInErrors
       udpOutDatagrams      =   6783

       tcpRtoAlgorithm      =    4       ttcpRtoMin      =    50
       tcpRtoMax            =    60000   tcpMaxConn      =    -1
       .
       .
       .

IP     ipForwarding         =   1        ipDefaultTT     =   255
       ipInReceives         =   13429    ipInHdrError    =     0
                                8        s
       .
       .
       .

ICMP   icmpInMsgs           =   116      icmpInErrors    =  0
       icmpInCksumErrs      =    0       icmpInUnknow
                                         n



IGMP:
       0 messages received
       0 messages received with too few bytes




       0 membership reports sent
```

The –r option of netstat displays the IP routing table.

```
Routing Table:
Destination          Gateway          Flags    Ref     Use   Interface
-----------------    ------------     -----    ---     ---   --------
localhost            localhost        UGHD     0         0   lo0
earth-bb             sleepy           U        3         1
software             pluto            U        3       147   lo0
224.0.0.0            pluto            UG       3         0   lo0
default              mars             UG       0        18
default              earth            UG       0        30
default              venus            UG       0        18
default              neptune          UG       0        26
default              saturn           UG       0         3
```

The fields in the netstat report mean:

- Flags shows the state of the route. The letters under this heading mean:
  - U The route is up.
  - G The route is through a gateway.
  - H The route is to a host.
  - D The route was dynamically created using a redirect.
- Ref shows the current number of routes sharing the same link layer.
- Use indicates the number of packets sent out.
- Interface lists the network interface used for the route.

nfsstat *Command*

The nfsstat utility can be used to identify NFS problems. nfsstat displays a summary of server and client statistics. Run nfsstat -c to show client statistics, and nfsstat -s to show server statistics.

The following example displays RPC and NFS data for the client, `pluto`.

```
pluto% nfsstat -c

Client rpc:
calls  badcalls  retrans badxid  timeout  wait     newcred timers
6888   123       10      51      101      0        0       138

Client nfs:
calls  badcalls  nclget  nclcreate
6765   0         6765    0

null   getattr   setattr root   lookup    readlink  read
0  0%  1364 20%   4 0%   0 0%   1643 24%   928 13%  1622%

wrcache write   create   remove  rename   link      symlink
0 0%    14 0%   11 0%    1 0%     0 0%     0 0%     0 0%

mkdir   rmdir   readdir  fsstat
1 0%    0 0%    2535 37% 10 21%
```

The `nfsstat` display shows the following fields:

- `calls` shows the total number of calls sent.
- `badcalls` is the total number of calls rejected by RPC.
- `retrans` is the total number of retransmissions. For this client, the number of retransmissions is less than 1 percent (10 time-outs out of 6888 calls). These may be caused by temporary failures. Higher rates may indicate a problem.
- `badxid` is the number of times that a duplicate acknowledgment was received for a single NFS request.
- `timeout` is the number of calls that timed out.
- `wait` is the number of times a call had to wait because no client handle was available.
- `newcred` is the number of times the authentication information had to be refreshed.

- `timers` is the number of times the time-out value was greater than or equal to the specified time-out value for a call.
- `readlink` is the number of times a `read` was made to a symbolic link. If this number is high (over 10 percent), it could mean that there are too many symbolic links.

The NFS distributed file service uses a remote procedure call (RPC) facility which translates local commands into requests for the remote host. The remote procedure calls are synchronous. That is, the client application is blocked or suspended until the server has completed the call and returned the results. One of the major factors affecting NFS performance is the retransmission rate.

If the file server cannot respond to a client's request, the client retransmits the request a specified number of times before it quits. Each retransmission imposes system overhead, and increases network traffic. Excessive retransmissions can cause network performance problems. If the retransmission rate is high, you could look for:

- Overloaded servers that take too long to complete requests
- An Ethernet interface dropping packets
- Network congestion which slows the packet transmission

Use `netstat -m` to display network statistics for each file system.

```
pluto% nfsstat -m
/usr/man from pluto:/export/svr4/man
 Flags:   hard,intr,dynamic read size=8192, write size=8192,  retrans = 5
 Lookups: srtt=14 (35ms), dev=4 (20ms), cur=3 (60ms)
 Reads:   srtt=17 (42ms), dev=6 (30ms), cur=5 (100ms)
 All:     srtt=15 (37ms), dev=7 (35ms), cur=5 (100ms)
```

This display reports the following statistics, in milliseconds:

- `srtt` is the smoothed average of the round-trip times.
- `dev` is the average deviations.
- `cur` is the current "expected" response time.

If you suspect that the hardware components of your network are creating problems, you need to look carefully at the cabling and connectors.

9

Setting Up and Maintaining Accounting 10

The SunOS 5.x accounting utilities are a group of programs that collect and record data about system usage, and then provide full reports of that data. The accounting utilities can be used for:

- Monitoring system usage
- Troubleshooting
- Locating and correcting performance problems
- Maintaining system security

The accounting utilities provide C language programs and shell scripts that organize the data into summary files and reports.

This chapter describes how the accounting utilities work, how to set up accounting on your system, and how to read the reports generated by the programs.

Overview of Accounting

Once it has been set up, system accounting runs mostly on its own. (For instructions on setting up an accounting system, see “Setting Up Accounting” on page 168.) The shell scripts that generate accounting reports reside in the `/usr/adm/acct` and `/usr/lib/acct` directories. Setting up automatic accounting involves putting these scripts into the `crontab` file so that `cron` will invoke them automatically.

The following is an overview of how accounting works.

- Between system startup and shutdown, raw data about system use (such as logins, processes run, and data storage) are collected in accounting files.
- Periodically (usually once a day), the `/usr/lib/acct/runacct` program processes the various accounting files and produces both cumulative summary files and daily accounting reports. The daily reports are printed by the `prdaily` program.
- The cumulative summary files generated by `runacct` can be processed and printed monthly by executing the `monacct` program. The summary reports produced by `monacct` provide an efficient means for billing users on a monthly or other fiscal basis.

Types of Accounting

The daily accounting can help you do four types of accounting: *connect accounting*, *process accounting*, *disk accounting*, and *fee calculations*.

Connect Accounting

Connect accounting enables you to determine the following:

- The length of time a user was logged in
- How the `tty` lines are being used
- The number of reboots on your system
- The frequency with which the accounting software was turned off and on

To provide this information, the system stores records of time adjustments, boot times, times the accounting software was turned off and on, changes in run levels, the creation of user processes (`login` processes and `init` processes), and the deaths of processes. These records (produced from the output of system programs such as `date`, `init`, `login`, `ttymon`, and `acctwtmp`) are stored in the file, `/var/adm/wtmp`. Entries in the `wtmp` file may contain the following information: a user's login name, a device name, a process ID, the type of entry, and a time stamp denoting when the entry was made.

Process Accounting

Process accounting enables you to keep track of the following data about each process run on your system:

- The user and group IDs of those using the process
- The beginning and elapsed times of the process
- The CPU time for the process (user time and system time)
- The amount of memory used
- The commands run
- The `tty` controlling the process

Every time a process dies, the `exit` program collects this data and writes it to the file `/var/adm/pacct`.

The `pacct` file has a default maximum size of 500 blocks that is enforced by the accounting shell script, `ckpacct` (normally run as a `cron` job). If `ckpacct` finds that `/var/adm/pacct` is larger than 500 blocks, it moves the file to `/var/adm/pacct`*n*, where *n* is the next unused incremental number.

Disk Accounting

Disk accounting enables you to gather and format the following data about the files each user has on disks:

- The name and ID of the user
- The number of blocks used by the user's files

This data is collected by a shell script called `dodisk`.

`dodisk` invokes the commands `acctdusg` and `diskusg`, which gather information for each file in the system.

`acctdusg` gathers all the disk accounting information. Each time it is invoked, this command can process a maximum of 3000 users. The slow mode syntax is:

`/usr/lib/acct/dodisk –o` *mountpoints*

If no mount points are specified, the root mount point is used.

Caution – Information gathered by running `dodisk` is stored in the `/var/adm/acct/nite/disktacct` file. This information is overwritten the next time `dodisk` is run. Therefore, avoid running `dodisk` twice in the same day.

`diskusg` may overcharge for files that are written in random access fashion, which may create holes in the files. This is because `diskusg` does not read the indirect blocks of a file when determining its size. Rather, `diskusg` determines the size of a file by looking at the *di_size* value of the inode.

Fee Calculations

If you charge your users for special services, such as restoring files and remote printing, you may want to use a program called `chargefee` to maintain service accounts. Fees charged to customers are recorded in a file called `/var/adm/fee`. Each entry in the file consists of a user's login name, user ID, and the fee.

Accounting Programs

All the accounting shell scripts and binary accounting programs are stored in `/usr/lib/acct`. The `acctcom` program is stored in `/usr/bin`. These programs, which are owned by `bin` (except for `accton`, which is owned by root), perform various functions. For example, `/usr/lib/acct/startup` helps initiate the accounting process when the system enters multiuser mode. The `chargefee` program is used to charge a particular user for a special service, such as restoring a file from tape. Other essential programs in the `/usr/lib/acct` directory include `monacct`, `prdaily`, and `runacct`. These and other programs are discussed in more detail in the following sections.

Setting Up Accounting

To set up system accounting to run while the system is in multiuser mode (system state 2), you need to create or modify four files:

- `/etc/rc0.d/K22acct` (create)
- `/etc/rc2.d/S22acct` (create)
- `/var/spool/cron/crontabs/adm` (modify)
- `/var/spool/cron/crontabs/root` (modify)

If you want accounting to be shut off during shutdown, link `/etc/rc0.d/k22acct` to `/etc/init.d/acct`.

- **Type the following command**:
  `ln –s /etc/init.d/acct /etc/rc0.d/K22acct`

If you want accounting to be turned on when the system is in multiuser mode (system state 2), link `/etc/rc2.d/S22acct` to `/etc/init.d/acct`.

- **Type the following command**:
  `ln -s /etc/init.d/acct /etc/rc2.d/S22acct`

Most of the `cron` entries needed for accounting are put into a database called `/var/spool/cron/crontabs/adm`. The sample entries in this database run `ckpacct` periodically, `runacct` daily, and `monacct` on a fiscal basis. You can vary the frequencies. Be sure to append this information to the file to avoid destroying any entries already present. For the `adm crontab`, assign `root` as the owner, `sys` as the group, and `644` as the permissions mode.

```
-------------------entries for adm crontab-----------------------------
#Min  Hour     Day      Month    Day      Command
#              of                Of
#              Month             Week
#-------------------------------------------------------------------------------
0     *        *        *        *        /usr/lib/acct/ckpacct
30    2        *        *        *        /usr/lib/acct/runacct 2> /var/adm/acct/nite/fd2log
30    9        *        *        5        /usr/lib/acct/monacct
-------------------------------------------------------------------------
```

Append the entry for `dodisk` to the `root crontab`, `/var/spool/cron/crontabs/root`. A sample is shown below.

```
--------------------entry for root crontab-------------------------------
#Min  Hour     Day      Month    Day      Command
#              of                Of
#              Month             Week
#-------------------------------------------------------------------------------
30  22  *   *   4   /usr/lib/acct/dodisk
```

Once these entries are in the database and the accounting programs have been installed, accounting should run automatically.

10

Daily Accounting

Here is a step-by-step summary of how SunOS system accounting works:

1. When the system is switched into multiuser mode, the `/usr/lib/acct/startup` program is executed. The `startup` program executes several other programs that invoke accounting.
2. The `acctwtmp` program adds a "boot" record to `/var/adm/wtmp`. In this record, the system name is shown as the login name in the `wtmp` record. Table 10-1 presents a summary of how the raw accounting data is gathered and where it is stored.

Table 10-1 Raw Accounting Data

File in `/var/adm`	**Information**	**Written By**	**Format**
`wtmp`	Connect sessions	`login`, `init`	`utmp.h`
	Changes	`date`	
	Reboots	`acctwtmp`	
	Shutdowns	`shutacct` shell	
`pacctn`	Processes	Kernel (when the process ends)	acct.h
		`turnacct switch` (creates a new file when the old one reaches 500 blocks)	
`fee`	Special charges	`chargefee`	`acct.h`
`acct/nite/disktacct`	Disk space used	`dodisk`	`tacct.h`

3. The `turnacct` program, invoked with the `on` option, begins process accounting. Specifically, `turnacct` executes the `accton` program with the argument `/var/adm/pacct`.
4. The `remove` shell script "cleans up" the saved `pacct` and `wtmp` files left in the `sum` directory by `runacct`.
5. The `login` and `init` programs record connect sessions by writing records into `/var/adm/wtmp`. Any date changes (using `date` with an argument) are also written to `/var/adm/wtmp`. Reboots and shutdowns using `acctwtmp` are also recorded in `/var/adm/wtmp`.

6. When a process ends, the kernel writes one record per process, in the form of `acct.h`, in the `/var/adm/pacct` file.

Two programs track disk usage by login: `acctdusg` and `diskusg`. They are invoked by the shell script `dodisk`.

Every hour, `cron` executes the `ckpacct` program to check the size of `/var/adm/pacct`. If the file grows past 500 blocks (default), the `turnacct` switch is executed. (The program moves the `pacct` file and creates a new one.) The advantage of having several smaller `pacct` files becomes apparent when trying to restart `runacct` if a failure occurs when processing these records.

If the system is shut down using `shutdown`, the `shutacct` program is executed automatically. The `shutacct` program writes a reason record into `/var/adm/wtmp` and turns off process accounting.

If you provide services on a request basis (such as file restorations), you can keep billing records by login, using the `chargefee` program. It allows you to add a record to `/var/adm/fee` each time a user incurs a charge. The next time `runacct` is executed, this new record is picked up and merged into the total accounting records.

1. `runacct` is executed by `cron` each night. `runacct` processes the accounting files: `/var/adm/pacct`*n*, `/var/adm/wtmp`, `/var/adm/fee`, and `/var/adm/acct/nite/disktacct`, to produce command summaries and usage summaries by login.

2. The `/usr/lib/acct/prdaily` program is executed on a daily basis by `runacct` to write the daily accounting information collected by `runacct` (in ASCII format) in `/var/adm/acct/sum/rprt`.*MMDD*.

3. The `monacct` program should be executed on a monthly basis (or at intervals determined by you, such as the end of every fiscal period). The `monacct` program creates a report based on data stored in the `sum` directory that has been updated daily by `runacct`. After creating the report, `monacct` “cleans up” the `sum` directory to prepare the directory’s files for the new `runacct` data.

10

`runacct` *Program*

The main daily accounting shell procedure, `runacct`, is normally invoked by `cron` outside of prime time hours. The `runacct` shell script processes connect, fee, disk, and process accounting files. It also prepares daily and cumulative summary files for use by `prdaily` and `monacct` for billing purposes.

The `runacct` shell script takes care not to damage files if errors occur. A series of protection mechanisms are used that attempt to recognize an error, provide intelligent diagnostics, and complete processing in such a way that `runacct` can be restarted with minimal intervention. It records its progress by writing descriptive messages into the file `active`. (Files used by `runacct` are assumed to be in the `/var/adm/acct/nite` directory, unless otherwise noted.) All diagnostic output during the execution of `runacct` is written into `fd2log`.

When `runacct` is invoked, it creates the files `lock` and `lock1`. These files are used to prevent simultaneous execution of `runacct`. The `runacct` program prints an error message if these files exist when it is invoked. The `lastdate` file contains the month and day `runacct` was last invoked, and is used to prevent more than one execution per day. If `runacct` detects an error, a message is written to the console, mail is sent to `root` and `adm`, locks are removed, diagnostic files are saved, and execution is ended.

Re-entrant States of the `runacct` *Script*

To allow `runacct` to be restartable, processing is broken down into separate reentrant states. The file `statefile` is used to keep track of the last state completed. When each state is completed, `statefile` is updated to reflect the next state. After processing for the state is complete, `statefile` is read and the next state is processed. When `runacct` reaches the `CLEANUP` state, it removes the locks and ends. States are executed as follows:

`SETUP` The command `turnacct switch` is executed to create a new `pacct` file. The process accounting files in `/var/adm/pacctn` (except for the `pacct` file) are moved to `/var/adm/Spacctn.`*MMDD*. The `/var/adm/wtmp` file is moved to `/var/adm/acct/nite/wtmp.`*MMDD* (with the current time record added on the end) and a new

/var/adm/wtmp is created. closewtmp and utmp2wtmp add records to wtmp.*MMDD* and the new wtmp to account for users currently logged in.

WTMPFIX — The wtmpfix program checks the wtmp.*MMDD* file in the nite directory for accuracy. Because some date changes will cause acctcon to fail, wtmpfix attempts to adjust the time stamps in the wtmp file if a record of a date change appears. It also deletes any corrupted entries from the wtmp file. The fixed version of wtmp.*MMDD* is written to tmpwtmp.

CONNECT — The acctcon program is used to record connect accounting records in the file ctacct.*MMDD*. These records are in tacct.h format. In addition, acctcon creates the lineuse and reboots files. The reboots file records all the boot records found in the wtmp file. CONNECT was previously divided into two steps called CONNECT1 and CONNECT2.

PROCESS — The acctprc program is used to convert the process accounting files, /var/adm/Spacct*n.MMDD*, into total accounting records in ptacct*n.MMDD*. The Spacct and ptacct files are correlated by number so that if runacct fails, the Spacct files will not be processed.

Caution – When restarting runacct in this state, remove the last ptacct file because it will not be complete.

MERGE — The MERGE program merges the process accounting records with the connect accounting records to form daytacct.

FEES — The MERGE program merges ASCII tacct records from the fee file into daytacct.

DISK — If the dodisk procedure has been run, producing the file disktacct, the DISK program merges the file into daytacct and move disktacct to /tmp/disktacct.*MMDD*.

MERGETACCT — The MERGETACCT merges daytacct with sum/tacct, the cumulative total accounting file. Each day, daytacct is saved in sum/tacct.*MMDD*, so that sum/tacct can be re-created if it is corrupted or lost.

`CMS`	The program `acctcms` is run several times. `acctcms` is first run to generate the command summary using the `Spacct`*n* files and write it to `sum/daycms`. The `acctcms` program is then run to merge `sum/daycms` with the cumulative command summary file `sum/cms`. Finally, `acctcms` is run to produce the ASCII command summary files, `nite/daycms` and `nite/cms`, from the files `sum/daycms` and `sum/cms`, respectively. The program `lastlogin` is used to create the log file `/var/adm/acct/sum/loginlog`, the report of when each user last logged in. (If `runacct` is run after midnight, the dates showing the time last logged in by some users will be incorrect by one day.)
`USEREXIT`	Any installation-dependent (local) accounting program can be included at this point. `runacct` expects it to be called `/usr/lib/acct/runacct.local`.
`CLEANUP`	Cleans up temporary files, run `prdaily` and saves its output in `sum/rpt.`*MMDD*, removes the locks, then exits.

`runacct` *Error Messages*

The `runacct` procedure can fail for a variety of reasons, the most common being a system crash, `/var` running out of space, or a corrupted `wtmp` file. If the `active`.*MMDD* file exists, check it first for error messages. If the `active` and `lock` files exist, check `fd2log` for any mysterious messages. See Appendix C, "Error Messages," for an explanation of error messages generated by `runacct`.

Files Produced by `runacct`

The following files produced by `runacct` (found in `/var/adm/acct`) are of particular interest:

`nite/lineuse`	`runacct` calls `acctcon` to gather data on terminal line usage from `/var/adm/acct/nite/tmpwtmp` and writes the data to `/var/adm/acct/nite/lineuse`. `prdaily` uses this data to report line usage. This report is especially useful for detecting bad lines. If the ratio between the number of logouts to logins is greater than about three to one, there is a good possibility that the line is failing.

`nite/daytacct`	This file is the total accounting file for the day in `tacct.h` format.
`sum/tacct`	This file is the accumulation of each day's `nite/daytacct` and can be used for billing purposes. It is restarted each month or fiscal period by the `monacct` procedure.
`sum/daycms`	`runacct` calls `acctcms` to process the data about the commands used during the day. This information is stored in `/var/adm/acct/sum/daycms`. It contains the daily command summary. The ASCII version of this file is `/var/adm/acct/nite/daycms`.
`sum/cms`	This file is the accumulation of each day's command summaries. It is restarted by the execution of `monacct`. The ASCII version is `nite/cms`.
`sum/loginlog`	`runacct` calls `lastlogin` to update the last date logged in for the logins in `/var/adm/acct/sum/loginlog`. `lastlogin` also removes from this file logins that are no longer valid.
`sum/rprt`.*MMDD*	Each execution of `runacct` saves a copy of the daily report that was printed by `prdaily`.

Fixing Corrupted Files

Unfortunately, this accounting system is not foolproof. Occasionally, a file will become corrupted or lost. Some of the files can simply be ignored or restored from the backup. However, certain files must be fixed to maintain the integrity of the accounting system.

Fixing `wtmp` *Errors*

The `wtmp` files seem to cause the most problems in the day-to-day operation of the accounting system. When the date is changed and the system is in multiuser mode, a set of date change records is written into `/var/adm/wtmp`. The `wtmpfix` program is designed to adjust the time stamps in the `wtmp` records when a date change is encountered. However, some combinations of date changes and reboots will slip through `wtmpfix` and cause `acctcon` to fail. The following steps show how to patch up a `wtmp` file.

▼ How to Fix Errors

1. **Type** `cd /var/adm/acct/nite` **and press Return.**
2. **Type** `fwtmp wtmp.`*MMDD* `xwtmp` **and press Return.**
   The `fwtmp` command converts the binary file `wtmp.`*MMDD* to the ASCII file `xwtmp`.
3. **Edit** `xwtmp`**. Delete the corrupted files, or delete all records from the beginning up to the date change.**
4. **Type** `fwtmp –ic xwtmp wtmp.`*MMDD* **and press Return.**
   This step converts the ASCII file `xwtmp` to a binary file, overwriting the corrupted file.

   If the `wtmp` file is beyond repair, create a null `wtmp` file. This will prevent any charging of connect time. As a side effect, the lack of a `wtmp` file prevents `acctprc` from identifying the login that owned a particular process; the process is charged to the owner of the first login in the password file for the appropriate user ID.

Fixing `tacct` *Errors*

If the installation is using the accounting system to charge users for system resources, the integrity of `sum/tacct` is important. Occasionally, mysterious `tacct` records will appear with negative numbers, duplicate user IDs, or a user ID of 65535. First, check `sum/tacctprev`, using `prtacct` to print it. If it looks all right, patch up the latest `sum/tacct.`*MMDD*, then re-create the `sum/tacct` file. The following steps outline a simple patch procedure.

▼ How to Fix `tacct` Errors

1. **Type** `cd /var/adm/acct/sum` **and press Return.**
2. **Type** `acctmerg –v tacct.`*MMDD* `xtacct` **and press Return.**
   The `–v` option converts the contents of `tacct.`*MMDD* from binary to ASCII format.
3. **Edit the** `xtacct` **file. Remove the bad records and write the duplicate records to another file.**

4. **Type** `acctmerg –i xtacct tacct.`*MMDD* **and press Return.**
   The `–i` option converts the `xtacct` file from ASCII format to binary.

5. **Type** `acctmerg tacctprv tacct.`*MMDD* `tacct` **and press Return.**
   This merges the files `tacct.prv` and `tacct.`*MMDD* into the file `tacct`.

   The current `sum/tacct` can be re-created by merging all existing `tacct.`*MMDD* files using `acctmerg`, since the `monacct` procedure removes all the old `tacct.`*MMDD* files.

Restarting `runacct`

Called without arguments, `runacct` assumes that this is the first invocation of the day. The argument *MMDD* is necessary if `runacct` is being restarted and specifies the month and day for which `runacct` will rerun the accounting. The entry point for processing is based on the contents of `statefile`. To override `statefile`, include the desired state on the command line. The following are some *sample* procedures.

- **To start** `runacct`**, type**:
  `nohup runacct 2 > var/adm/acct/nite/fd2log`

- **To restart** `runacct`**, type**:
  `nohup runacct 0601 2 > /var/adm/acct/nite/fd2log`

- **To restart** `runacct` **in a specific state, type**:
  `nohup runacct 0601 WTMPFIX 2 > /var/adm/acct/nite/fd2log`

Billing Users

The `chargefee` program stores charges for special services provided to a user, such as file restoration, in the file `fee`. This file is incorporated by `runacct` every day.

- **To register special fees, type**:
  `chargefee` *login_name amount*

where *amount* is an integer amount to be charged. Most locations prefer to set up their own shell scripts for this function, with codes for services rendered. The operator then needs only to identify the service rendered. The system can tabulate the charge.

The monthly accounting program `monacct` produces monthly summary reports similar to those produced daily. The `monacct` program also summarizes the accounting information into the files in the `/var/adm/acct/fiscal` directory. This information can be used to generate monthly billing. To generate a monthly billing, many UNIX system administrators customize the accounting process with their own shell scripts.

Setting Up Non-Prime Time Discounts

UNIX system accounting provides facilities to give users a discount for non-prime time system use. For this to work, you must inform the accounting program of the dates of holidays and the hours that are considered nonprime time, such as nights. To do this, you must edit the `/etc/acct/holidays` file that contains the prime/nonprime table for the accounting program. The format is composed of three types of entries:

- *Comment Lines* – Comment lines are marked by an asterisk in the first column of the line. Comment lines may appear anywhere in the file.
- *Year Designation Line* – This line should be the first data line (noncomment line) in the file and must appear only once. The line consists of three fields of four digits each (leading white space is ignored). For example, to specify the year as 1994, prime time start at 9 a.m., and nonprime time start at 4:30 p.m., the following entry would be appropriate:

  ```
  1994 0900 1630
  ```

  A special condition allowed in the time field is that the time `2400` is automatically converted to `0000`.
- *Company Holidays Lines* – These entries follow the year designation line and have the following general format:

  ```
  Date      Description of Holiday
  ```

The date field has the format *month/day* and indicates the date of the holiday. The holiday field is actually commentary and is not currently used by other programs. A sample holiday list appears below.

Table 10-2 Holiday List

Month/Day	Holiday
1/1	New Year's Day
5/28	Memorial Day
7/4	Independence Day
9/3	Labor Day
11/22	Thanksgiving Day
11/23	Day after Thanksgiving
12/25	Christmas Day

Daily Accounting Reports

The `runacct` shell script generates four basic reports upon each invocation. These reports cover the areas of connect accounting, usage by login on a daily basis, command usage reported by daily and monthly totals, and a report of the last time users were logged in. The four basic reports generated are:

- *Daily Report* – Shows line utilization by `tty` number.
- *Daily Usage Report* – Indicates usage of system resources by users (listed in order of UID).
- *Daily Command Summary* – Indicates usage of system resources by commands, listed in descending order of use of memory (in other words, the command that used the most memory is listed first). This same information is reported for the month with the monthly command summary.
- *Last Login* – *S*hows the last time each user logged in (arranged in chronological order).

The following sections describe the reports and the meaning of the data presented in each one.

Daily Report

This report gives information about each terminal line used. A sample daily report appears below.

```
Jun 26 09:53  1994 DAILY REPORT FOR sfxbs Page 1

from      Thu Jun 25 17:45:22 1994
to        Fri Jun 26 09:51:25 1994
1         runacct
1         acctcon

TOTAL DURATION IS 966 MINUTES
LINE      MINUTES     PERCENT # SESS # ON  # OFF
term/23   25          3       7      7     3
term/22   157         16      6      6     3
TOTALS    183         --      13     13    7
-----------------------------------------------------
```

The `from` and `to` lines specify the time period reflected in the report—the period from the time the last accounting report was generated until the time the current accounting report was generated. It is followed by a log of system reboots, shutdowns, power fail recoveries, and any other record dumped into `/var/adm/wtmp` by the `acctwtmp` program; see `acct(1M)`.

The second part of the report is a breakdown of line utilization. The `TOTAL DURATION` tells how long the system was in multiuser state (accessible through the terminal lines). The columns are:

`LINE`	The terminal line or access port.
`MINUTES`	The total number of minutes that the line was in use during the accounting period.
`PERCENT`	The total number of `MINUTES` the line was in use, divided into the `TOTAL DURATION`.
`# SESS`	The number of times this port was accessed for a `login` session.
`# ON`	Identical to `SESS`. (This column does not have much meaning anymore. It used to list the number of times that a port was used to log in a user.)
`# OFF`	This column reflects the number of times a user logs out and any interrupts that occur on that line. Generally, interrupts occur on a port when `ttymon` is first invoked when the system is brought to

multiuser state. If the # OFF exceeds the # ON by a large factor, the multiplexer, modem, or cable is probably going bad, or there is a bad connection somewhere. The most common cause of this is an unconnected cable dangling from the multiplexer.

During real time, you should monitor /var/adm/wtmp because it is the file from which the connect accounting is geared. If the wtmp file grows rapidly, execute acctcon –l *file* < /var/adm/wtmp to see which tty line is the noisiest. If the interrupting is occurring frequently, general system performance will be affected.

Daily Usage Report

The daily usage report gives a breakdown of system resource utilization by user. A sample of this type of report appears below.

```
Jun 29  09:53  1994   DAILY USAGE REPORT FOR sfxbs Page 1

      LOGIN  CPU    (MINS) KCORE-MINS    CONNECT(MINS)  DISK    # OF   # OF # DISK   FEE
UID   NAME  PRIME   NPRIME PRIME  NPRIME PRIME  NPRIME   BLOCKS  PROCS  SESS SAMPLES
0     TOTAL    5      12      6     16     131     51        0    1114   13      0      0
0     root     2       8      1     11       0      0        0     519    0      0      0
3     sys      0       1      0      1       0      0        0      45    0      0      0
4     adm      0       2      0      1       0      0        0     213    0      0      0
5     uucp     0       0      0      0       0      0        0      53    0      0      0
999   rly      3       1      5      2     111     37        0     269    1      0      0
7987  jan      0       0      0      1      20     14        0      15    6      0      0
```

The data provided include the following:

UID	This is the user ID.
LOGIN NAME	This is the login name of the user. This information is useful because it identifies a user who has multiple login names.
CPU-MINS	This represents the amount of time the user's process used the central processing unit. This category is divided into PRIME and NPRIME (non-prime) utilization. The accounting system's version of this data is located in the file /etc/acct/holidays.

KCORE-MINS	This represents a cumulative measure of the memory a process uses while running. The amount shown reflects kilobyte segments of memory used, per minute. This measurement is also broken down into PRIME and NPRIME amounts.
CONNECT-MINS	This identifies the amount of "real time" used. This column identifies the amount of time that a user was logged in to the system. If the amount of time is high and the number shown in the column # OF PROCS is low, you can conclude the owner of the login logs in first thing in the morning and hardly touches the terminal the rest of the day. This column is also divided into PRIME and NPRIME use.
DISK BLOCKS	When the disk accounting programs have been run, the output is merged into the total accounting record (daytacct) and shows up in this column. This disk accounting is accomplished by the program acctdusg. For accounting purposes, a block is 512 bytes.
# OF PROCS	This column reflects the number of processes that were invoked by the user. This is a good column to watch for large numbers, indicating that a user may have a shell procedure that has run out of control.
# OF SESS	This column shows the number of times a user logged on to the system.
# DISK SAMPLES	This indicates how many times the disk accounting was run to obtain the average number of DISK BLOCKS listed earlier.
FEE	An often unused field in the total accounting record, the FEE field represents the total accumulation of widgets charged against the user by the chargefee shell procedure; see acctsh(1M). The chargefee procedure is used to levy charges against a user for special services performed, such as file restoration.

Daily Command Summary

The daily command summary report shows the system resource use by command. With this report, you can identify the most heavily used commands and, based on how those commands use system resources, gain insight on how best to tune the system. The daily and monthly reports are virtually the same; however, the daily summary reports only on the current accounting period while the monthly summary reports on the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of monacct.

These reports are sorted by `TOTAL KCOREMIN`, which is an arbitrary yardstick but often a good one for calculating drain on a system.

A sample daily command summary appears below.

```
 Jun 29 09:52 1994 DAILY COMMAND SUMMARY Page 1

    TOTAL COMMAND SUMMARY
         PRIME           PRIME    PRIME
COMMAND  NUMBER  TOTAL   TOTAL    TOTAL     MEAN      MEAN      HOG      CHARS     BLOCKS
NAME     CMDS   KCOREMIN CPU-MIN  REAL-MIN SIZE-K    CPU-MIN   FACTOR   TRNSFD     READ

TOTALS  1114      2.44    16.69   136.33     0.15     0.01      0.12  4541666      1926

sh       227      1.01     2.45    54.99     0.41     0.01      0.04   111025       173
vi        12      0.35     0.62    44.23     0.55     0.05      0.01   151448        60
sed      143      0.09     0.82     1.48     0.10     0.01      0.55    14505        35
sadc      13      0.08     0.19     1.45     0.44     0.01      0.13   829088        19
more       3      0.04     0.07     2.17     0.59     0.02      0.03    30560         1
cut       14      0.03     0.09     0.28     0.37     0.01      0.33      154        13
uudemon.  76      0.03     0.66     2.30     0.05     0.01      0.29    43661        13
uuxqt     29      0.03     0.30     0.72     0.08     0.01      0.42    80765        35
mail       4      0.02     0.06     0.09     0.37     0.01      0.60     4540         9
ckstr     21      0.02     0.11     0.13     0.17     0.01      0.85        0         4
awk       13      0.02     0.12     0.21     0.15     0.01      0.54      444         2
ps         2      0.02     0.10     0.13     0.17     0.05      0.77     8060        21
find       9      0.02     3.35     5.73     0.00     0.37      0.58   355269       760
sar        1      0.01     0.19     0.24     0.08     0.19      0.80   564224         4
acctdisk   2      0.01     0.01     0.06     1.02     0.01      0.22        0         9
mv        24      0.01     0.14     0.17     0.10     0.01      0.81     3024        36
  .
  .
  .
```

The data provided, by column, include the following:

`COMMAND NAME`	This is the name of the command. Unfortunately, all shell procedures are lumped together under the name `sh` because only object modules are reported by the process accounting system. It's a good idea to monitor the frequency of programs called `a.out` or `core` or any other name that does not seem quite right. `acctcom` can be used to determine who executed a suspiciously named command and if superuser privileges were used.
`PRIME NUMBER CMNDS`	This is the total number of invocations of this particular command during prime time.
`TOTAL KCOREMIN`	This is the total cumulative measurement of the kilobyte segments of memory used by a process per minute of run time.
`PRIME TOTAL CPU-MIN:`	This is the total processing time this program has accumulated during prime time.
`PRIME TOTAL REAL-MIN`	This is the total real-time (wall-clock) minutes this program has accumulated.
`MEAN SIZE-K`	This is the mean of the `TOTAL KCOREMIN` over the number of invocations reflected by `NUMBER CMDS`.
`MEAN CPU-MIN`	This is the mean derived between the `NUMBER CMDS` and `TOTAL CPU-MIN.`
`HOG FACTOR`	This is the total CPU time divided by the elapsed time. This shows the ratio of system availability to system use, providing a relative measure of the total available CPU time consumed by the process during its execution.
`CHARS TRNSFD`	This column, which may go negative because of overflow, is a total count of the number of characters pushed around by the read and write system calls.
`BLOCKS READ`	This is a total count of the physical block reads and writes that a process performed.

Total Command Summary

The monthly command summary is similar to the daily command summary. The only difference is that the monthly command summary shows totals accumulated since the last invocation of `monacct`. A sample report appears below.

```
    TOTAL COMMAND SUMMARY

COMMAND  NUMBER  TOTAL     TOTAL     TOTAL     MEAN   MEAN    HOG     CHARS        BLOCKS
NAME      CMDS KCOREMIN   CPU-MIN   REAL-MIN  SIZE-K CPUMIN FACTOR   TRNSFD        READ

TOTALS 301314  300607.70 4301.59  703979.81   69.88   0.01   0.01 6967631360   10596385

troff     480   58171.37   616.15    1551.26   94.41   1.28   0.40  650669248     194926
rnews    5143   29845.12   312.20    1196.93   95.59   0.06   0.26 1722128384    2375741
uucico   2710   16625.01   212.95   52619.21   78.07   0.08   0.00  228750872     475343
nroff    1613   15463.20   206.54     986.06   74.87   0.13   0.21  377563304     277957
vi       3040   14641.63   157.77   14700.13   92.80   0.05   0.01  116621132     206025
expire     14   13424.81   104.90     265.67  127.98   7.49   0.39   76292096     145456
comp     3483   12140.64    60.22     423.54  201.62   0.02   0.14    9584838     372601
ad_d       71   10179.20    50.02    1158.31  203.52   0.70   0.04   11385054      19489
as       2312    9221.59    44.40     285.52  207.68   0.02   0.16   35988945     221113
gone      474    8723.46   219.93   12099.01   39.67   0.46   0.02   10657346      19397
i10       299    8372.60    44.45     454.21  188.34   0.15   0.10   60169932      78664
find      760    8310.97   196.91     728.39   42.21   0.26   0.27   58966910     710074
ld       2288    8232.84    61.19     425.57  134.55   0.03   0.14  228701168     279530
fgrep     832    7585.34    62.62     199.11  121.14   0.08   0.31   22119268      37196
sh      56314    7538.40   337.60  291655.70   22.33   0.01   0.00   93262128     612892
du        624    5049.58   126.32     217.59   39.97   0.20   0.58   16096269     215297
ls      12690    4765.60    75.71     541.53   62.95   0.01   0.14   65759473     207920
vnews      52    4235.71    28.11     959.74  150.70   0.54   0.03   28291679      28285
  .
  .
  .
```

See the listing under the section “Daily Command Summary” on page 182 for a description of the data.

Last Login Report

This report gives the date when a particular login was last used. You can use this information to find unused logins and login directories that may be archived and deleted. A sample report appears below.

```
Feb 13 04:40 1994 LAST LOGIN Page 1

00-00-00    **rje** 88-01-01    jlr    88-02-09   cec42   88-02-13    cec20
00-00-00    **rje** 88-01-13    crom   88-02-10   jgd     88-02-13    cec22
00-00-00    3bnet   88-01-14    usg    88-02-10   wbr     88-02-13    cec23
00-00-00    adm     88-01-17    cec11  88-02-11   cec30   88-02-13    cec24
00-00-00    daemon  88-01-17    cec38  88-02-11   cec41   88-02-13    cec25
00-00-00    notes   88-01-17    cec40  88-02-11   cec43   88-02-13    cec26
00-00-00    oas     88-01-18    cec60  88-02-11   cec53   88-02-13    cec27
00-00-00    pds     88-01-19    cec35  88-02-11   cec54   88-02-13    cec3
00-00-00    polaris 88-01-19    cec37  88-02-11   cec55   88-02-13    cec31
00-00-00    rje     88-01-22    dmk    88-02-11   cec56   88-02-13    cec32
00-00-00    shqer   88-01-26    ask    88-02-11   cec57   88-02-13    cec4
00-00-00    sys     88-01-26    cec39  88-02-11   cec58   88-02-13    cec6
00-00-00    trouble 88-01-27    sync   88-02-11   jwg     88-02-13    cec7
00-00-00    usors   88-02-02    pkl    88-02-11   skt     88-02-13    cec8
00-00-00    uucp    88-02-03    ibm    88-02-11   tfm     88-02-13    commlp
00-00-00    wna     88-02-03    slk    88-02-12   cec21   88-02-13    djs
87-07-06    lp      88-02-04    cec59  88-02-12   cec28   88-02-13    epic
87-07-30    dgn     88-02-05    cec33  88-02-12   cec29   88-02-13    jab
87-08-19    blg     88-02-05    cec34  88-02-12   csp     88-02-13    jcs
87-12-08    emna    88-02-05    cec36  88-02-12   drc     88-02-13    mak
88-01-14    s       88-02-05    cec51  88-02-12   emw     88-02-13    dn
88-01-09    rib     88-02-05    dfh    88-02-12   je      88-02-13    mlp
88-01-25    dmf     88-02-05    fsh    88-02-12   kab     88-02-13    nbh
88-01-25    emda    88-02-05    pkw    88-02-12   rap     88-02-13    rah
  .
  .
  .
```

Looking at the `pacct` *File With* `acctcom`

At any time, you can examine the contents of the `/var/adm/pacct`*n* files, or any file with records in the `acct.h` format, by using the `acctcom` program. If you don't specify any files and don't provide any standard input when you run this command, `acctcom` reads the `pacct` file. Each record read by `acctcom` represents information about a dead process (active processes may be examined by running the `ps` command). The default output of `acctcom` provides the following information:

- Command name (# sign if it was executed with superuser privileges)
- User
- `tty` name (listed as `?` if unknown)
- Starting time
- Ending time
- Real time (in seconds)
- CPU time (in seconds)
- Mean size (in Kbytes)

The following information can be obtained by using options to `acctcom`:

- State of the `fork/exec` flag (1 for `fork` without `exec`)
- System exit status
- Hog factor
- Total `kcore` minutes
- CPU factor
- Characters transferred
- Blocks read

The options are:

`-a`	Show some average statistics about the processes selected. (The statistics are printed after the output is recorded.)
`-b`	Read the files backward, showing latest commands first. (This has no effect if reading standard input.)
`-f`	Print the `fork/exec` flag and system exit status columns. (The output is an octal number.)
`-h`	Instead of mean memory size, show the hog factor, which is the fraction of total available CPU time consumed by the process during its execution. Hog factor = *total_CPU_time/elapsed_time.*
`-i`	Print columns containing the I/O counts in the output.
`-k`	Show total `kcoreminutes` instead of memory size.
`-m`	Show mean core size (this is the default).
`-q`	Don't print output records, just print average statistics.

`-r`	Show CPU factor: *user_time* / (*system_time* + *user_time*).
`-t`	Show separate system and user CPU times.
`-v`	Exclude column headings from the output.
`-C` *sec*	Show only processes with total CPU time (system plus user) exceeding *sec* seconds.
`-e` *time*	Show processes existing at or before *time*, given in the format *hr[:min[:sec]]*.
`-E` *time*	Show processes starting at or before *time*, given in the format *hr[:min[:sec]]*. Using the same *time* for both –S and –E shows processes that existed at the time.
`-g` *group*	Show only processes belonging to *group*.
`-H` *factor*	Show only processes that exceed *factor*, where *factor* is the "hog factor" (see the –h option.
`-I` *chars*	Show only processes transferring more characters than the cutoff number specified by *chars*.
`-l` *line*	Show only processes belonging to the terminal `/dev/line`.
`-n` *pattern*	Show only commands matching *pattern* (a regular expression as in ed except that "+" means one or more occurrences).
`-o` *ofile*	Instead of printing the records, copy them in acct.h format to *ofile*.
`-O` *sec*	Show only processes with CPU system time exceeding *sec* seconds.
`-s` *time*	Show processes existing at or after *time*, given in the format *hr*[:*min*[:*sec*]].
`-S` *time*	Show processes starting at or after *time*, given in the format *hr*[:*min*[:*sec*]].
`-u` *user*	Show only processes belonging to *user*.

Accounting Files

The `/var/adm` directory structure contains the active data collection files and is owned by the `adm` login (currently user ID of 4).

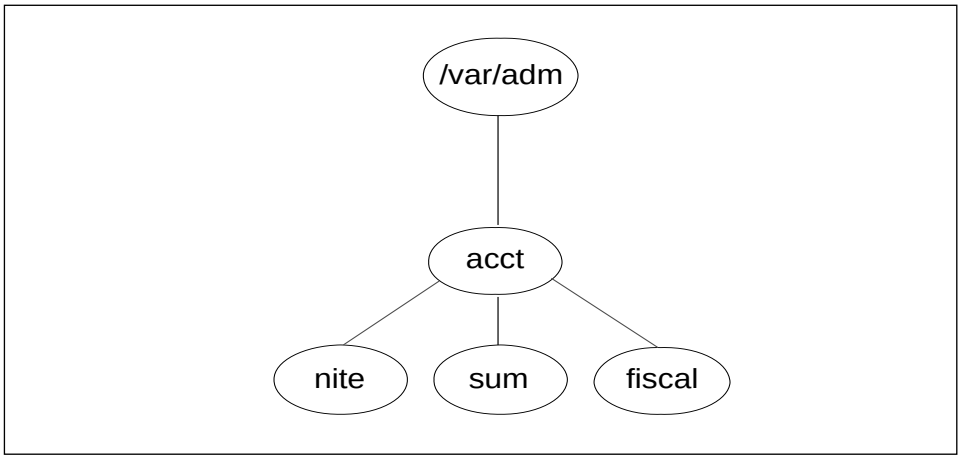


Figure 10-1 Directory Structure of `/var/adm`

A brief description of the files found in the `/var/adm` directory follows:

`dtmp`	Output from the `acctdusg` program
`fee`	Output from the `chargefee` program, ASCII `tacct` records
`pacct`	Active process accounting file
`pacct`*n*	Process accounting files switched using `turnacct`
`Spacct`*n.MMDD*	Process accounting files for *MMDD* during execution of `runacct`

The `/var/adm/acct` directory contains the `nite`, `sum`, and `fiscal` directories, which contain the actual data collection files. For example, the `nite` directory contains files that are reused daily by the `runacct` procedure. A brief summary of the files in the `/var/adm/acct/nite` directory follows:

`active`	Used by `runacct` to record progress and print warning and error messages
`active.`*MMDD*	Same as active after `runacct` detects an error
`cms`	ASCII total command summary used by `prdaily`
`ctacct.`*MMDD*	Connect accounting records in `tacct.h` format
`ctmp`	Output of `acctcon1` program, connect session records in `ctmp.h` format. (`acctcon1` and `acctcon2` are provided for compatibility purposes)

`daycms`	ASCII daily command summary used by `prdaily`
`daytacct`	Total accounting records for one day in `tacct.h` format
`disktacct`	Disk accounting records in `tacct.h` format, created by the `dodisk` procedure
`fd2log`	Diagnostic output during execution of `runacct` (see "Setting Up Accounting" at the beginning of this chapter)
`lastdate`	Last day `runacct` executed (in `date +%m%d` format)
`lock`	Used to control serial use of `runacct`
`lineuse`	`tty` line usage report used by `prdaily`
`log`	Diagnostic output from `acctcon`
`log.`*MMDD*	Same as log after `runacct` detects an error
`owtmp`	Previous day's `wtmp` file
`reboots`	Beginning and ending dates from `wtmp` and a listing of reboots
`statefile`	Used to record current state during execution of `runacct`
`tmpwtmp`	`wtmp` file corrected by `wtmpfix`
`wtmperror`	Place for `wtmpfix` error messages
`wtmperror.`*MMDD*	Same as `wtmperror` after `runacct` detects an error
`wtmp.`*MMDD*	`runacct`'s copy of the `wtmp` file

The `sum` directory contains the cumulative summary files updated by `runacct` and used by `monacct`. A brief summary of the files in the `/var/adm/acct/sum` directory follows:

`cms`	Total command summary file for current fiscal period in internal summary format
`cmsprev`	Command summary file without latest update
`daycms`	Command summary file for the day's usage in internal summary format
`loginlog`	Record of last date each user logged on; created by `lastlogin` and used in the `prdaily` program
`rprt.`*MMDD*	Saved output of `prdaily` program

`tacct`	Cumulative total accounting file for current fiscal period
`tacctprev`	Same as `tacct` without latest update
`tacct.`*MMDD*	Total accounting file for *MMDD*

The fiscal directory contains periodic summary files created by monacct. A brief description of the files in the `/var/adm/acct/fiscal` directory follows:

`cms`*n*	Total command summary file for fiscal period *n* in internal summary format
`fiscrpt`*n*	Report similar to `rprtn` for fiscal period *n*
`tacct`*n*	Total accounting file for fiscal period *n*

Quick Reference to Accounting

- **To start accounting:**

  `/usr/lib/acct/startup`

- **To turn off accounting:**

  `/usr/lib/acct/shutacct`

- **To switch the `pacct` file to the `pacct`*n* file:**

  `/usr/lib/acct/ckpacct`

- **To examine the contents of `pacct`:**

  `/bin/acctcom`

- **To charge a fee:**

  `/usr/lib/acct/chargefee` *login_name amount*

- **To process accounting files into a daily summary:**

  `/usr/lib/acct/runacct 2 > /var/adm/acct/nite/fd2log`

- **To do disk accounting:**

  `/usr/lib/acct/dodisk`

- **To create a monthly accounting report:**

  `/usr/lib/acct/monacct` *fiscal_number*

♦ **To print** `tacct.h` **files in ASCII format**:

`/usr/lib/acct/prtacct` *filename*

Tuning Kernel Parameters A

The SunOS 5.x kernel modules are automatically loaded when they are needed. This makes rebuilding the kernel unnecessary. There are parameters for the kernel and kernel modules that you can tune. However, you shouldn't need to change these parameters, except under special circumstances. Many parameters automatically scale as a function of the value assigned to the `maxusers` parameter.

This appendix lists some of the kernel parameters that you can change or tune. You can change any global variable that is one of the following data types—`int`, `long`, `short`, or `char`—by entering a line in the `/etc/system` file.

▼ How to List the Kernel Parameters

To see the current values assigned to the kernel parameters.

♦ **Type** `sysdef -i` **and press Return.**

▼ How to Change the Value of a Parameter

To set the value of a parameter:

1. **Enter a line in the** `/etc/system` **file in the form:**
   `set` *parameter=value*

   The following example sets the value of the `max_nprocs` to `500` parameter.

```
set max_nprocs=500
```

2. **Reboot the system.**
   The kernel parses the `/etc/system` file during autoconfiguration and overrides the default value for the parameters specified in this file.

To set the value of a module variable:

1. **Enter a line in the** `/etc/system` **file in the form:**
   `set` *module_name*`:`*variable=value*

   The following example sets the value of the `msginfo_msgmap` parameter in the `msgsys` module to `150`.

```
set msgsys:msginfo_msgmap=150
```

2. **Reboot the system.**
   The kernel parses the `/etc/system` file during autoconfiguration and overrides the default value for the parameters specified in this file.

Buffer Cache Parameters

The `bufhwm` parameter specifies the maximum size for buffer cache memory usage expressed in units of 1K bytes. The default is 2% of physical memory.

UFS File System Parameters

The following table describes the tunable UFS parameters.

Table A-1 UFS File System Parameters

Parameter	Description
`ufs_ninode`	Maximum size of the inode table (default = `max_nprocs` + 16 + `maxusers`+ 64)
`ncsize`	Number of `dnlc` entries (default = `max_nprocs`+16+`maxusers` + 64); `dnlc` is the directory-name lookup cache.

STREAMS *Parameters*

The following table describes the tunable STREAMS parameters.

Table A-2 STREAMS Parameters

Parameter	Default	Description
`nstrpush`	`9`	This is the maximum number of STREAMS pushes allowed.
`strmsgsz`	`0`	This is the maximum size for the STREAMS message that a user can create. A value of `0` indicates no upper bound. This parameter may disappear entirely in a future release.
`strctlsz`	`1024`	This is the maximum size of the `ctl` part of a message.
`strthresh`	`0`	This is the maximum amount of dynamic memory that the STREAMS subsystem can consume, in bytes. Once this threshold is passed, any pushes, opens, and writes on a STREAMS devices will fail for non-root processes. A value of `0` means no limit.
`sadcnt`	16	Number of `sad` devices

A

Interprocess Communication (IPC) Parameters

The following table describes the tunable interprocess communication parameters.

Table A-3 Message Queue Parameters

Parameter	Default	Description
msginfo_msgmap	100	Number of entries in the message map
msginfo_msgmax	2048	Maximum message size
msginfo_msgmnb	4096	Maximum bytes on queue
msginfo_msgmni	50	Number of message queue identifiers
msginfo_msgssz	8	Segment size of a message (should be a multiple of the word size)
msginfo_msgtql	40	Number of system message headers
msginfo_msgseg	1024	Number of message segments (must be < 32768)
seminfo_semmap	10	Number of entries in semaphore map
seminfo_semmni	10	Number of semaphore identifiers
seminfo_semmns	60	Number of semaphores in the system
seminfo_semmnu	30	Number of undo structures in the system
seminfo_semmsl	25	Maximum number of semaphores, per id
seminfo_semopm	10	Maximum number of operations, per semaphore call
seminfo_semume	10	Maximum number of undo entries, per process
seminfo_semvmx	32767	Semaphore maximum value
seminfo_semaem	16384	Maximum value for adjustment on exit
shminfo_shmmax	1048576	Maximum shared memory segment size
shminfo_shmmin	1	Minimum shared memory segment size
shminfo_shmmni	100	Number of shared memory identifiers
shminfo_shmseg	6	Segments, per process

▼ How to Tune the Message Queue Parameters

- **Tune the** `msgsys` **parameters using the following syntax:**

  `set msgsys:msginfo_`*variable*`=`*value*

- **Tune the** `semsys` **parameters using the following syntax:**

  `set semsys:seminfo_`*variable*`=`*value*

- **Tune the** `shmsys` **parameters using the following syntax:**

  `set shmsys:shminfo_`*variable*`=`*value*

A

TPI Loopback Pseudo-Driver Parameters

The following table describes the tunable TPI loopback pseudo-driver parameters.

Table A-4 TPI Loopback Pseudo-Driver Parameters

Parameter	Default	Description
`tune_t_gpgslo`	`25`	If `freemem` < `t_getpgslow`, the system starts to steal pages from processes. If `freemem` drops below `t_gpgslo`, the system wakes up the swapper process. The swapper will try to free some memory by swapping processes to disk.
`tune_t_fsflushr`	`30`	Rate at which `fsflush` is run, in seconds
`tune_t_minarmem`	`25`	The minimum available resident (not swappable) memory needed to avoid deadlock, in pages
`tune_t_minasmem`	`25`	The minimum available swappable memory needed to avoid deadlock, in pages
`tune_t_flckrec`	`512`	The maximum number of active `frlocks`
`putbufsz`	`2000`	Size of the `putchar` buffer
`npty`	`48`	Total number of 4.*x* `pseudo-ttys` configured
`pt_cnt`	`48`	Total number of 5.*x* `pseudo-ttys` configured

Note – Both the `freemem` and `t_gpgslo` parameters are defined in pages. Utilities like `vmstat` translates `freemem` into bytes from pages.

▼ How to Tune the TPI Loopback Pseudo-Driver Parameters

♦ **Tune the `tune` parameters using the following syntax:**
`set tune:`*variable=value.*

Miscellaneous Parameters

The following table describes a tunable miscellaneous parameter.

Table A-5 `Miscellaneous` Parameter

Parameter	Default	Description
`lwp_default_stksize`	`8192`	Size of the kernel stack for `lwps`. Do not adjust this value unless there is a kernel overflow. The value is expressed in bytes and must be a multiple of `PAGESIZE` bytes.

A

The Scheduler B

This appendix contains reference information for the SunOS 5.x scheduler.

The *scheduler* (or dispatcher) is the portion of the kernel that controls the allocation of the CPU to processes. It determines when processes run and for how long, depending on their assigned priorities. Priorities are based on scheduling class and process behavior. Four scheduling classes are supported by default: timesharing, system, real-time and interactive.

The scheduler has an overriding effect on the performance of a system.

This appendix is organized as follows:

Note – The fundamental scheduling entity is the kernel thread. For single-threaded processes, scheduling the kernel thread is synonymous with process scheduling.

B

About the Scheduler

The SunOS 5.x scheduler controls the order in which processes run and the amount of CPU time each process may use before another process can run.

The scheduler allocates CPU time to processes according to the scheduling policies defined for each scheduling class. Associated with each scheduling class is a set of priority levels or queues. Ready-to-run processes are moved among these queues. Within a class, you can view these queues as a contiguous set of priority levels. These priority levels are mapped into a set of global scheduling priorities.

The global priority of a process determines when it runs—the scheduler runs the process with the highest global priority that is ready to run. Processes with numerically higher priorities run first, and processes with the same priority run using a round robin scheduling policy.

Once the scheduler assigns a process to a CPU, the process runs until one of the following events occur:

- The process uses up its time slice.
- The process blocks waiting for an event (for example, I/O) or a suspended lock.
- The process is pre-empted by a higher-priority process.

By default, all real-time processes have higher priorities than any system process, and all system processes have higher priorities than any timesharing process.

A process inherits its scheduler parameters from its parent process, including its scheduler class and its priority within that class. A process changes class only from a user request (with the `priocntl` command or system call). The system manages the priority of a process based on user requests and the policy associated with the scheduling class of the process.

Scheduler Class Policies

The following sections describe the scheduling policies of the three default classes: timesharing, system, and real-time.

Timesharing Class Policies

In the default configuration, the initialization process (`init`) belongs to the timesharing class. Because processes inherit their scheduler parameters, all user login shells—and consequently the processes run from those shells—begin as timesharing processes.

The goal of the timesharing policy is to provide good response time for interactive processes and good throughput for processes that use a lot of CPU time. The scheduler tries to divide the CPU's time fairly between processes, subject to the priorities associated with the processes. Those with higher priorities get more attention than those with lower priorities. However, to prevent any one job (process) from hogging the CPU, the scheduler can move jobs from high priorities to low priorities and vice versa.

The scheduler switches CPU allocation frequently enough to provide good response time, but not so frequently that it spends too much time doing the switching. Time slices are typically on the order of a few hundredths of a second.

The timesharing policy changes priorities dynamically and assigns time slices of different lengths. Once a process has started, its timesharing priority varies according to how much CPU time it's getting, how much time it's spending in queues, and other factors. The scheduler raises the priority of a process that "sleeps." (A process sleeps, for example, when it starts an I/O operation such as a terminal read or a disk read.) Entering sleep states frequently is characteristic of interactive tasks such as editing and running simple shell commands. On the other hand, the timesharing policy lowers the priority of a process that uses the CPU for long periods without sleeping.

The default timesharing policy gives larger time slices to processes with lower priorities. A process with a low priority is likely to be stuck in the CPU. Other processes get the CPU first, but when a lower-priority process finally gets the CPU, it gets a bigger chunk of time. If a higher-priority process becomes ready to run during a time slice, however, it pre-empts the running process.

The scheduler manages timesharing processes using parameters in the timesharing parameter table `ts_dptbl`. This table contains information specific to the timesharing class. It is automatically loaded into core memory from the `TS_DPTBL` loadable module located in the `/kernel/sched` directory.

System Class Policies

The system class uses a fixed-priority policy to run kernel processes such as servers, and housekeeping processes such as the page daemon. Their priorities are not dynamically adjusted like timesharing processes. The system class is reserved for use by the kernel, and users may neither add nor remove a process from the system class. Priorities for system-class processes are set up in the kernel code for the kernel processes, and, once established, these priorities do not change. (User processes running in kernel mode are not in the system class.)

Real-Time Class Policies

The SunOS 5.x operating system uses a real-time scheduling policy as well as a timesharing policy. Real-time scheduling allows users to set fixed priorities on a per-process basis, so that critical processes can run in predetermined order. The real-time scheduler never moves jobs between priorities. Real-time priorities change only when a user requests a change (using the `priocntl` command). Contrast this fixed-priority policy with the timesharing policy, in which the system changes priorities to provide good interactive response time.

The user process with the highest real-time priority always gets the CPU as soon as it can be run, even if other processes are ready to run. An application can be written so that its real-time processes have a guaranteed response time from the operating system.

Note – As long as there is a real-time process ready to run, no process and no timesharing process runs. Other real-time processes can run only if they have a higher priority. Real-time processes managed carelessly can have a dramatic negative effect on the performance of timesharing processes.

The real-time policy gives higher-priority processes smaller time slices, by default. The higher priorities are allocated to real-time processes that are driven by external events. The operating system must be able to respond instantly to I/O. The lower-priority real-time processes are those that need more computation time. If a process with the highest priority uses up its time slice, it runs again because there is no process with a higher priority to pre-empt it.

The scheduler manages real-time processes by using parameters in the real-time parameter table `rt_dptbl`. This table contains information specific to the real-time class. It is automatically loaded into core from the `RT_DPTBL` loadable module located in the `/kernel/sched` directory.

Scheduler Configuration

This section describes the parameters and tables that control the scheduler configuration. A basic assumption is that your work load is reasonable for your system resources, such as CPU, memory, and I/O. If your resources are inadequate to meet the demands, reconfiguring the scheduler won't help.

You can display or change (fine tune) the scheduler parameters in a running system for both the timesharing and real-time classes by using the `dispadmin` command. Changes made by the `dispadmin` command do not survive a reboot. To make permanent changes in scheduler configuration, you must change the scheduler parameter tables in the appropriate loadable module: `TS_DPTBL` or `RT_DPTBL` provided in the `/kernel/sched` directory. See `ts_dptbl(4)` and the `rt_dptbl(4)` for instructions on replacing these modules.

The primary user command for controlling process (or process) scheduling is `priocntl(1)`. With this command, a user can start a process at a specified priority or manipulate the priorities of running processes. You can find out what classes are configured on your system with the `priocntl -l` command. The primary function call for controlling process scheduling is `priocntl(2)`.

See Chapter 7, "Managing Processes," for examples of using the `priocntl` command. See *System Services Guide* for a detailed descriptions of real-time programming, and the `dispadmin(1M)` and `priocntl(1)` commands.

B

Default Global Priorities

The following table shows the scheduling order and ranges of global priorities for each scheduler class.

Table B-1 Scheduling Order and Global Priorities

Scheduling Order	Global Priority	Scheduler Class
First	159	
	.	
	.	Real-Time
	.	
	100	
	99	
	.	
	.	System
	.	
	60	
	59	
	.	
	.	Timesharing
	.	
Last	0	

How Global Priorities Are Constructed

When your operating system is built, it constructs the global priorities from the tunable parameters and scheduler parameter tables described in the following sections. There isn't any command that will show you this complete global priority table. However, the `dispadmin` command displays the priorities (from 0 to *n*) specific to the real-time and timesharing classes. You can display the global priority of an active process with the `ps -cl` command.

Initial Global Priorities of Processes

A timesharing process inherits its scheduling class and priority from its parent process. The `init` process is the first process to entire the timesharing class.

System processes initially run with a priority that depends on the process's importance (which is programmed into the kernel). The most important system processes start with a priority at or near the top of the system class range.

Tunable Parameters

This section describes the tunable parameters that control scheduler configuration. To change any of these kernel parameters, enter a line in the `/etc/system` file with the format:

`set` *parameter=value*

See the reference page for `system(4)` for more information.

The parameters described in this section control aspects of process scheduling, timesharing policy, and real-time policy.

The initial priority of a real-time process is determined when the process is put into the real-time scheduling class.

The `-p` option of the `priocntl` command is used to specify the relative priority within the real-time class.

This is added to the base priority of the real-time class, which by default is 100. For example:

`priocntl -e -c RT -p 20` *command*

would put the command into execution at a real-time priority of 120.

Process Scheduling Parameters

The following kernel parameters control aspects of process scheduling:

- `maxclsyspri`

  `maxclsyspri` is the maximum global priority of processes in the system class. When the kernel starts system processes, it assigns their priorities using the value of `maxclsyspri` as a reference point. `maxclsyspri` must have a value of 39 or greater, because the kernel assumes that the total range of system class priorities is at least 40.

  If you change this parameter, you must rebuild the scheduling class tables with values that correspond to the maximum priorities that you assign.

- `sys_name`

  `sys_name` is the character string name of the system scheduler class. The default value of `sys_name` is `SYS`.

Timesharing Policy

The following parameter is specified in the `TS` loadable module, which controls the timesharing policy:

- `ts_maxupri`

  `ts_maxupri` specifies the range within which users may adjust the priority of a timesharing process, using the `priocntl(l)` command or the `priocntl(2)` system call. The valid range for the user-supplied priority in the timesharing class is from +`ts_maxupri` to –`ts_maxupri`. The default value of `ts_maxupri` is 20 (which sets the range between +20 and –20, emulating the behavior of the older, less general scheduler interfaces, `nice` and `setpriority`.)

  The value of `ts_maxupri` is independent of the configured number of global timesharing priorities. In the default configuration, there are 0-59 timesharing priorities, but users may adjust their priorities only within a range of –20 to +20, relative to the system-calculated priority of the process. See “Example of Using priocntl to Designate Priority” on page 121 for more information.

To change the value of this parameter, enter a line in `/etc/system` with the format:

```
set TS:ts_maxupri=value
```

Real-Time Policy

The following parameter is specified in the RT loadable module, which controls the real-time policy:

- `rt_maxpri`

  `rt_maxpri` specifies the maximum priority to assign to real-time processes. The default value of `rt_maxpri` is 159.

  If you change this parameter, you must rebuild the scheduling class tables with values that correspond to the maximum priorities that you assign.

  To change the value of this parameter, enter a line in `/etc/system` with the format:

  ```
  set RT:rt_maxupri=value
  ```

Scheduler Parameter Tables

The scheduler uses the following tables:

- `rt_dptbl` – The real-time scheduler (or dispatcher) parameter table is used to manage real-time processes.
- `ts_dptbl` – The timesharing scheduler (or dispatcher) parameter table is used to manage timesharing processes.
- `ts_kmdpris` – The kernel-mode table is used to manage sleeping timesharing processes that own critical resources.

These tables define scheduling policy by setting the scheduling parameters to use for real-time and timesharing processes. The parameters specify how much CPU time processes get at different priority levels.

Default time slices for the priority levels are specified in the `ts_dptbl` and `rt_dptbl` configuration tables, which are defined in the `TS_DPTBL` and `RT_DPTBL` loadable modules. These modules are automatically loaded from the `/kernel/sched` directory into the kernel as needed.

The time slices are specified in units (quanta) with a resolution defined by a "resolution" line. The default resolution is 1000, which means the time quantum values are interpreted as milliseconds. This is derived from the reciprocal of the specified resolution in seconds. The quanta are rounded up to the next integral multiple of the system clock's resolution in clock ticks. (The system clock ticks `HZ` times per second, where `HZ` is a hardware-dependent constant defined in the `param.h` header file.) For example, if the clock tick is 10 milliseconds, 42 quanta is rounded up to 50 milliseconds.

Timesharing Parameter Table

A default version of the `ts_dptb`, is delivered with the system in `/kernel/sched/TS_DPTBL`. The default configuration has 60 timesharing priorities.

The `dispadmin -c TS -g` command on the following page displays a sample `ts_dptbl` table.

Figure B-1 Sample `ts_dptbl` Table

```
# ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait PRIORITY
                                                    LEVEL
     200        0        59        0        50        #  0
     200        0        59        0        50        #  1
     200        0        59        0        50        #  2
     200        0        59        0        50        #  3
     200        0        59        0        50        #  4
     200        0        59        0        50        #  5
     200        0        59        0        50        #  6
     200        0        59        0        50        #  7
     200        0        59        0        50        #  8
     200        0        59        0        50        #  9
     160        0        59        0        51        #  10
     160        1        59        0        51        #  11
     160        2        59        0        51        #  12
     160        3        59        0        51        #  13
     160        4        59        0        51        #  14
     160        5        59        0        51        #  15
     160        6        59        0        51        #  16
     160        7        59        0        51        #  17
     160        8        59        0        51        #  18
     160        9        59        0        51        #  19
     120        10       59        0        52        #  20
     120        11       59        0        52        #  21
     120        12       59        0        52        #  22
     120        13       59        0        52        #  23
     120        14       59        0        52        #  24
     120        15       59        0        52        #  25
     120        16       59        0        52        #  26
     120        17       59        0        52        #  27
     120        18       59        0        52        #  28
     120        19       59        0        52        #  29
     80         20       59        0        53        #  30
```

```
# ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait PRIORITY
                                                    LEVEL
     80         21       59        0        53        #  31
     80         22       59        0        53        #  32
     80         23       59        0        53        #  33
     80         24       59        0        53        #  34
     80         25       59        0        54        #  35
     80         26       59        0        54        #  36
     80         27       59        0        54        #  37
     80         28       59        0        54        #  38
     80         29       59        0        54        #  39
     40         30       59        0        55        #  40
     40         31       59        0        55        #  41
     40         32       59        0        55        #  42
     40         33       59        0        55        #  43
     40         34       59        0        55        #  44
     40         35       59        0        56        #  45
     40         36       59        0        57        #  46
     40         37       59        0        58        #  47
     40         38       59        0        58        #  48
     40         39       59        0        58        #  49
     40         40       59        0        59        #  50
     40         41       59        0        59        #  51
     40         42       59        0        59        #  52
     40         43       59        0        59        #  53
     40         44       59        0        59        #  54
     40         45       59        0        59        #  55
     40         46       59        0        59        #  56
     40         47       59        0        59        #  57
     40         48       59        0        59        #  58
     40         49       59        0        59        #  59
```

Table B-2 describes the fields in the `ts_dptbl` table.

Table B-2 Fields in the `ts_dptbl` Table

Field Name	Description
`ts_quantum` (runtime)	Contains the time slice (in milliseconds by default) that a process at a given priority is allowed to run before the scheduler re-evaluates its priority. If the process uses up its entire time slice, it is put on the expired-level (`ts_tqexp`) queue. Time slices run from 40 milliseconds for the highest priority (`59`) to 200 milliseconds (`0`) for the lowest priority.
`ts_tqexp` (expired level)	Determines the new process priority for a process whose time slice has expired. If a process uses its whole time slice without sleeping, the scheduler changes its priority to the level indicated in the `ts_tqexp` column. The expired level is lower than the prior level. For example, a process with a priority of `30` that used up its time slice (80 milliseconds) will get a new priority of `20`. (See Figure B-1 on page 211.)
`ts_slpret` (sleeplevel)	Determines the priority assigned to a process when it returns from sleep. A process may sleep during certain system calls or when waiting for I/O (for example, servicing a page fault or waiting for a lock). When a process returns from sleep, it is always a given a priority of `59`. (See Figure B-1 on page 211.)
`ts_maxwait` (waittime)	Specifies the number of seconds a process will be left on a dispatch queue without expiring its time slice. If it does not use its time slice (in `ts_maxwait` seconds), its new priority will be set to `ts_lwait`. This is used to prevent a low-priority process from being starved of CPU time.
`ts_lwait` (waitlevel)	Contains the new priority for a ready-to-run process that has exceeded the maximum wait time (`ts_maxwait`) without getting its full time slice.
`PRIORITY LEVEL`	Contains global priorities. Processes put in queues at the higher priority levels run first. The global priorities run from a high of `59` to a low of `0`. This is the only column in the table that is not tunable.

Real-Time Parameter Table

A default version of `rt_dptbl` is delivered with the system in the `/kernel/sched/RT_DPTBL` loadable module.

The results of the `dispadmin -c RT -g` command show a sample `rt_dptbl` below.

Figure B-2 Sample `rt_dptbl` Table

```
# TIME QUANTUM                PRIORITY
# (rt_quantum)                  LEVEL
      1000                #        0
      1000                #        1
      1000                #        2
      1000                #        3
      1000                #        4
      1000                #        5
      1000                #        6
      1000                #        7
      1000                #        8
      1000                #        9
       800                #       10
       800                #       11
       800                #       12
       800                #       13
       800                #       14
       800                #       15
       800                #       16
       800                #       17
       800                #       18
       800                #       19
       600                #       20
       600                #       21
      600                 #       22
       600                #       23
       600                #       24
       600                #       25
       600                #       26
       600                #       27
       600                #       28
       600                #       29
       400                #       30
```

```
# TIME QUANTUM                PRIORITY
# (rt_quantum)                  LEVEL
       400                #       31
       400                #       32
       400                #       33
       400                #       34
       400                #       35
       400                #       36
       400                #       37
       400                #       38
       400                #       39
       200                #       40
       200                #       41
       200                #       42
       200                #       43
       200                #       44
       200                #       45
       200                #       46
       200                #       47
       200                #       48
       200                #       49
       100                #       50
       100                #       51
       100                #       52
       100                #       53
       100                #       54
       100                #       55
       100                #       56
       100                #       57
       100                #       58
       100                #       59
```

Table B-3 describes the fields in the real-time parameter table

Table B-3 Fields in the `rt_dptbl` Table

Field Name	Description
`rt_glbpri`	Contains global priorities.Processes put in queues at the higher priority levels run first. Note that the `dispadmin` command, which you can use to display the table, shows only the relative priorities within the class, and not the global priorities. This column cannot be changed with `dispadmin`.
`rt_qntm`	Describes the default time slice (in milliseconds) a process with this priority (`rt_glbpri`) may run before the scheduler gives another process a chance. The time slice for a real-time process can be specified with the `-t` option of the `priocntl` command.

Kernel-Mode Parameter Table

The scheduler uses the kernel-mode parameter table, `ts_kmdpris`, to manage sleeping timesharing processes. A default version of `ts_kmdpris` is delivered with the system, in the `/kernel/sched/TS_DPTBL` loadable module, and is automatically built into the kernel as part of system configuration. See the reference page for `ts_dptbl(4)`.

Note – The kernel assumes that it has at least 40 priorities in `ts_kmdpris`. It panics if it does not.

The kernel-mode parameter table is a one-dimensional array of global priorities from 60 through 99. If a process owns a critical resource, it is assigned a kernel priority so that it can release the resource as soon as possible. Critical resources are:

- An exclusive lock on a page
- A read lock on a readers/writer lock

Prior to SunOS 5.3, processes were assigned kernel priorities while they were asleep. This ensured that the resources they were waiting for were not paged out before they had a chance to execute again.

In order to do this at SunOS 5.3, processes return from sleep with the highest time-share priority (59).

Error Messages C

This appendix documents the error messages generated by the accounting and ASET tools. The error messages are shown in `Courier font` and are listed in alphabetical order. For each message, the following is given:

Meaning: This section clarifies and expands the message.

Action: This section explains how to resolve or fix the problem that is causing the message to be issued.

Accounting Error Messages

`ERROR: locks found, run aborted`

Meaning: The files `lock` and `lock1` were found. Either two processes are trying to run `runacct` simultaneously or the last `runacct` aborted abnormally without cleaning up the locks. Check the `fd2log` for messages.

Action: Remove the files and restart `runacct`.

`ERROR: acctg already run for today's date: check /var/adm/acct/nite/lastdate`

Meaning: The date in `lastdate` and today's date are the same.

Action: Remove `lastdate`.

`ERROR: turnacct switch returned rc=n`

Meaning: The `accton` program must be owned by `root` and have the `setuid` bit set.

Action: Check the integrity of `turnacct` and `accton`. Make sure the `accton` program is owned by `root` and the `setuid` bit is set.

`ERROR: Spacctn.MMDD already exists: file setups probably already run`

Meaning: The `Spacctn.`*MMDD* file exists.

Action: Check status of files, then run setups manually, if necessary.

`ERROR: /var/adm/acct/nite/wtmp.MMDD already exists: run setup manually`

Meaning:`/var/adm/wtmp` has already been copied to `/var/adm/acct/nite/wtmp.`*MMDD*.

Action: Run the set up manually.

`ERROR: wtmpfix errors see /var/adm/acct/nite/wtmperror.MMDD`

Meaning: `wtmpfix` detected a corrupted `wtmp` file.

Action: Use `fwtmp` to correct the corrupted file.

`ERROR: Invalid state, check /var/adm/acct/nite/statefile`

Meaning: The file, `statefile`, is probably corrupted.

Action: Check `statefile` and read `active` before restarting.

`ERROR: runacct called with invalid arguments`

Meaning: You have typed the command incorrectly.

Action: Check your parameters and try again.

ASET Error Messages

This section lists and describes the error messages produced by ASET.

```
ASET failed.
Usage:   aset  [-n user[@host]]
               [-d aset_dir]
               [-l sec_level]
               [-u user_file]
               [-p]
```

Meaning: A command line syntax error was made when you invoked ASET.

Action: Re-enter the command following the correct syntax. Refer to the `aset`(1M) manual page for a detailed explanation of the command line syntax.

```
ASET failed: no mail program found.
```

Meaning: ASET is directed to send the execution log to a user, but no mail program can be found.

Action: Install a mail program.

```
Usage: aset [-n user[@host]] in /bin/mail or
/usr/ucb/mail.
```

```
Cannot decide current and previous security levels.
```

Meaning: ASET cannot determine what the security levels are for the current and previous invocations.

Action: Ensure the current security level is set either through the command line option or the `ASETSECLEVEL` environment variable. Also, ensure that the last line of `ASETDIR/archives/asetseclevel.arch` correctly reflects the previous security level. If these values are not set or are incorrect, specify them correctly.

```
ASET working directory undefined.
To specify, set ASETDIR environment variable or
use command line option -d.
ASET startup unsuccessful.
```

Meaning: The ASET working (operating) directory is not defined, or defined incorrectly.

Action: Use the `ASETDIR` environment variable or the `-d` command line option to specify it correctly, and restart ASET.

```
ASET working directory $ASETDIR missing.
ASET startup unsuccessful.
```

Meaning: The ASET working (operating) directory is not defined, or it is defined incorrectly. This may be because the `ASETDIR` variable or the `-d` command line option refers to a nonexistent directory.

Action: Ensure that the correct directory— that is, the directory containing the ASET directory hierarchy—is referred to correctly.

```
Cannot expand $ASETDIR to full pathname.
```

Meaning: ASET cannot expand the directory name given by the `ASETDIR` variable or the `-d` command line option to a full path name.

Action: Ensure that the directory name is given correctly, and that it refers to an existing directory to which the user has access.

```
aset: invalid/undefined security level.
To specify, set ASETSECLEVEL environment variable or
use command line option -l, with argument= low/med/high.
```

Meaning: The security level is not defined or it is defined incorrectly. Only the values `low`, `med`, or `high` are acceptable.

Action: Use the `ASETSECLEVEL` variable or the `–l` command line option to specify one of the three values.

```
ASET environment file asetenv not found in $ASETDIR.
ASET startup unsuccessful.
```

Meaning: ASET cannot locate an `asetenv` file in its working directory.

Action: Ensure there is an `asetenv` file in ASET's working directory. See the `asetenv(4)` manual page for the details about this file.

```
filename doesn't exist or is not readable.
```

Meaning: The file referred to by *filename* doesn't exist or is not readable. This can specifically occur when using the `-u` option where you can specify a file that contains a list of users whom you want to check.

Action: Ensure the argument to the `–u` option exists and is readable.

```
ASET task list TASKLIST undefined.
```

Meaning: The ASET task list, which should be defined in the `asetenv` file, is not defined. This can mean that your `asetenv` file is bad.

Action: Examine your `asetenv` file. Ensure the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure the file is intact. See the `asetenv(4)` manual page for the content of a good `asetenv` file.

```
ASET task list $TASKLIST missing.
ASET startup unsuccessful.
```

Meaning: The ASET task list, which should be defined in the `asetenv` file, is not defined. This can mean that your `asetenv` file is bad.

Action: Examine your `asetenv` file. Ensure the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure the file is intact. See the `asetenv(4)` manual page for the content of a good `asetenv` file.

```
Schedule undefined for periodic invocation.
No tasks executed or scheduled. Check asetenv file.
```

Meaning: ASET scheduling is requested using the `-p` option, but the variable `PERIODIC_SCHEDULE` is undefined in the `asetenv` file.

Action: Check the `User Configurable` section of the `asetenv` file to ensure the variable is defined and is in proper format.

```
Warning! Duplicate ASET execution scheduled.
Check crontab file.
```

Meaning: ASET is scheduled more than once. In other words, scheduling is requested while a schedule is already in effect. This is not necessarily an error if more than one schedule is indeed desired, just a warning that normally this is unnecessary since you should use the `crontab(1)` scheduling format if you want more than one schedule.

Action: Verify, through the `crontab(1)` command interface, that the correct schedule is in effect. Ensure that no unnecessary `crontab` entries for ASET are in place.

Index

A

E

F

G

I

Q

R

S

T

U

V

W