System Administration Volume 1

CONCURRENT COMPUTER CORPORATION™

0890429-080

October 2002

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Concurrent Computer Corporation makes no warranties, expressed or implied, concerning the information contained in this document.

To report an error or comment on a specific portion of the manual, photocopy the page in question and mark the correction or comment on the copy. Mail the copy (and any additional comments) to Concurrent Computer Corporation, 2881 Gateway Drive Pompano Beach, FL 33069. Mark the envelope **"Attention: Publications Department**." This publication may not be reproduced for any other reason in any form without written permission of the publisher.

Printed in U. S. A.

Revision History:		Level:	Effective With:
Original Release	-- July 28, 1994	000	OS Release 1.1
Previous Release	-- October 2001	070	OS Release 5.1
Current Release	-- October 2002	080	SAR 631

Volume 1 Contents

Chapter 1 Introduction to Basic Administration

Part 1 Setting Up the System

Chapter 2 Setting Up the Work Environment

Chapter 3 Booting and System States

Chapter 4 Creating and Managing User Accounts

Chapter 5 Managing Ports

Chapter 6 Collecting Data on System Use

Chapter 7 Installing Add-on Software

Chapter 8 Directories and Files

Chapter 10 Trusted Facility Management

Part 2 Security Administration

Chapter 11 Introduction to Security

Chapter 12 Installing Software on an Enhanced Security System

Chapter 13 Maintaining an Enhanced Security System

Chapter 14 User Account and Group Management

Chapter 15 Administering Printers, Terminals, and Devices

Chapter 16 File Protection

Chapter 17 Administering Mandatory Access Control and Multilevel Directories

Chapter 18 Trusted Backup and Restore

Chapter 19 Security Procedures

Appendix A Interrupts

Glossary

Volume 1 Index

Volume 2 Index

Illustrations

Screens

Tables

Volume 2 Contents

Chapter 1 Introduction to Basic Administration

Part 1 File System and Storage Device Administration

Chapter 2 Managing Storage Devices

Chapter 3 Managing File System Types

Chapter 4 File System Problems

Part 2 System Performance Administration

Chapter 5 Managing System Performance

Chapter 6 Managing Dynamically Loadable Modules

Chapter 7 Tunable Parameters

Chapter 8 Configuring and Building the Kernel

Chapter 9 Process Scheduling

Part 3 Backup and Restore Services

Chapter 10 Archiving and Restoring Data

Chapter 11 Backup and Restore Services

Part 4 Print Service Administration

Chapter 12 Basic Print Service

Chapter 13 Advanced Print Service

Part 5 The sysadm Interface

Chapter 14 Using the sysadm Interface

Chapter 15 Customizing the sysadm Interface

Chapter 16 Modifying the sysadm Interface

Glossary

Volume 2 Index

Ilustrations

Screens

Tables

1
Introduction to Basic Administration

1 Introduction to Basic Administration

How to Use This Book

System Administration is a book in two volumes that will help you understand the job of a system administrator and tell you how to set up, configure, and maintain the PowerMAX OS™[1]. The PowerMAX OS is based on UNIX®[2] System V Release 4.2 ES/MP. For reasons of brevity, the term "operating system" or "OS" will sometimes be used throughout this manual.

You may be personally responsible for maintaining a single computer. Or you may be an administrator for a large organization in which many users share a network of computers. In either case, this book will help you install and maintain various services on your system, and serve the needs of your users.

PowerMAX OS includes multiprocessing capabilities. With these capabilities come the associated need to manage multiple processors and processes. There are new commands that allow administrators to control the availability of processors, to display information about processors, and to bind processes to processors. This binding can be exclusive, that is, it can exclude all other user processes. In addition, there is a new entity, the lightweight process (LWP). This entity can be bound to a processor and it is the basic entity that is scheduled for the OS. These new features are described, where appropriate, in these two volumes.

This book explains how to do administrative tasks using the UNIX command line interface, often referred to as the "shell." We assume that you know how to enter commands at the shell prompt, and that you understand such UNIX system fundamentals as the directory structure and the shell. You should also feel comfortable working with the computer hardware itself; you should know how to boot your computer, how to shut it down, and how to install peripherals (such as modems, terminals, and printers). For information on these and other hardware topics, see your computer installation manual and any documents that came with the peripherals.

New Administrators

If you have no experience as a UNIX system administrator, this book adds structure to what can seem a confusing tangle of individual commands. Begin by reading *"Introduction to System Administration"*, which describes the duties of an administrator, suggests how to organize those duties, and tells you where this book offers more information about them.

[1] PowerMAX OS, Night Hawk, PowerMAXION and Power Hawk are trademarks of Concurrent Computer Corporation.

[2] UNIX is a registered trademark, licensed exclusively by X/Open Company Ltd.

Once you have a general idea of what's involved, you can read individual chapters as necessary, to learn about particular tasks you need to do. Many of the tasks described in this book are required of all administrators. Some of the tasks you may never need to do, depending on your resources and your users.

Experienced Administrators

If you are an experienced administrator you'll probably use this book as a reference to procedures. When you want more information than is covered here, you'll find it in the relevant on line manual pages.

How This Book Is Organized

System Administration is divided into two volumes, each containing parts covering discrete topics. The next sections describe the contents of volume 1 and volume 2.

What's In Volume 1

Volume 1 of *System Administration* contains two parts. Part 1, "*Setting Up the System*", begins where the applicable platform *Release Notes* ends: it assumes you've finished installing the software on your computer. Now you're ready to power up the computer and set up your system. It includes the following chapters:

- "*Setting Up the Work Environment*" describes some tasks you may need to complete or re-execute following installation of the system. It covers changing basic system parameters if necessary (such as the date and time on your computer), assigning passwords to system logins or administrative commands, and local memory administration.
- "*Booting and System States*" explains how start and stop the system and how to do tasks that affect the way in which your computer operates, or that provide information on the current state of your computer.
- "*Creating and Managing User Accounts*" tells you how to set up and control accounts for users and user groups, file and directory access, and command authorization on your computer.
- "*Managing Ports*" tells about the Service Access Facility, and how to administer the **`listen`** and **`ttymon`** port monitors.
- "*Collecting Data on System Use*" tells how to monitor system use—by time, user account, or specified software—of the resources on your system. Use these programs to bill users and create optimization strategies for resource usage. "Directories and Files" provides a map to the system directories and files that you will need to know about as a system administrator.

- “I*nstalling Add-on Software*” tells how to install software packages or sets on your system, how to store packages on your system for later installation, and how to remove packages.
- “*Directories and Files*” provides a map to the system directories and files that you will need to know about as a system administrator.
- “Administering Privilege” discusses executable file privileges, process privileges, the privileged user mechanism, the least privilege mechanism, and how to choose and enable a privilege mechanism.
- “Trusted Facility Management” gives administrative users specific guidelines on how to install, configure, and run the TFM database, how to administer roles and individual commands, how to use the tfadmin command, and other ways to grant privilege.

Part 2, “Security Administration” includes the following chapters:

- “Introduction to Security” will help you understand the security needs of your system, and the role that you play in assuring system security as an administrator. It is intended to describe the rationale for security, how the various mechanisms are implemented on the system, and the procedures you should follow to keep your system secure.
- “Installing Software on an Enhanced Security System” provides administrative users specific guidelines on how to install, configure, and run the Operating System (OS) with the Auditing and Enhanced Security Utilities installed. It illustrates the intended use of the features available to administrators.
- “Maintaining an Enhanced Security System” describes procedures and guidelines you should use in maintaining your Enhanced Security system.
- “User Account and Group Management” describes items of particular security relevance when creating new accounts and managing existing accounts. This chapter contains guidelines on items involved in managing and creating user accounts of particular security relevance.
- “Administering Printers, Terminals, and Devices” covers the particular security aspects of administering devices.
- “File Protection” discusses file attributes that have security relevance. Attributes specific to device files are discussed in the previous chapter.
- “Administering Mandatory Access Control and Multilevel Directories”. The Mandatory Access Control (MAC) mechanism enforces access restrictions that do not depend on the actions of the user. MAC is based on the comparison of security levels that are assigned to users, processes, files, and other objects. MAC supplements Discretionary Access Control (DAC) to prevent accidental or malicious disclosure of sensitive information.
- “Trusted Backup and Restore” tells you how to perform trusted backups and how to restore backed-up data if the Enhanced Security Utilities are installed on your system. The trusted backup commands save both the data and file security attributes necessary for maintaining security.
- “Security Procedures” tells how to set up a security policy, and gives procedures for checking user accounts, devices, and files. It tells how to check

for files with fixed privileges, etc., how to use cron, how to check the TFM database, and how to check the level databases.

- "Appendix A, Interrupts" describes the interrupt structure for Night Hawk, PowerMAXION and Power Hawk systems.
- Alphabetical Index for Volume 1 and Volume 2. The Volume 2 Index refers to subject material contained in Volume 2 of the *System Administration* Manual (Pubs No. 0890430).

What's In Volume 2

Volume 2 of *System Administration* contains five parts.

Part 1, "File System and Storage Device Administration", contains the following chapters:

- "*Managing Storage Devices*" tells how to add, remove and maintain devices such as disks, tape drives, CD-ROM and ethernet/FDDI controllers. This chapter also provides information on Virtual Partition (VP), what it is, and how to configure and de-configure VP in your system.
- "*Managing File System Types*" tells how to create and maintain each of the following types of file systems:
  - **ufs**
  - **xfs**
  - **sfs**
- "*File System Problems*" tells how to check file systems for consistency.

Part 2, "System Performance Administration" contains the following chapters:

- "*Managing System Performance*" describes ways to monitor and enhance the performance of your system.
- "*Managing Dynamically Loadable Modules*" tells how to add, remove and maintain DLMs on demand. This chapter also explains the concepts underlying the mechanism.
- "*Tunable Parameters*" describes procedures for modifying tunable parameters contained in the Operating System. The chapter also provides guidelines about when and what you should tune, and instructions for reconfiguring the operating system to enable new parameters.
- "*Configuring and Building the Kernel*" explains how to configure and deconfigure the kernel, how to configure kernel modules, how to configure non-required kernel modules, how to update hardware adapter files, and how to build and install the kernel. Also describe the config utility, which is a menu-driven centralized utility that provides a front-end interface to the current configuration method. Also discussed is methods on how to reduce kernel size.

- "*Process Scheduling*" explains the working of the system scheduler, a program that determines when processes run and thus greatly affects performance. Once you understand how the scheduler works, you'll be able to (a) judge whether you need to change the default tuning (to improve performance), and (b) foresee problems that may arise from the misuse (whether accidental or intentional) of its functions.

Part 3, "Backup and Restore Services" contains the following chapters:
(**Note**: This service only supported by PowerMAX OS release 4.2 and earlier.)

- "*Archiving and Restoring Data*" describes services for archiving and restoring data. Chapter covers: tape archiving and restoring using tar and cpio, dumping and restoring files using fsdump and fsrestore, and the comparison between the fsdump/fsrestore and backup/restore service utilities and how to restore file systems using the menu driven File System Restore Utility.

- "*Backup* and *Restore Services*" describe services that allow you to make copies of files at regular intervals, and provide copies to users who lose original files. Backing up your system's files to another medium is an important means of protecting your system from loss of data. A solid backup schedule will ensure that files that are lost or damaged could be restored later from backup copies. The backup and restore services contain a comprehensive set of utilities and files for customizing your system's backup and restore procedures.

Part 4, "Print Service Administration" contains the following chapters:

- "*Basic Print Service*" describes basic Line Printer services. Information includes how to configure a single printer on a single computer, manage everyday aspects of printers for your users, such as manage printer classes, print queue priorities, and troubleshooting.

- "*Advanced Print Service*" describes configuring printers on a network. In addition, information on administering print filters, and pre-printed forms, is given. There are also hints for troubleshooting network printing problems and customizing the print service.

Part 5, "The sysadm Interface" contains the following chapters:

- "*Using the sysadm Interface*" contains detailed instructions for using the **`sysadm`** command. For the most common administrative procedures, a user-friendly menu interface is provided with this command. It includes a sample walk-through for one menu and definition of all the menu system components.

- "*Customizing the sysadm Interface*" describes how you can modify the sysadm interface to suit your environment.

- Alphabetical Index for volume 2 subject material.

Related Documents

The area of network administration is large and complex, and the operating system includes several approaches to it. For that reason, we present networking topics separately, in a book titled *Network Administration.* Because you may be setting up network services using a mix and match of applications and protocols, *Network Administration* is organized as seven parts that address the major topics listed below.

- Part 1, “Network Services Administration”

  This part is an overview of networking. It provides information on selecting a network and setting up name-to-address mapping. It discusses the connection server (which establishes connections for network services that communicate over TLI connection-oriented and dialup connections), using authentication schemes (for additional system security), and setting up and administering ID mappings (for users on remote systems). It also covers administration and use of the Basic Networking Utilities (BNU) (for communicating to other systems that support BNU), and interactive remote execution (REXEC) utilities (to allow remote administration of a machine).

- Part 2, “Mail Service Administration”

  This part describes administration of the online facility that allows users to exchange messages. Once basic networking is configured, you don't need to do any additional administration to use the mail facility. This part, however, will help you set up some special features, such as establishing a domain name, setting up mail directories to be shared across a networked file system, and setting up a connection to another site that uses the Simple Mail Transfer Protocol (SMTP).

- Part 3, “TCP/IP Network Administration”

  This part provides information needed to set up and run TCP/IP on your system. The discussion includes information about configuring Internet addresses and describes how to use TCP/IP commands and files to implement a wide range of TCP/IP features. First, it introduces you to important TCP/IP concepts you should be familiar with as an administrator. Next, it steps you through some basic TCP/IP administrative tasks. Finally, it describes some features in depth, such as domain name service and troubleshooting. Topics include the concepts you need to understand to effectively administer your system, step-by-step procedures for many basic administrative tasks, how to expand and manage growing systems by setting up and using routers and subnets, how to use SNMP to do network monitoring and management functions, concepts and procedures relating to domain name service, how to diagnose problems and tune your system to improve TCP/IP performance, how to obtain and complete IP address registration forms, how to obtain and complete domain name registration forms, and how to synchronize time among the machines on your network.

- Part 4, “Distributed File System Administration”

  This part describes the DFS command interface for NFS. For example, the DFS software provides you with the **`share`** command, which allows you to share a resource on your system using NFS. DFS Administration is covered in the following chapters: “Introduction to DFS Administration”, “Setting Up DFS”, “Using DFS Commands and Files”, and “DFS sysadm Interface”.

- Part 5, “Network File System Administration”

  This part tells you how to set up and maintain NFS on your system, including how to share and mount resources, how to mount resources automatically using a feature called the automounter, and how to set up Secure NFS. NFS Administration is covered in the following chapters: “Introduction to NFS Administration”, “Setting Up NFS”, “Sharing and Mounting NFS Resources Explicitly”, “Obtaining NFS Information”, “Troubleshooting and Tuning NFS”, “Setting Up Secure NFS”, “Using the NFS Automounter”, “The NFS Network Lock Manager”, and “Using the NFS sysadm Interface”.

- Part 6, “Remote Procedure Call Administration”.

  This part tells you how to administer the files used by RPC, a mechanism for resource sharing between hosts used by NFS and NIS. Information about setting up and establishing secure RPC domains is also provided.

- Part 7, “Network Information Service Administration”

  This part explains how to set up, administer, and update NIS, a distributed database service used for password and host file administration.

- Glossary

  Contains definitions for terms and abbreviations used throughout this book.

Introduction to System Administration

To help you get started, this section provides an overview of the tasks and features that make up system administration. It then points you to the section of the book where you can find more information. Before you use this book you may find it useful to become familiar with the shell and file editors, as well as system features such as file permissions and logins. To do this, we recommend you read the *User's Guide.*

Chapter 11, *“Introduction to Security”* in volume 1, explains the concept of security embodied in the Enhanced Security Utilities and gives an overview of security concepts and mechanisms. The section *“How the Components of the System Work Together”* in Chapter 11, explains in more detail the assignment of security levels and the access checking mechanisms and algorithms as they apply with the Enhanced Security Utilities installed.

If you intend to administer your system according to B2/B1 security criteria, as established by the *Trusted Computer System Evaluation Criteria* (DoD 5200.28-STD, 1985), see the section entitled *“System Startup and Security”* in Chapter 11.

Administrative Interfaces

Almost all the procedures in this book can be done by issuing shell level commands. The descriptions and procedures in each chapter are presented in terms of shell level commands. Basic shell administrative commands are in the base operating system when you install it. The OS provides a non-graphical menu interface that helps you do administrative functions without using shell commands. It is accessed through the **sysadm** command. When the functions described in a chapter can be done through the menu interface, brief instructions for invoking the appropriate menu are included in a section near the end of the chapter. Because the menu interface is self-explanatory (it includes on-line help), this book does not explain how to use it to complete particular tasks. However, "Using the sysadm Interface", defines all the components of the menu system, and walks you through one menu.

Administration Procedures

The administrative facilities available can be intimidating for the first-time system administrator. To help you become comfortable with administration, we have provided an overview of some of the most important administrative procedures.

Set Up

Most aspects of system setup are handled automatically through the installation procedure. However, you may find you need to change some system parameters, or assign administrative logins before you can use it effectively. This information includes setting: the date, time and time zone; the system name; and the communications node name.

Accurate date/time stamps will ensure accurate representations of when files are created, processes are run, and mail is transferred. The system name and communication node name identify your system to the applications and users that access it.

For information on how to set up your system, see Chapter 2, "*Setting Up the Work Environment*" in volume 1.

Privileges

Privilege, in the simplest terms, is the ability to override system restrictions on the actions of users. All operating systems allow users to exercise special privilege, under certain conditions, to perform sensitive operations. Sensitive system operations are those which affect the configuration of the system or its availability to users.

Most users cannot, for example, execute commands affecting the hardware or software configuration of the system. Activities such as mounting and checking file systems, adding users, modifying user profiles, adding and removing peripherals, installing application software, password administration, and administration of the user terminal lines, are restricted to certain users.

In previous UNIX releases, the restriction of privilege is implemented by designing a special user identifier (UID) of 0; the login name historically associated with this UID is `root`.

When a person logs in as `root`, that person has unrestricted access to every file on the system, and the ability to alter system operation. Commands that execute sensitive system operations check to see whether the effective UID of the process requesting the operation is 0. If it is, the user process is given unlimited access to the system.

The `root` login in previous UNIX releases possesses, in effect, the one privilege necessary to override all system restrictions on command execution and access: the superuser privilege.

The OS provides an alternative privilege mechanism that is more flexible to suit the needs of the user community. Now, rather than investing the power to issue any command on the system to one user, you can give partial superuser power to several users. By assigning privileges linked to specific tasks, you essentially assign a role to each user.

The Trusted Facility Management tools (TFM) maintains a database of users and the commands they may execute with privileges. This database is set up automatically when your machine comes up; if you're the first person to set up a login name on the system, your login will be recorded in the TFM database and all available privileges will be given to you. Later, however, you can add the logins of others to whom you want to assign privileges for specific tasks. By assigning task-specific privileges in this database, you can avoid conferring the amount of authority that makes the term “superuser” (another popular term for the owner of UID `0`) meaningful.

This does not mean the UID of `0` is no longer significant. As the “owner” of your system (the first person to log in on a new system, which is usually the administrator), you'll still be assigned the UID of `0` and you'll still enjoy special privileges associated with that UID. Now, however, having that UID isn't the only way to obtain privileges.

Booting the System

Each time you power up your computer, a complex series of steps occur automatically to start the system. By default, **`/stand/unix`** is booted.

As part of the boot procedure, a set of processes are started and file systems are mounted as defined by the system state set for your system.

System States

The system state defines the levels of activity and accessibility for your system. In a single user state (1, S, or s), only one user can be active on the system and many of the file systems are not accessible. In multi-user state (2), other terminals can access the system and more local file systems are mounted. In Networking state (3), all multi-user processes are started, plus remote directories are mounted.

Shutdown system states are 0 (power down state) and 6 (stop, then reboot). (See the **`init(1M)`** online manual page for more details on system states.)

Multi-User State

The system boots to system state 2 (multi-user state) by default. The default system state is set by the **`initdefault`** line in the **`/etc/inittab`** file. All entries in the **`inittab`** file that correspond to the **`initdefault`** system state are run during the start-up process.

When the system boots up to system state 2, commands beginning with `S` in the **`/etc/rc2.d`** and **`/etc/dinit.d`** directories are also started. These commands do things like start print schedulers, clean up old spool files, and start up network daemons. (There are also directories for system states 0, 1, and 3.)

Some of the processes started up at boot time are run as background processes. These background processes run continuously, waiting for something to occur. Examples are the **`listen`** process, which monitors networking ports for incoming network requests, and **`ttymon`**, which listens to terminal ports for login requests.

To change your default system state, you would edit the **`/etc/inittab`** file and change the number `2` in the following line to the state number you want (`1` or `3`, for example).

```
is:2:initdefault:
```

Shutdown State

Before you turn off your system you must go to system state 0. When you go down to system state 0, commands beginning with `K` in the **`/etc/rc0.d`** directory are run. Primarily, these commands are run to kill daemon processes started in higher system states. Once the computer has completed its transition to state 0, you can turn off the computer.

The following is an example of the **`shutdown`** command:

```
shutdown -y -g0 -i0
```

Users and Groups

Definition of users and passwords is the most important means of protecting your system security. User and group assignments are used to define the ownership of files on the system and the accessibility of those files to other users.

User and group management can be broken down into the following types of duties:

- Default Environment - Before you add a user, you can set up a series of defaults that will apply to each user you add to your system. The defaults will define such things as the directory under which the user's home directory will be added (**`/home`**), the group the user will be a part of (**`other,1`**), the point at which the login will expire or become inactive after disuse, and the location of default files to be placed in the user's home directory (**`/etc/skel`**).

  The files in the **`/etc/skel`** directory are automatically copied to the user's home directory when the user is added to the system. A sane **`.profile`** is the most important default file to have in **`/etc/skel`**. In that file you can define the user's mail file, terminal type, path, and other information that will make the user's login immediately usable. When users log in, they can then tailor their **`.profile`** to their own needs.

There is also a system-wide profile (**`/etc/profile`**) that is executed each time the user logs in. It can be set up to do things that apply to each user, like print the message of the day, check if the user has mail, and list the availability of file system space.

- Adding Groups - By defining groups on your system, you can add a layer of accessibility that applies to a group of users, instead of just an individual user. After you create a group and add users to that group, those users can share files and directories that are not accessible to users outside that group.
- Adding Users - When you add a user to your system, you can use the defaults described above (or change them), assign that user to a group, and add a password for that user.

Once users are added to your system, you will have to support those users. The system provides methods for communicating with users immediately [**`wall(1)`** command] when they log in (**`/etc/motd`** file), or when they read the news [**`news(1)`** command].

Peripherals (Terminals, printers, networks, media)

As you add peripheral hardware to your computer, it will be up to you to identify each peripheral to the operating system. Peripheral connections are made through outlets on your computer called I/O (input/output) ports. Before you can use a port, you need to allocate it for use by a particular device. Instructions for doing this are in Chapter 5, "*Managing Ports*", in volume 1 of the *System Administration* book. Once you have allocated the ports on your system, connect your terminals, printers, and modems. For details about physically installing each peripheral, see the hardware manual that accompanies it.

Terminals

When you connect a terminal to your system, you need to define the device number associated with the port and the line speed. There is also a set of terminal attributes you can modify as needed. Procedures for adding and removing terminals are contained in Chapter 5, "*Managing Ports*" in volume 1 of the *System Administration* book.

Printers

If you want to make printers available to your users, you should install the printers and the LP print service software. See your printer installation manual for hardware installation instructions. The "*Basic Print Service*" and "*Advanced Print Service*" chapters contained in *volume 2*, describe how to add a printer and provide a full description of the LP service.

Networks

When describing computer networks, it is useful to make a distinction between media (the hardware that carries information from one computer to another) and services (the software that lets you use the network for file transfer, remote login, and remote execution).

Documentation that comes with the media (communications boards, modems, etc.) usually describe how to set the media up on your system. Networking services considered to be part of the system include Basic Networking Utilities (BNU), Remote Execution Utilities (REXEC), TCP/IP Network Services, Network File System (NFS), Remote Procedure Call Services (RPC). and Network Information Service (NIS) services.

Setting up and administering your system networking services is described in the *Network Administration* guide.

Media (Hard Disks and Tapes)

One important category of peripheral devices is storage media, such as hard disks and tape drives. To learn how to install storage devices, see the "*Managing Storage Devices*" chapter in volume 2 of the *System Administration* book. That chapter also describes how to format media, label them, and partition them.

Back up and restore facilities are described in their own chapters later in this book. Those facilities are used to create copies of your stored data, usually on removable media so you can restore the data later in case it is lost or destroyed.

Communicating with Users

As an administrator, you will frequently want to send messages to users. To do so, use any of the following four tools.

Message of the Day

When you want to ensure that every user who logs in to the system will see an announcement or inquiry, simply add your message to the **/etc/motd** file. When a user logs in, all messages in that file are displayed, as in the following example:

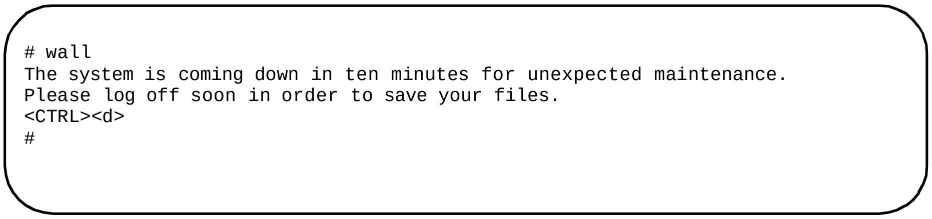
```
The system will be down from 1700 to 2300 hours on
Friday, September 30, for upgrades and preventive
maintenance.
```

Edit the **/etc/motd** file regularly to remove obsolete notices.

The wall Command

When you really need to get in touch with users fast, use the **`wall`** command to write to all who are logged-on.

It is good practice to reserve the **`wall`** command for those times when you need to ask users to log off quickly. For example, you can warn users of unexpected shutdowns:



```
# wall
The system is coming down in ten minutes for unexpected maintenance.
Please log off soon in order to save your files.
<CTRL><d>
#
```

While **`wall`** is unmatched for getting urgent information out quickly, some users dislike having their work interrupted by an uninvited message. Many users guard against this by including the command **`mesg -n`** in their **`.profile`**. This command blocks the output of **`wall`** sent by ordinary users. But a user with appropriate privileges can execute the **`wall`** command and override the **`mesg -n`** command. See the **`wall(1)`** in the online manual page for more information.

News

A somewhat less intrusive way to get information to your users is the **`/var/news`** command. It displays the contents of message files that have been placed in a designated directory. As with a bulletin board, anyone can post messages and anyone who “passes by” can read them.

To post a news item:

1. Create a file and type your message in it. A filename that suggests the content can be helpful later in identifying files you may want to edit or remove.
2. Move this message file to the **`/var/news`** directory:

   `mv` *filename* `/var/news`

News items remain in **`/var/news`** until they are removed.

Unlike the message of the day, which users cannot turn off, **`news`** can be totally ignored. To read news items, a user must type **`news`** at the shell prompt. Only then will the computer display news items posted since the last time the user executed **`news`**.

Users have a choice of ways to read the news: select only some items, read and delete items, ignore all items, or remove all items. See the **`news(1)`** online manual page for more information.

If you don't want to leave it entirely to chance, there are a couple of things you can do to make it more likely that your users will read news regularly. You can add the **news** command with no options as the last line of the default user profile (**/etc/skel/.profile**) so that new users will have this command in their **$HOME/.profile** files. Or you can ask users to add **news** as the last line in their **.profile** files themselves. Doing so causes news items to be displayed upon logging in, just before the shell prompt appears.

Electronic Mail

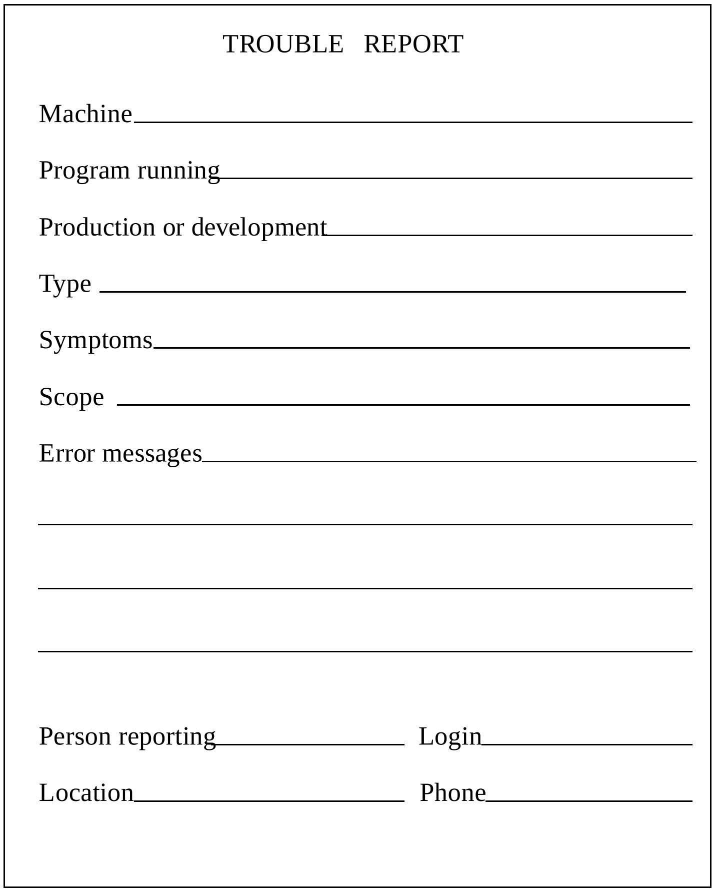
The **mail** and **mailx** commands allow you and your users to communicate with each other more privately. If your system is part of a network, you can also use these commands to communicate with people on other systems.

mail is the basic command for sending and receiving messages. **mailx** builds upon **mail** by adding to it additional features that are useful for storing messages in files, adding headers, and so on. If you use **mailx**, you may find it helpful to use a file called **.mailrc** to customize its behavior to suit your needs. For details about using this file, see the **mailx(1)** online manual page.

Collecting Users' Requests

From time to time, you'll need to collect forms and feedback from users. You may find it useful to keep your own log of problems reported by users (in addition to the system log described in the “*Security*” chapters under “Part 2” of volume 1). Users' problems fall into patterns, and by keeping a record of how you resolve them, you can avoid reinventing the wheel when a problem recurs.

We also strongly recommend you provide users with a formal mechanism for reporting problems. The form shown in Figure 1-1 is an example of typical trouble report that you can use to keep track of system problems.



TROUBLE REPORT

Machine______________________________

Program running__________________________

Production or development____________________

Type _______________________________

Symptoms_____________________________

Scope ______________________________

Error messages___________________________

____

____

____

Person reporting____________ Login____________

Location________________ Phone____________

Figure 1-1. Typical Trouble Report Form

Notation Conventions Used in This Book

This section describes the notation conventions used in this book.

- References to literal computer input and output (such as commands entered by the user or screen messages produced by the system) are shown in a `monospace` font, as in the following example:

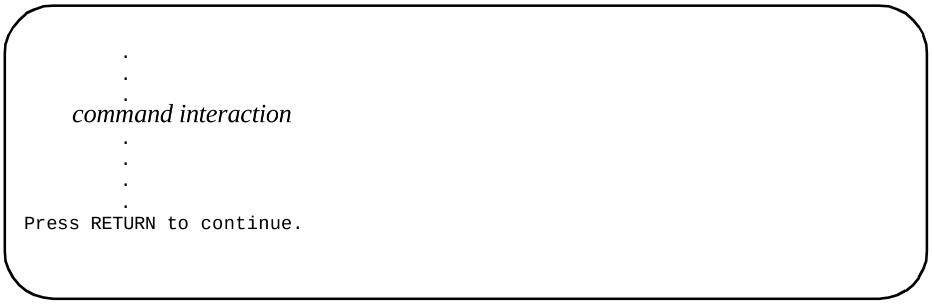
```
$ ls -l report.oct17
-rw-r--r--   1 jim   doc   3239 May 26 11:21 report.oct17
```

- Commands that are too long to fit on one line are separated by a backslash (\). This is not a character to be typed, but indicates that the command line continues on one line.
- Substitutable text elements (that is, text elements that you are expected to replace with specific values) are shown in an *italic* font, as in the following example:

$ **cat** *file*

The *italic* font is a signal that you are expected to replace the word *file* with the name of a file.

- Comments in a screen display**-that** is, asides from the author to the reader, as opposed to text that is not computer output**-are** shown in an *italic* font and are indented, as in the following example:



```
       .
       .
       .
   command interaction
       .
       .
       .
       .
Press RETURN to continue.
```

- Instructions to the reader to type input usually do not include explicit instructions to press the <RETURN> key at the appropriate times (such as after entering a command or a menu choice) because this instruction is implied for all system commands and menus.

In one circumstance, however, an instruction to press the <RETURN> key is explicitly provided: when, during an interactive routine, you are expected to press <RETURN> without having typed any text, an instruction to do so will be provided, as follows:

```
Type any key to continue: <RETURN>
$
```

- Keyboard references are sometimes shown with the key graphic. <Enter> and <Esc> are two examples.
- Control characters are shown by the string <CTRL>-*char* where *char* is a character such as “d” in the control character <CTRL><d>. To enter a control character, hold down the <CTRL> key and press the letter shown. Be sure to type the letter exactly as specified: when a lowercase letter is shown (such as the “d” in the example above), enter that lowercase letter. If a character is shown in upper case (such as <CTRL><D>), you should enter an upper case letter.
- The system prompt signs shown in examples of interactive sessions are the standard default prompt signs.
  - the dollar sign **($)** for an ordinary user
  - the pound sign **(#)** for the owner of the **root** login.

Referenced Documentation

The following manuals are referenced in this manual:

HN6200 Console Reference Manual	0830047
HN6200 Architecture Manual	0830048
HN6800 Console Reference Manual	0830045
HN6800 Architecture Manual	0830046
PowerMAXION Console Reference Manual	0830052
PowerMAXION Architecture Manual	0830053
Motorola SBC Console Reference Manual	0830050
Power Hawk Series 700 Console Reference Manual	0830059
PowerMAX OS Programming Guide	0890423
Device Driver Programming	0890425
Users Guide	0890428
System Administration Volume 2	0890430
Audit Trail Administration	0890431
Network Administration	0890432
Compilation SystemsVolume 1 (Tools)	0890459
Compilation SystemsVolume 2 (Concepts)	0890460
Documentation Overview	0890470
VERITAS Volume manager (VxVM) *System Administrator's Guide*	0890471
VERITAS Volume Manager (VxVM) *User's Guide*	0890472

The following release notes are referenced in this manual:

HN6200/HN6800 PowerMAX OS Release Notes	0890454-(reln, e.g. 3.1)
Motorola SBC PowerMAX OS Release Notes	0891058-(reln, e.g. 3.1)
Power MAXION PowerMAX OS Release Notes	0891061-(reln, e.g. 3.1)

The following release notes are referenced in this manual: (Cont)

TurboHawk PowerMAX OS Release Notes	0891071-(reln, e.g. 3.1)
Power Hawk Series 700 PowerMAX OS Rel. Notes	0891084-(reln, e.g. 3.1)

1
Setting Up the System

Replace with Part 1 tab

Part 1 - Setting Up the System

2
Setting Up the Work Environment

2
Setting Up the Work Environment

Introduction

NOTE

(When your system is being run in compliance with the security criteria described in Part 2 of this book, “Security Administration’, the setup procedures in this chapter are governed by the setup procedures in Chapter 10, “Installing Software on an Enhanced Security System” This chapter (“Setting Up the Work Environment”), and the applicable platform Release Notes should be consulted by first-time users.

Many tasks traditionally associated with the initial setup of the Operating System (OS) are now accomplished automatically during the installation process. (See the applicable platform Release Notes for detailed information). Even so, you may want to change some administrative parameters, or set some that weren't set during installation (such as passwords for system logins), before users are allowed on the system. This chapter gives instructions for using shell commands to change the parameters defined during the system installation process. The on-line system manual pages provides detailed descriptions of the shell commands.

If the Operations, Administration and Maintenance (OA&M) package (a non-graphical menu interface) is installed on your system, you can use it to complete many of these tasks. (See *“System Setup through OA&M Menus”* later in this chapter.)

Overview of System Setup

Once you've completed the installation process, most administrative parameters will be set up for your system. You may also have created logins for some users. Following installation, you can change those administrative parameters, add logins for other users, and assign system passwords. This chapter provides instructions for the following tasks:

- set or change system parameters
  - system date and time
  - system name or node name
  - user login names
  - system passwords

Additional Setup Tasks

In addition to the setup procedures described in this chapter, there are a few other tasks you may want to complete before people start using the system. The following is a list of those tasks with references to the instructions for them.

- Setting up security level definitions for the Mandatory Access Control mechanism: see Chapter 17, “Administering Mandatory Access Control and Multilevel Directories” in this book.
- Populating the Trusted Facility Management Database so users have the privileges required to perform privileged tasks: see Chapte r9“Administering Privilege” and Chapter 10, “Trusted Facility Management” in this book.
- Setting security-related parameters for user logins: see Chapter 14, “User Accounts and Group Management” in this book.
- Setting the Secure Attention Key (SAK) for all system ports: see Chapter 15, “Administering Printers, Terminals and Devices” in this book.
- Configuring and enabling the audit mechanism if you have auditing on your system: see *Audit Trail Administration.*
- Configuring and enabling port monitors for terminals and other system access ports: see Chapter 5, “Managing Ports for more information.
- Installing terminals for users: see the appropriate terminal manuals.
- Creating file systems for users: see the “Managing File System Types” chapter in volume 2 of the *System Administration* book.
- Formatting disk, cartridge tapes, etc: see the “Managing Storage Devices” chapter in volume 2 of the *System Administration* book.
- Installing local and remote printers: see the appropriate printer manuals.
- Installing other software packages depending upon the intended use of your system. (See the applicable platform Release Notes for further information.)
- Changing the default system state: see Chapter 3, “Booting and System States”.
- Adding other user logins: see Chapter 4, “Creating and Managing User Accounts” for further information.
- Changing users' passwords: see Chapter 14, “User Account and Group Management” for instructions on using **passwd**.

Changing System Parameters

Most system setup tasks that you do during installation are usually done only once. However some tasks must be repeated every so often. For example,

- As system passwords and administrative command passwords age, you must change them.
- If your system has been corrupted or down for any period of time, you must reset the system date and time.
- When users leave your system, you should change system passwords, administrative passwords, and any security passwords.
- If your system has been added to a network, and your existing system name and node name are already being used by another system, you must change your system name and node name.

This section describes how to add or change system parameters that you set during initial system installation or setup.

Changing the System Date and Time

3. Set the date and time for your system by issuing the **`date`** command as follows:

   `date` *`MMddhhmm`*`[`*yy*`]`

   where *MM* is the month, *dd* is the day, *hh* is the hour (in 24-hour clock format), *mm* is the minute, and *yy* is the last two digits of the year (with the current year being the default). For example, to set the date to October 6, 12:45 A.M., of the current year, enter the following:

   ```
   date 10060045
   ```

   Adjustments for daylight savings time will be made automatically once you have set the system date and time.

4. To change the time zone for your computer, edit the file **`/etc/TIMEZONE`** and change the line which reads

   `TZ=`*value*

   where *value* must have two parts; *std* and *offset*. For example, `EST5` can be the value of `TZ` for the east coast of the United States (EST meaning “Eastern Standard Time,” and `5` meaning a five hour offset from Coordinated Universal Time. If your time zone has daylight savings time, that can be specified by adding the appropriate code to *value*. For example, the east coast USA example becomes `EST5EDT` when the daylight savings code is added. The following table shows the time zone codes for the USA:

Table 2-1. USA Time Zone Codes

Zone Name	Standard Time Code	Daylight Savings Code
Greenwich	GMT	GDT
Atlantic	AST	ADT
Eastern	EST	EDT
Central	CST	CDT
Mountain	MST	MDT
Pacific	PST	PDT
Yukon	YST	YDT
Alaska	AST	ADT
Bering	BST	BDT
Hawaii	HST	

See **`environ(5)`** for information on defining the time zone in more detail.

Creating or Changing Administrative Commands and System Login Passwords

Many administrative commands are used not only by the system administrator, but by conventional users, as well. However, giving users access to the **`root`** login so that they can use these commands is undesirable for security reasons. To allow access to administrative commands without allowing use of the **`root`** login, assign passwords for the commands themselves, just as you assign passwords for user logins.

In addition, the operating system provides eight login names for important directories and commands that are usually used by administrators, but that are sometimes needed by users, as well. Only users with a need to know should be given these passwords.

By using one of these logins, users can access the desired directory or command. They will, however, be prompted for the password first. When the command has finished executing, the user will be returned to the shell, or to a **`login:`** prompt if he or she logged in initially with an administrative login.

You can use the **`passwd`** command to create passwords for these special commands and logins.

Using passwd to Protect Administrative Commands and System Logins

To assign a password to an administrative command or a system login, follow the steps below.

1. If you don't know what commands or logins might need a password, execute

   ```
   logins -s
   ```

   to see a list of all system logins and administrative commands that can take a password.

2. To create a new password for an administrative command or system login, or to change an existing one, run the **`passwd`** command as follows:

   `passwd` *cmd_name*

   or

   `passwd` *login_name*

   If a password already exists, you will be asked to enter it. Then you will be asked for the new password and, after you enter it, you'll be asked to verify it. (Passwords are not displayed on the screen.)

If you are assigning a password to a system login, *login_name* can be one of the following:

`root` This login has no restrictions; it overrides all other logins, protections, and permissions. By entering the command **`su root`** at the system prompt, or by typing **`root`** at the login prompt, a user gains access to the entire operating system. The password for **`root`** should be carefully protected.

`uucp` Owns some administrative files in **`/usr/lib/uucp.`**

`nuucp` Used by remote computers to log on to the system and start file transfers from **`/var/spool/uucppublic.`**

`login_name` may also be one of the following, but these should be considered place holders for file ownership, not valid logins.

`sys`	Owns many system files
`bin`	Owns most of the commands in **`/usr/bin`**
`adm`	Owns some administrative files in **`/var/adm`**
`daemon`	controls background processes, (the system daemon)
`lp`	Owns the object and spooled data files

If you want to create more than a few passwords (or if you don't know which commands or logins still require passwords), you may want to use the system administration menus instead. See *“System Setup through OA&M Menus”* later in this chapter for details on the use of the menu interface. If you want to change an existing password, however, you cannot use the menus; you must use the **`passwd`** command.

Changing the System Name and Node Name

Every computer has a system name and a node name. These names are defined by assigning values to the system name parameter and the node name parameter. The node name must be unique within the network.

These names are normally defined during installation (see the *Release Notes* for detailed information), but they may be changed later.

Begin by reading the guidelines for selecting a node name in the "Network Services" chapter in *Network Administration.* Once you have selected a node name you can assign it to the node name parameter by issuing the **setuname** command. You can run **setuname** from either multi-user state or single-user state but, in either case, you must be logged in as **root**.

Run the **setuname** command with the **-n** option, as follows:

```
setuname -n node_name
```

If necessary, you can run the **setuname** command with the **-s** option to change the system name, as follows:

```
setuname -s system_name
```

After you run **setuname**, remove the file **/etc/confnet.d/inet/up.save,** and reboot the system. Rebooting will cause some system files that use the system or node name to be updated. Some other system files will have to be updated manually.

When you change a system name, node name, or network address, other software may break unexpectedly, so it is good practice not to change them once they are defined. If you do change these identifiers, you will need to locate all system files that refer to them, check them, and possibly update their values. (You will also need to check and update references to other computers on your network when they get changed.)

The list which follows names most of the files that you will need to check. Not all of the files listed will exist on all computers. Nor is this an exhaustive list: for example, some third-party application programs put the system name or node name into files that they supply. The files you will need to check fall into three categories:

- those that contain node or system names but not network addresses

```
$HOME/.rhosts
/etc/X0.hosts
/etc/dfs/dfstab      #may contain names of NFS clients
/etc/dfs/lid_and_priv
/etc/hosts.equiv
/etc/inet/inetd.conf
/etc/lp/Systems
/etc/mail/names
/etc/net/ticlts/hosts
/etc/net/ticots/hosts
/etc/net/ticotsord/hosts
/etc/nodename
/etc/rc2.d/S11uname
/etc/rc2.d/S18setuname
```

```
/etc/vfstab          #may contain names of NFS servers
/etc/vfstab          #may contain names of NFS servers
```

- those that contain network addresses but not node or system names

```
/etc/inet/networks
/etc/inet/rc.inet    #may contain subnet information
/etc/inet/rc.inet    #may contain subnet information
/etc/resolv.conf
/etc/saf/tcp/_pmtab
```

- those that contain network addresses and node or system names

```
/etc/confnet.d/inet/interface
/etc/inet/hosts      # this file is linked to /etc/hosts
/etc/uucp/Systems*
```

The files in **`/etc/confnet.d/inet`** should be updated using the **`configure(1M)`** command, as in **`configure -i`**.

The file **`/etc/nodename`** is updated by running **`uname -S`** or **`setuname.`**

NOTE

If the LP print service is installed on your system when you change the node name, you must restart it after making the change.

System Information

To look up the existing system name and node name, use the **`uname(1)`** command with the options **`-s`** and **`-n`** as follows:

```
uname -s -n
```

The system will respond by displaying two names, such as

```
UNIX_SV localB
```

Here the system name is `UNIX_SV`, and the node name is `localB`.

In addition to this information, you might want to look up other facts about your system. Other options, including **`-a`**, a convenient way of getting all available information about your system at once, are described on the online manual page **`uname(1)`**.

Configuring Multiprocessors

The operating system includes multiprocessing capabilities. With these capabilities come the associated need to manage multiple processors and processes. A new software key mechanism limits the number of users and processors.

Using the appropriate console commands, processors can be placed in the on line or off line mode prior to boot. Refer to the **tu** and **td** commands described in the *Console Reference Manual* for more information on how to place processors in the on line or off line mode.

The **psrinfo** command allows you to display information about processors.

The **run**, **rerun** and **pbind** commands allow you to control which processes run on each processor in the system.

The new command, **keyadm(1M)**, and **licensekeys(4)**, the serial number database file, are part of the new software key mechanism.

User and Processor Limits

The Operating System Multiprocessor includes a mechanism that coincides with the license fee structure for this release. This software key feature limits the number of processors that can be used when running a specific operating system binary. The software key also controls the number of concurrent users.

There is a processor-limit-base key and a user-limit-base key. To increase the processor limit, the administrator licenses an upgrade consisting of some distribution media and documentation. The new command, **keyadm,** sets and displays resource limits, where the resources are USERS and PROCESSORS. This command manages the **licensekeys(4)** format.

Processor Identification

Processors are identified with a processor ID number that gives them a unique tag within the system. The state of the processors in your system can be examined by using the **psrinfo(1M)** command.

All processors are on line after bootup, and processors may not be taken off line while the system is up.

Binding Processes to Processors

The executable entity for the operating system is a light-weight process. By default, an LWP can execute on any processor in the system. You can use the **run** and **rerun** commands to bind a process (all the associated LWPs) or an LWP to a set of processors, or you can use the **pbind(1M)** command to bind a process (all the associated LWPs) or LWP to a single processor. Binding LWP restricts the LWP to execute only on the specified processor(s). Once bound, all child processes created by this process, are bound by default to the same processor(s). The **run**, **rerun** and **pbind(1M)** commands can also be used to unbind a process (all the associated LWPs) or LWP from processors. Once unbound, the process or LWP can execute on any processor in the system.

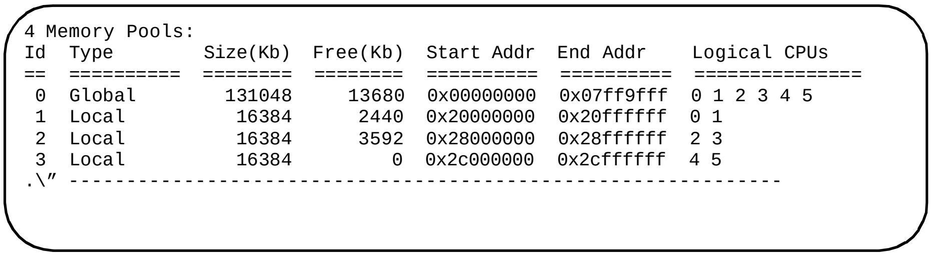
Local Memory Administration

Local memory is a pool of memory that is located on a processor board. A local memory pool is shared by all of the processors that reside on that board. Local memory is a feature that improves the performance of multiprocessor systems. This section describes the commands needed by the System Administrator to:

- See the size of local memory
- See the local memory physical address space
- Check local memory utilization
- See how memory pools are allocated
- See how much local memory is free
- Establish default NUMA policies
- Query the current default NUMA policies
- Load the kernel text into local memory

Query for Size of Local Memory

Use the **`hwstat`** command to query the system for information about local memory. When the **`hwstat`** command is issued, a full screen of information is displayed. Screen 2-1 depicts just a partial view showing only those parameters related to local memory.



```
4 Memory Pools:
Id  Type        Size(Kb)  Free(Kb)  Start Addr  End Addr    Logical CPUs
==  ==========  ========  ========  ==========  ==========  ===============
 0  Global        131048     13680  0x00000000  0x07ff9fff  0 1 2 3 4 5
 1  Local          16384      2440  0x20000000  0x20ffffff  0 1
 2  Local          16384      3592  0x28000000  0x28ffffff  2 3
 3  Local          16384         0  0x2c000000  0x2cffffff  4 5
.\” ----------------------------------------------------------------
```

Screen 2-1. Local Memory Usage, Static Picture

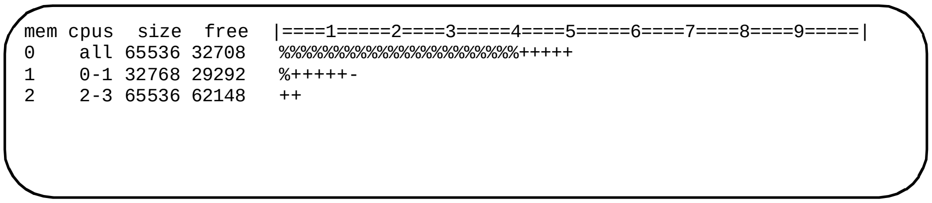
Several column headings provide specific information about local memory. They are:

Id	identifier number assigned to the memory pool
Type	global or local memory
Size	size of the memory pool
Free	amount of memory not used

Start Addr	starting physical address of memory pool
End Addr	last physical address of memory pool
Logical CPUs	the logical CPUs that are not remote to the memory pool

Checking Local Memory Utilization

Use the **`mpstat`** command to display various statistics about local memory. When the **`mpstat`** command is issued, a screen of information is displayed. Scree n2-2 depicts just a partial view showing only those parameters related to local memory.



```
mem cpus  size  free  |====1=====2====3=====4====5=====6====7====8====9=====|
0    all 65536 32708   %%%%%%%%%%%%%%%%%%%%%%+++++
1    0-1 32768 29292   %+++++-
2    2-3 65536 62148   ++
```

Screen 2-2. Local Memory Statistics, Dynamic Picture

The four column headings special to local memory are:

mem shows the different memory pool identifiers. For example: 0 for global, 1 through 2 for local memory.

cpus shows how memory pools are allocated to CPUs. For example; memory pool 2 in Screen 2-2 is allocated to CPUs 2 and 3.

size indicates the total amount of memory in kilobytes for each memory pool.

free indicates the amount of free memory in kilobytes for each memory pool.

The horizontal bar chart shows percentages up to 100%. Three different symbols are used in the bar chart:

% represents the fraction of the memory pool containing locked pages.

+ represents the fraction of the memory pool containing pages that are in use but not locked.

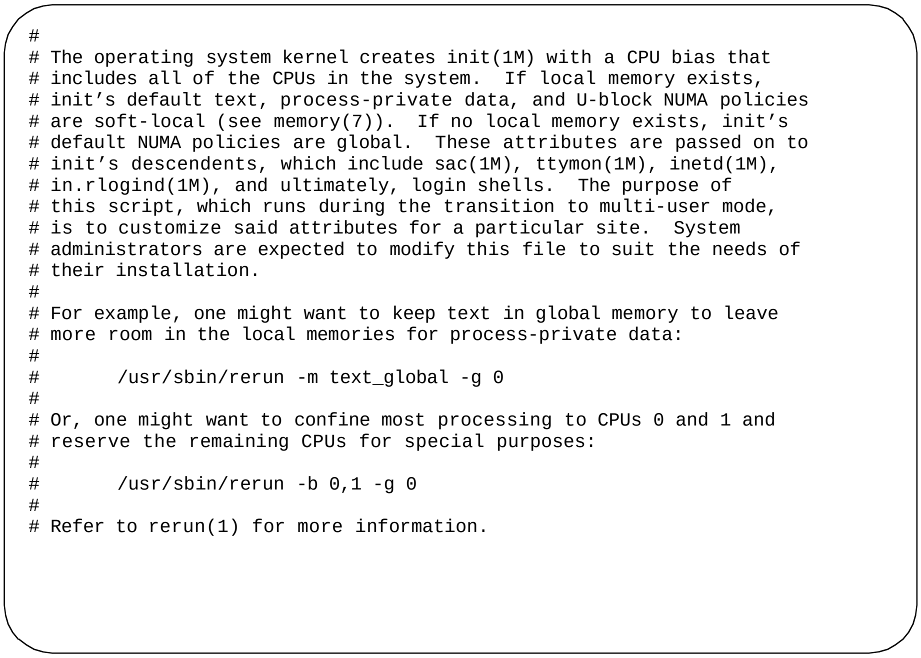
- represents the fraction of the memory pool containing free, dirty pages. Free dirty pages must be cleaned before they can be reused.

Blank represents the fraction of the memory pool containing free clean pages.

Refer to the `mpstat`(1) man page for more information, and the arguments applicable to this command.

Establishing Default NUMA Policies

Default NUMA policies affecting all processes running on a system may be set by the system administrator by altering the NUMA policies of the **init** process as set in the **rerun** script in the **/etc/init.d** directory. Guidelines for determining the appropriate default NUMA policy are outlined in the "Memory Management" chapter of the *PowerMAX OS Programming Guide. Screen 2-3* shows a typical set of statements that can be used in the rerun script.



```
#
# The operating system kernel creates init(1M) with a CPU bias that
# includes all of the CPUs in the system.  If local memory exists,
# init’s default text, process-private data, and U-block NUMA policies
# are soft-local (see memory(7)).  If no local memory exists, init’s
# default NUMA policies are global.  These attributes are passed on to
# init’s descendents, which include sac(1M), ttymon(1M), inetd(1M),
# in.rlogind(1M), and ultimately, login shells.  The purpose of
# this script, which runs during the transition to multi-user mode,
# is to customize said attributes for a particular site.  System
# administrators are expected to modify this file to suit the needs of
# their installation.
#
# For example, one might want to keep text in global memory to leave
# more room in the local memories for process-private data:
#
#       /usr/sbin/rerun -m text_global -g 0
#
# Or, one might want to confine most processing to CPUs 0 and 1 and
# reserve the remaining CPUs for special purposes:
#
#       /usr/sbin/rerun -b 0,1 -g 0
#
# Refer to rerun(1) for more information.
```

Screen 2-3. Default NUMA Policies Set by the rerun **Script**

Query the Current Settings of the Memory Flags

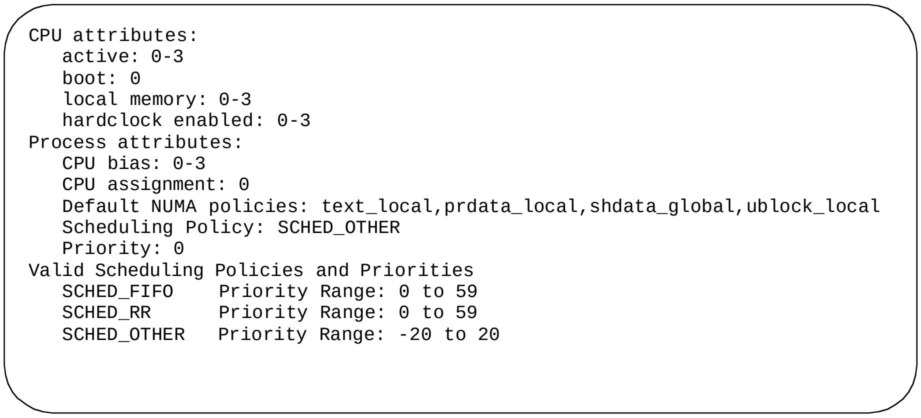
The binding of the text, private data, shared data, and U-block regions of a process to local memory is explicitly controlled by the **memdefaults(2)** system call, or by using the **run(1)** or **rerun(1)** commands.

By using the **memdefaults(2)** system call in a program you can query the current settings of the process memory flags. You can also type **run** by itself to see a display of the defaults.

Screen 2-4 depicts a partial view showing the output of the **run** command. Note the line "Default NUMA policies" which shows bindings for the process.

The binding of shared memory regions to local memory is explicitly controlled by the value of the **shmflg** argument on a **shmget(2)** system call.

.



```
CPU attributes:
   active: 0-3
   boot: 0
   local memory: 0-3
   hardclock enabled: 0-3
Process attributes:
   CPU bias: 0-3
   CPU assignment: 0
   Default NUMA policies: text_local,prdata_local,shdata_global,ublock_local
   Scheduling Policy: SCHED_OTHER
   Priority: 0
Valid Scheduling Policies and Priorities
   SCHED_FIFO    Priority Range: 0 to 59
   SCHED_RR      Priority Range: 0 to 59
   SCHED_OTHER   Priority Range: -20 to 20
```

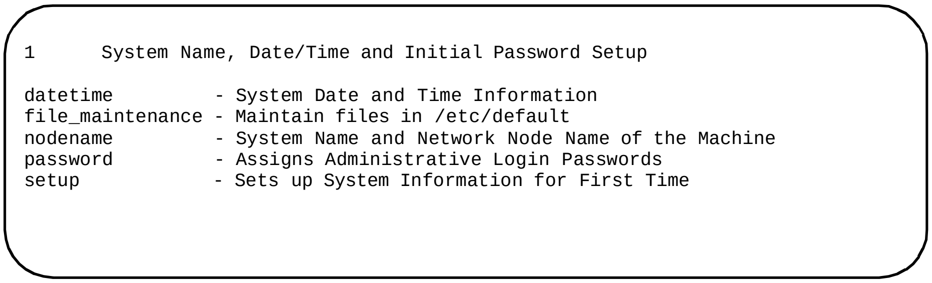
Screen 2-4. Query for Local Memory Binding Defaults

How to Load Kernel Text into Local Memory

By default, the operating system itself resides entirely in global memory, leaving all local memory for use by applications. Kernel text, however, can be replicated in selected local memory pools by setting the value of the corresponding KTEXTLOCAL*n* system tunable parameter to 1 (*n* denotes a number ranging from **1** to **4**, where **1** represents the first local memory pool, **2** the second, and so on, in order according to processor board slot number). After changing a tunable parameter, you must rebuild the kernel by using **idbuild(1M)** and then reboot your system. For additional information on tunable parameters, refer to the “Tunable Parameters” chapter of *System Administration Volume 2*. Note that when kernel text is replicated in local memory, the original copy of kernel text remains in global memory.

System Setup through OA&M Menus

The system administration menus are only available if the Operations, Administration and Maintenance (OA&M) package is installed on your system. To access the menu for system setup, type **sysadm system_setup**. The following menu will appear on your screen:



```
1      System Name, Date/Time and Initial Password Setup

datetime         - System Date and Time Information
file_maintenance - Maintain files in /etc/default
nodename         - System Name and Network Node Name of the Machine
password         - Assigns Administrative Login Passwords
setup            - Sets up System Information for First Time
```

Screen 2-5. Main Menu for System Setup

The following table shows how the tasks listed on the `system_setup` menu correspond to the shell commands discussed throughout this chapter.

Table 2-2. sysadm Tasks and Shell Commands

Task to Be Performed	**`sysadm`** Task	Shell Command
Set up or display the system date and time	`datetime`	**`date(1)`**

Set up or display the system name and nodename of a machine	`nodename`	**`setuname(1M)`**
		`uname(1M)`

Maintain files in **`/etc/default`**	`file_maintenance`	**`defadm(1M)`** or perform manually
Display the system name and nodename of a machine	`nodename`	**`uname(1M)`**
Assign or change administrative passwords	`password`	**`passwd(1)`**

Quick Reference to System Setup

- Change passwords for administrative functions:

  `passwd` *login_name*
  or
  `passwd` *cmd_name*

- Display the system name and node name:

  `uname` **`-s -n`**

- Set or change the system name:

  `setuname` **`-s`** *system*

  where *system* is an alphanumeric string.

- Set or change the node name:

  ```
  setuname -n node
  ```

  where *node* is an alphanumeric string.

- Set or reset the system clock:

  ```
  date MMddhhmm[yy]
  ```

  where *MM* is the month, *dd* is the day in the month, *hh* is the hour (24-hour system), *mm* is the minute, and *yy* is the last two digits of the year with the current year as the default. (The *yy* argument is optional.) To set your local time zone, use the **`setup`** command. The **`date`** command adjusts the system date and time as necessary for daylight savings time.

- Shut down to single-user state:

  ```
  shutdown -y -i1
  ```

- Return to multi-user state:

  ```
  init 2
  ```

3
Booting and System States

3
Booting and System States

Introduction

Managing a computer involves many responsibilities. Typically, management tasks affect the operation of the computer as a whole, and every user on the computer. This chapter tells you how to use shell commands to do tasks that affect the way your computer operates or that provide information on the current state of your computer, and covers the following:

- A brief overview of the boot procedure that explains how the operating system is loaded from disk into memory and then executed.
- An overview of system states that explains what happens after the computer is booted to the default system state, how to change the default system state, and how to change system states after power up.
- Detailed procedures on how to achieve fast boot up and fast shut down.
- Detailed procedures for shutting down and rebooting your system, and displaying information about system configuration and about users.

If installed, online manual pages (man pages) provide detailed descriptions of the shell commands. For additional information about managing your computer, see your hardware manual.

If the Operations, Administration and Maintenance (OA&M) package (a non-graphical menu interface) is installed on your system, you can use it to complete many of these tasks. See *"Machine Management through OA&M Menus"* later in this chapter for details.

Overview of the Boot Procedure

Root disk is the disk that the system is booted from. See Figure 3-1 for the general layout of an Operating System (OS) root disk that is divided into default slices. The disk layout shown in the figure is a generalized root-disk layout; your disk may have additional slices. (For default slice layouts, see *"Device Names and Default Partitions"* in the chapter "Managing Storage Devices" in volume 2 of the *System Administration* book.)

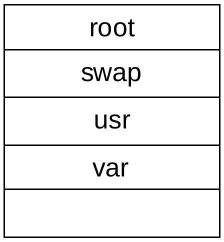


Figure 3-1. Generalized Hard Disk Default Slicing

On most computers, booting is a multi-step process that results in the load of the bootable operating system (**`/stand/unix`**).

The bootable operating system `unix` is found in the **`root`** slice. This slice contains all the bootable programs and data files used during the boot procedure.

Root contains **`/stand.`** The contents of **`/stand`** include the following:

`unix`

This is the bootable operating system. When the boot program is loaded, it searches for the `unix` program, and then loads it into memory. Once `unix` is loaded, the boot program passes control to it. Once the bootable operating system is running, various system daemons are started, and the system enters one of several system states, sometimes called “init states.” (See the description of **`/etc/inittab`** under “*Early Initialization*” later in this chapter.)

You can enter `unix` or **`/stand/unix`** at any prompt requiring the name of the bootable operating system.

`boot`

Loader for any standalone program. See the online manual page **`boot(8)`** for more information.

`fastcopy`

`fastcopy` will copy one file system to another. A file system is defined by a device name, a controller number, a drive number, a partition number and a bus number. `dsk(0,0,0,0)`, means: copy from disk, controller 0, drive 0, partition 0, and bus 0. See the online manual page **`fastcopy(8)`** for more information.

`ls`

For each directory argument, **`ls`** lists the contents of the directory; for each file argument, **`ls`** repeats its name and inode number. See the online manual page **`ls(8)`** for more information.

`cat`

For the name specified, **`cat`** (catenate and print), prints the file on the console terminal. See the online manual page **`cat(8)`** for more information.

`format`

`format` is a standalone program which is used to format, verify, or define partition sizes for disk drives on a disk controller. See the online manual page **`format(8)`** for more information.

dlvia — **dlvia** is a standalone program used to modify (download) the Interface Adapter (VIA) operational firmware. See the online manual page **dlvia(8)** for more information.

For more information on the operating system files in **/stand,** see *"Configuring the Operating System"* (under "Managing System Performance")" in volume 2 of *System Administration.*

Boot Scenarios

There are several reasons for a system boot to take place:

- A reboot of the computer is explicitly requested while the operating system is running (for example: **shutdown -i6.** In this case, all system activity is stopped (all processes are killed, and so on). The system checks to see whether a new `unix` needs to be configured, and if so, configures one. Then, the system unmounts all file systems and boots `unix.` The system comes up in the state defined by the **initdefault** entry in **/etc/inittab.**
- A crash occurs and the computer automatically reboots. In this case, the procedure is the same as for the first case, above.
- The operator boots the system via the console terminal (see section "System Specific For Booting and Taking Down the System".) On power up, the system boot occurs automatically.

See *"Configuring the Operating System"* (under "Managing System Performance") in volume 2 of *System Administration* for a description of the configuration process.

System States

Once booted, the operating system operates in one of seven system states — software configurations of the system under which only selected groups of processes exist.

You may be familiar with system states as "init states," "run states," "run levels," or "run modes." These terms are synonymous; for the sake of simplicity, however, we'll use only one term (system state) in this chapter.

You can change from one system state to another with the **init** or **shutdown** (which calls **init)** commands. For details about these commands, see the appropriate reference online manual pages. Table 3-1 shows the factory-configured states in which your system can operate.

Table 3-1. Factory Configured States

System State	Description
`0`	Shutdown state. In this state, you can safely turn off the computer's power.
`1`	Administrative state. File systems required for multi-user operations are mounted and logins requiring access to multi-user file systems can be used. When the system is going from state 2 to state 1, some services are stopped and some processes are killed; otherwise, the system continues operating as it did in state 2.
`s` or `S`	Enter single-user state. When the system changes to this state as the result of a command, the terminal from which the command was executed becomes the system console. (The terminal device is linked to **`/dev/syscon`**.) This is the only system state that doesn't require the existence of a properly formatted **`inittab`** file. If this file does not exist, then by default, the only legal system state that **`init`** can enter is the single-user state. The set of filesystems mounted and the list of processes killed when a system enters state `s` are not always the same; which filesystems are mounted and which processes are killed depends on the method used for putting the system into state `s` and the rules in force at your computer site. The following paragraphs describe state “ `s`” in three circumstances: when the system is brought up to `s` with **`init`**; when the system is brought down (from another state) to `s` with **`init`**; and when the system is brought down to `s` with **`shutdown`**. When the system is brought up to `s` with **`init`**, the only filesystems mounted are `/` (root), and **`/var`**. (Three filesystem types, **`/proc`**, **`/system/processor, and /dev/fd`** are also mounted.) Filesystems for users' files are not mounted. With the commands available on the mounted filesystems, you can manipulate the filesystems or transition to other system states. Only essential kernel processes are kept running. When the system is brought down to `s` with **`init`**, all mounted filesystems remain mounted and all processes started by **`init`** that are running in the multi-user state are killed. Because all login related processes are killed, users cannot access the system while it's in this state. In addition, any process for which the `utmp` file has an entry will be killed. This last condition ensures that all port monitors started by the Service Access Controller (SAC) will be killed and all services started by these port monitors, including `ttymon` login services, will be killed.

Table 3-1. Factory Configured States (Cont.)

System State	Description
s or S	(The SAC is a daemon that maintains the port monitors on a server computer in the state specified by the system administrator.) Other processes not started directly by `init` (such as `cron`) will remain running. When you change to `s` with `shutdown`, the system is restored to the state in which it was running when you first booted the computer and came up in single-user state, as described above.
2	Multi-user state. In this state, file systems are mounted and multi-user services are started. Multi-user is the default state upon bootup. Even if the Unlimited User Upgrade package is not installed on your computer, the system will initialize to state 2 by default, and will allow up to two users.
3	Networking state. Used to start Network File System (NFS), mount remote resources, and offer your resources automatically.
6	Stop and reboot the operating system to the state defined by the **`initdefault`** entry in **`/etc/inittab.`** If necessary, configure a new bootable operating system before the reboot. (The **`rc6`** procedure is invoked for this.)
Q or q	Re-examine the **`/etc/inittab`** file.

NOTE

In addition, you can define system state 4 or system state a, b, or c. System state 4 is provided strictly for your convenience; it's not used in the delivered operating system. Similarly, system states a, b, and c are pseudo-states that may be used to run certain commands without changing the current system state. Process those **`/etc/inittab`** file entries assigned the a, b, or c system state. These states are supported by **`init`** but, like system state 4, are not used in the delivered system.

As delivered, the system enters the multi-user state (state 2) on power up. File systems are mounted, daemons started, and system services are made available. These activities are performed by the **`rc2`** script. [See **`rc2(1M)`**.]

Not all activities, however, should be performed in multi-user state. For example, you should never change your system's configuration while users are accessing it because data may be lost. By changing the system to its single-user state, you can be sure you are the only person logged on to the system and, therefore, that it is safe to perform critical system administration tasks, such as changing the system configuration.

The Single-User State (System States)

The parameters of single-user state differ depending on how the system enters it. Your computer may be in single-user state when:

- the system is brought up to `s` with **`init`**
- the system is brought down (from another state) to `s` with **`init`**
- the system is brought down to `s` with **`shutdown(1M)`**

When the system is brought up to `s` with **`init`**, the only file systems mounted are **`/`** (root) and **`/var.`** (Three file system types, **`/proc,`** **`/dev/fd,`** and **`/system/processor`** are also mounted.) With the commands available on these file systems, you can manipulate your file systems or transition your system to other states.

When the system is brought down to `s` with **`init`**, all file systems remain mounted and some processes (such as `tty-related` functions) are terminated. Because all login related processes are killed, users cannot access the system while it's in this state.

When you change to `s` with **`shutdown`**, the system is restored to the state in which it was running when you first booted the computer and came up in single-user state, as described above.

Entering the Multi-User State During Boot Up

When you reboot your system, it enters the default system state that appears in the **`initdefault`** line of the **`/etc/inittab`** file. Normally, this line contains a `2` in the second field, indicating that the system is to be put into multi-user state by default. The following is an example **`initdefault`** entry:

```
is:2:initdefault:
```

You can change the default system state of the computer by changing the second field of the **`initdefault`** line in your **`/etc/inittab`** file.

Once the computer boots and control passes to the **`unix`** program, the following events occur as the system enters multi-user state:

1. Early system initializations are started by **`init`**.
2. **`root`** is checked and mounted by **`ckroot.`**
3. File systems are checked and mounted by **`bcheckrc.`**
4. The system state change is prepared by the **`rc2`** procedure.
5. The system is made public via the spawning of the service access facility (**`ttymon`** and **`sac`**).
6. The **`dinit`** script completes initialization of processes that can be delayed until after a **`login`** prompt is displayed (such as **`lp`** and **`cron`**).

The most important events that occur as **unix** comes up in multi-user state are shown in Figure 3-2.

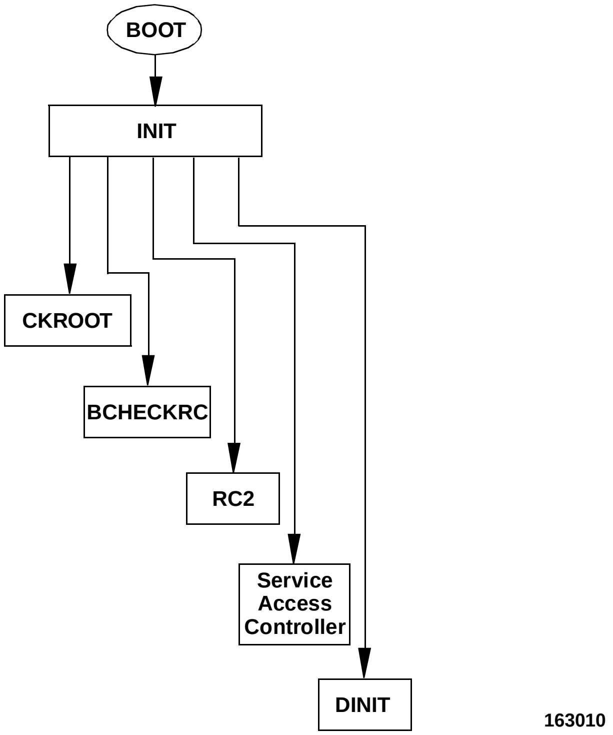


Figure 3-2. A Look at System Initialization

The **init** daemon spawns each of the processes shown in Figur e3-2 in the order shown (from left to right). When all these processes have been called, the system is in multi-user state (system state 2).

Early Initialization

After the operating system is loaded into core memory via the boot program (see *"Overview of the Boot Procedure"* early in this chapter), the **init** daemon is created. This process immediately scans **/etc/inittab** for entries of the type **sysinit**. A sample listing of these entries is shown in Scree n3-1.

```
swap::sysinit:/sbin/swap -a /dev/swap >/dev/sysmsg 2>&1
cr::sysinit:/sbin/ckroot >/dev/sysmsg 2>&1
mi::sysinit:/sbin/sh -c '[ -x /sbin/macinit ] && /sbin/macinit' >/dev/sysmsg
2>&1
mm::sysinit:/etc/conf/bin/idmodreq -c `/etc/conf/bin/idkname -c` >/dev/null
2>&1
ldmd::sysinit:/etc/conf/bin/idmodload >/dev/sysmsg 2>&1
ap::sysinit:/sbin/autopush -f /etc/ap/chan.ap
bchk::sysinit:/sbin/bcheckrc </dev/console >/dev/sysmsg 2>&1
key::sysinit:/sbin/sh -c '[ -x /sbin/keyadm ] && /sbin/keyadm -s USERS
PROCESSORS'
>/dev/sysmsg 2>&1
onl::sysinit:/sbin/psradm -n -a
bu::sysinit:/etc/conf/bin/idrebuild reboot </dev/console >/dev/sysmsg 2>&1
me::sysinit:/etc/conf/bin/idmkenv >/dev/sysmsg 2>&1
xdc::/sysinit:/sbin/sh -c 'if [ -x /etc/rc.d/es_setup ] ; then /etc/rc.d/
es_setup
; fi' >/dev/console 2>&1
ia::sysinit:/sbin/creatiadb </dev/console >/dev/sysmsg 2>&1
```

Screen 3-1. Sample `sysinit` Entries in an `/etc/inittab` File

These entries are executed in the order in which they are listed in the file.

Preparing the System State Change

Now the system must be placed in a particular system state. First, **`init`** scans each entry in the **`/etc/inittab`** file for the value **`initdefault`** in the third field (the “action” field). Including the value **`initdefault`** in the third field of an entry means the default system state is defined in the second field of that entry. For example, in the first line of Screen 3-2, the `2` in the second field, followed by the value **`initdefault`** in the third field, means the default system state for this system is system state 2 (multi-user state).

NOTE

> The examples shown in this section are specific to system state 2, but note that processing for system state 3 is very similar.

Once **`init`** has identified the default system state as system state 2, it searches the table for all entries that specify processes for system state 2. Typical system state 2 entries are shown in Screen 3-2.

```
is:2:initdefault:
bd:56:wait:/etc/conf/bin/idrebuild </dev/console >/dev/sysmsg 2>&1
r0:0:wait:/sbin/rc0 off >/dev/sysmsg 2>&1 </dev/console
r1:1:wait:/sbin/rc1 >/dev/sysmsg 2>&1 </dev/console
r2:23:wait:/sbin/rc2 >/dev/sysmsg 2>&1 </dev/console
r3:3:wait:/sbin/rc3 >/dev/sysmsg 2>&1 </dev/console
r5:5:wait:/sbin/rc0 firm >/dev/sysmsg 2>&1 </dev/console
r6:6:wait:/sbin/rc0 reboot >/dev/sysmsg 2>&1 </dev/console
sd:0:wait:/sbin/uadmin 2 0 >/dev/sysmsg 2>&1 </dev/console
fw:5:wait:/sbin/uadmin 2 2 >/dev/sysmsg 2>&1 </dev/console
rb:6:wait:/sbin/uadmin 2 1 >/dev/sysmsg 2>&1 </dev/console
li:23:wait:/usr/bin/ln /dev/systty /dev/syscon >/dev/null 2>&1
sc:234:respawn:/usr/lib/saf/sac -t 300
co:12345:respawn:/usr/lib/saf/ttymon -g -v -p “Console Login: “ -d /dev/console
-1 console
d2:23:wait:/sbin/dinit >/dev/sysmsg 2>&1 </dev/console
```

Screen 3-2. System State 2 Processes

The processes defined in these entries are executed before the system enters multi-user system state during power up or reboot. (See *“System State Directories”* later in this chapter.)

The fifth line shown in Screen 3-2 calls the **`rc2`** script. The **`rc2`** script accomplishes (among other things) the following:

- cleans up **`/tmp,`** removes any **`uucp`** logs
- sets up and mounts the file systems not mounted by **`bcheckrc`**
- starts network processes (if installed)

Next, **`init`** searches the **`/etc/inittab`** file for system state 2 processes and executes them in the order found in the file. In this example, **`init`** performs the following functions:

- starts the service access controller (**`sac`**) for the ports
- starts the tty monitor (**`ttymon`**) for the console

The system is now available for users to log on as shown by the **`login:`** prompt that appears on users' terminals.

Finally, the last line in the **`/etc/inittab`** file calls the **`dinit`** script. A sample of this line is shown in Scree n3-3.

```
d2:23:wait:/sbin/dinit >/dev/sysmsg 2>&1 </dev/console
```

Screen 3-3. The `dinit` Process

The **`dinit`** script completes initialization of processes that can be delayed until after a **`login`** prompt is displayed (thus reducing the amount of time a user has to wait for a login prompt). The directory **/etc/dinit.d** contains the initialization scripts for each of these processes. These include:

- starting the **cron** daemon
- making the LP print service available for use (if installed)
- making **uucp** available for use (if installed)

When this is complete, the full multi-user environment is established and the system is in system state 2.

Changing System States After Bootup

There may be times when you want to change system states after you have booted up your computer. For most administrative duties, you must change the system state from the multi-user state (system state 2) to system state `S, s`, or `1`.

To change states, the **shutdown** or **init** command can be executed with an argument that specifies the desired system state. Use of the **shutdown** command is generally recommended. (See Table 3-1 for a list of available arguments.)

Changing to Single-User State (System State “s”)

Some administrative functions can be done only when the system is in single-user state. The recommended way to go to single-user state is by running the **shutdown -is** command. This command executes all the files in the **/etc/rc0.d** directory by invoking the **/sbin/rc0** procedure. The **shutdown** command accomplishes, among other things, the following:

- closes all open files and stops all user processes
- stops all daemons and services
- writes all system buffers out to the disk
- unmounts all file systems necessary for multi-user operations, but not needed in single-user state (such as **/home)**

NOTE

An **init 1** issued from multi-user mode would function in a similar manner, but would not unmount file systems, and would not kill all processes and services.

There are two ways to change to single-user state:

- you can run the **shutdown -is** command (recommended)
- you can run the **init s** command

After these programs complete the change to single-user state, you get the message:

```
INIT: New run level: S
INIT: SINGLE USER MODE

Type <CTRL><d> to proceed with normal startup,
(or give root password for single-user mode)
```

Type the root password and press <RETURN>; you will get the single-user prompt (#). You are now ready to perform tasks that should be done only in single-user state.

Changing to Multi-User State (System State 2)

System state 2 is the normal operating state of the system when it is not connected to a network. In this system state, several users can be logged on, using the system's resources simultaneously. To change to multi-user state from single-user state, run:

```
init 2
```

This procedure executes the **`/sbin/rc2`** script, which runs processes in the **`/etc/rc2.d`** directory, and also runs **`/sbin/dinit,`** which runs processes in the **`/etc/dinit.d`** directory. Running **`init 2`** also initiates the Service Access Facility which manages access to your system through ports and other communication devices. (See Chapter 5, “Managing Ports” in this volume for details.)

CAUTION

> The **`/var`** file system must be mounted before you can execute **`init 2`** if it is a separate file system. Normally **`/var`** is mounted automatically during the boot up operation before entering single-user mode.

Changing to Networking State (System State 3)

Before you can perform administrative tasks associated with NFS, you must change the system to system state 3 (defined as the Networking state). To change to the Networking state, run:

```
init 3
```

This procedure executes the **`/sbin/rc2, /sbin/rc3,`** and **`dinit`** scripts. These scripts run processes in all directories associated with system states 2 and 3, respectively. Running **`init 3`** also initiates the Service Access Facility, which manages access to your system through ports and other communication devices. (See Chapter 5, “Managing Ports” in this volume for details.)

In addition to the scripts in **/etc/rc2.d** (the multi-user state directory), scripts in **/etc/rc3.d** (the Networking state directory) are run to do the following, if configured:

- advertise your resources to remote computers
- mount remote resources on your computer

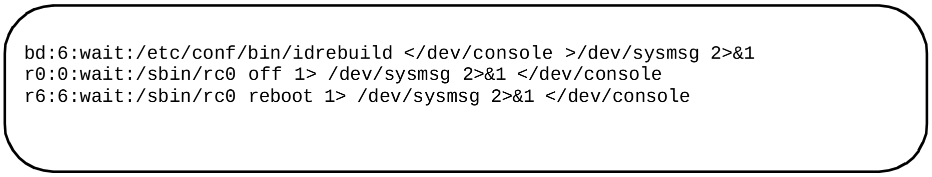
If you are in a Networking environment, you may want to change your **initdefault** entry in **/etc/inittab** from 2 to 3, so you automatically come up in Networking state when you reboot your computer.

CAUTION

The **/var** file system must be mounted before executing **init 3** if it is a separate file system. Normally **/var** is mounted automatically during the boot up operation before entering single-user mode.

Changing to the Reboot State (System State 6)

Rebooting the operating system (defined as system state 6) kills most active processes and forces the computer into a condition similar to its condition during power up. After you install a software package that requires reinitialization of the system for proper operation, reboot the computer.



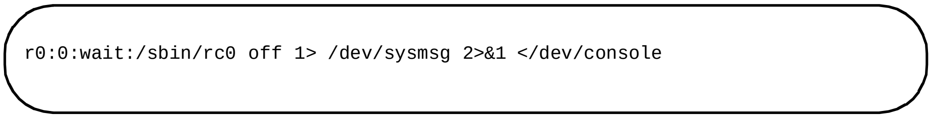
```
bd:6:wait:/etc/conf/bin/idrebuild </dev/console >/dev/sysmsg 2>&1
r0:0:wait:/sbin/rc0 off 1> /dev/sysmsg 2>&1 </dev/console
r6:6:wait:/sbin/rc0 reboot 1> /dev/sysmsg 2>&1 </dev/console
```

Screen 3-4. System State 6 Processes (from the Sample /etc/inittab File)

The **rc0** and **rc6** scripts are linked

Shutting Down the System

To shut down your system, run **shutdown -i0.** The following entries in the **/etc/inittab** file apply to shutting the system down:



```
r0:0:wait:/sbin/rc0 off 1> /dev/sysmsg 2>&1 </dev/console
```

Screen 3-5. System State 0 Processes (from the Sample /etc/inittab File)

The **rc0** procedure is called to clean up and stop all user processes, daemons and other services, and to unmount the file systems.

System State Directories

System states 0, 1, 2, and 3 each have a directory of files that are executed in transitions to and from that state. These directories are **/etc/rc0.d, /etc/rc1.d, /etc/rc2.d,** and **/etc/rc3.d,** respectively. States 2 and 3 also execute files in the **/etc/dinit.d** directory. Most files in these directories are linked to files in the **/etc/init.d** directory. Typically, their purpose is to start and stop various system services or daemons.

The system state files are named according to the following conventions:

S*NNname*
or
K*NNname*

Each file name consists of three parts:

S or K — The first letter specifies whether the process should be started (S) or killed (K) on entering the new system state.

NN — The next two characters form a number between 00 and 99. They show the order in which the files will be started (S00, S01, S02, and so on) or killed (K00, K01, K02, and so on).

name — The rest of the file name is the name of the file in the **/etc/init.d** directory to which this file is linked.

For example, the **/etc/init.d/lp** shell script is linked to the **/etc/dinit.d/S80lp** and **/etc/rc0.d/K20lp** files. When run with the **start** option the **/etc/init.d/lp** script executes **/usr/lib/lpsched** to start the printing scheduler. When run with the **stop** option it executes **/usr/lib/lpshut** to kill the printing scheduler.

When you run **init 2, init** runs the scripts in **/etc/dinit.d,** one of which is **S80lp.** The script **S80lp** is executed with the **start** option, as follows:

```
sh S80lp start
```

Similarly, when you run **init 0, /etc/rc0.d/K20lp** is executed with the **stop** option:

```
sh K20lp stop
```

Running either of these scripts is the same as running **/etc/init.d/lp** with the appropriate **start/stop** option.

There are files under **/etc/init.d** that are not linked to any files under the **/etc/rc***num*.d directories, where *num* is an integer identifying the system state to which the directory applies. These files are left in the **/etc/init.d** directory for the convenience of the administrator.

Because these files are shell scripts, you can read them to see what they do. You can also change the files, although it is preferable to create your own versions because the delivered scripts may change in future releases.

Follow these rules when creating your own scripts:

- Link the script [see **`ln(1)`**] to files in appropriate system state directories, using the naming convention described above.
- Place the file containing your script in the **`/etc/init.d`** directory if it is a script that must be run before a user should log in.
- If a start script may run after a user logs in, place it in the **`/etc/dinit.d`** directory. This is a performance consideration in that, the fewer processes that have to run before a login prompt is displayed, the quicker a user may be able to login.

Shutting Down Your System

An Overview of Booting and System States

Many administrative tasks require the system to be shut down to a system state other than multi-user state. (See *"System States"* earlier in this chapter for further information.) Users, however, cannot access the system if it is not in multi-user state. Follow these guidelines to avoid inconveniencing other users:

- Schedule tasks that affect service for periods of low system use. Inform users of the times the system will be unavailable.
- Check to see who is logged in before taking any actions that would affect active users. The **`whodo(1M)`** and **`who(1)`** commands list users on the system.
- Warn users as early as possible about changes in system states or pending maintenance actions. Try to give them a reasonable amount of time to stop working and log off before you take the system down. When you can't give much advance notice, try to tell users when the system will be available again.

  If you must interrupt service unexpectedly, broadcast a warning to all users' screens with the **`wall(1M)`** command. See Chapter 1, "Introduction to Basic Administration" of this book for details about sending messages to users.
- Use the news [see **`news(1)`**] and the message of the day (contained in **`/etc/motd)`** to inform users about changes in hardware, software, policies, and procedures.
- Keep a record of administrative activities in a system log notebook. This log can prove invaluable in recovering your system should you make a serious error.

Execute the **shutdown** command to shut down your computer. Several options are available for use with **shutdown**, as shown in the following example.

```
shutdown -i0 -g0 -y
```

where the arguments are defined as:

-i0 Change the system to state 0 (off).

This is the default. If the **shutdown** command is issued without a **-i** argument, the system will change to state 0.

-g0 Allow a grace period of 0 seconds (for users to log off).

The grace period is a tunable parameter found in the **/etc/default/shutdown** file. If the Unlimited User Upgrade package is installed, the default grace period is 60 seconds. If this package is not installed, the default is 0 seconds, and the **-g0** argument is not necessary. You may specify any number of seconds you choose in the **shutdown** file, or with the **-g** option.

-y Answer `yes` to questions asked by **shutdown**.

Specifying a **-y** argument causes the `Do you want to continue (y or n)` prompt to be automatically answered with a `y`.

See the **shutdown(1M)** manual page for detailed information.

The **shutdown** command acts differently depending upon the number of users on the system. If there is more than one user on the system, **shutdown** broadcasts a message that the system is coming down, and waits for a 60-second grace period (or for the grace period specified in the **shutdown** file, or by the **-g** option) before prompting whether to continue with the shutdown or not. If you are the only user on the system, no message is broadcast, and the grace period is set to 0 seconds. In either case, **shutdown** flushes the system buffers, closes any open files, stops all user processes and daemons currently running, and unmounts file systems.

Kernel Rebuilding

If **idbuild** is invoked without any options, it sets a flag and exits without actually rebuilding the kernel. The system later checks the flag and invokes **idbuild -B** to rebuild the kernel automatically if the flag is set.You will see the following prompt before the system kicks off the kernel rebuild:

```
Confirm


The Operating System kernel will now be rebuilt to
incorporate recent configuration changes.


Strike ENTER when ready
or ESC to stop.
```

If you press the ENTER key, the kernel is rebuilt, the new kernel is installed as **/stand/unix,** and the original kernel is saved as **/stand/unix.old.** If an error occurs during the kernel rebuild, an error message will be mailed to **root** and also displayed on the console.

However, if you press the ESC key, the rebuild will be aborted with the following message:

```
The rebuild of the Operating System kernel has been
aborted.
A rebuild will be attempted at next reboot.
```

If you do not respond to this prompt within 30 seconds, **idrebuild** proceeds as if you had pressed the ENTER key.

You can enforce the confirmation in **idrebuild** at every rebuild attempt by using **defadm** to assign the value NO to the variable AUTOREBLD as follows:

```
defadm idtools AUTOREBLD=NO
```

The 30 second timeout described above is disabled when confirmation is enforced in this manner.

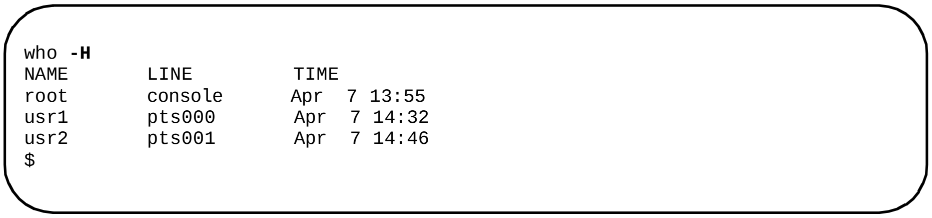
After a new kernel is built, the system will be shut down again, and then automatically rebooted with the new kernel.

New Kernel Installation

As previously stated, if the new kernel is built successfully, the system saves the original kernel in **/stand/unix.old** if it is possible. Then, it copies the newly built kernel to **/stand/unix.** If the copy fails, the system will try the copy again after removing the saved old kernel.

shutdown Example - Multiple Users

1. Though not required, it is useful to determine if there are users on the system by executing a **who -H** command, as in the following example.



```
who -H
NAME       LINE         TIME
root       console      Apr  7 13:55
usr1       pts000       Apr  7 14:32
usr2       pts001       Apr  7 14:46
$
```

2. Type **shutdown -i0** or **shutdown**
3. If more than one user is logged on, the system automatically notifies users that it's about to be shut down, as shown in the following example.

```
Broadcast Message from root (console) on unix Wed Feb 19 07:30:27 . . .
The system will be coming down in 60 seconds.
Please log off.
$
```

4. After a 60-second grace period, the following message is displayed.

```
Shutdown started.    Thu May 15 17:10:57 EDT 1992

Broadcast Message from root (console) Thu May 15 17:10:59
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.
Do you want to continue (y or n):
```

5. Respond by typing `y` to continue the shutdown, or `n` to cancel it. Following a `y` response there is another 60-second grace period. Then messages are displayed, and the system shuts down.

```
Do you want to continue (y or n): y <RETURN>

INIT: New run level: 0
The system is coming down. Please wait.

CPU 0 trap511 halt executed
00168D08 [00168D08] halting % F460C001 jmp r1
#>
```

After these messages are received, you can safely turn off the power to your computer.

shutdown Example - One User

If you are the only user on the system and you enter **`shutdown -y`**, console messages such as the following will appear.

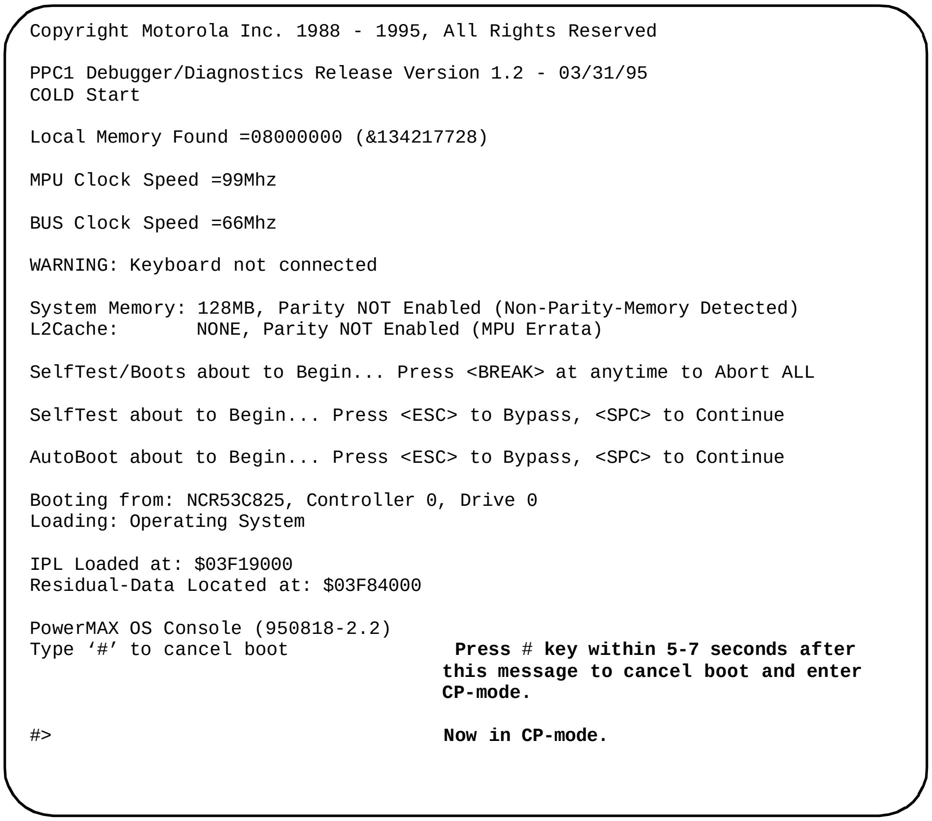
```
Shutdown started.    Mon Jul  7 12:17:57 EDT 1992

INIT: New run level: 0
The system is coming down.  Please wait.

CPU 0 trap511 halt executed
00168D08 [00168D08] halting % F460C001 jmp r1
#>
```

After these messages are received, you can safely turn off the power to your computer.

System Specific For Booting And Taking Down The System

How to Enter CP-Mode (Power Hawk Series 600 Systems Only)

The examples in Screen 3-6 show how to bring the Power Hawk Series 600 system to the CP-mode during boot-up. This sequence is begun on power up, or by pressing the RESET switch on front of the CPU. If the boot process is not interrupted, the system will enter the multi-user state.

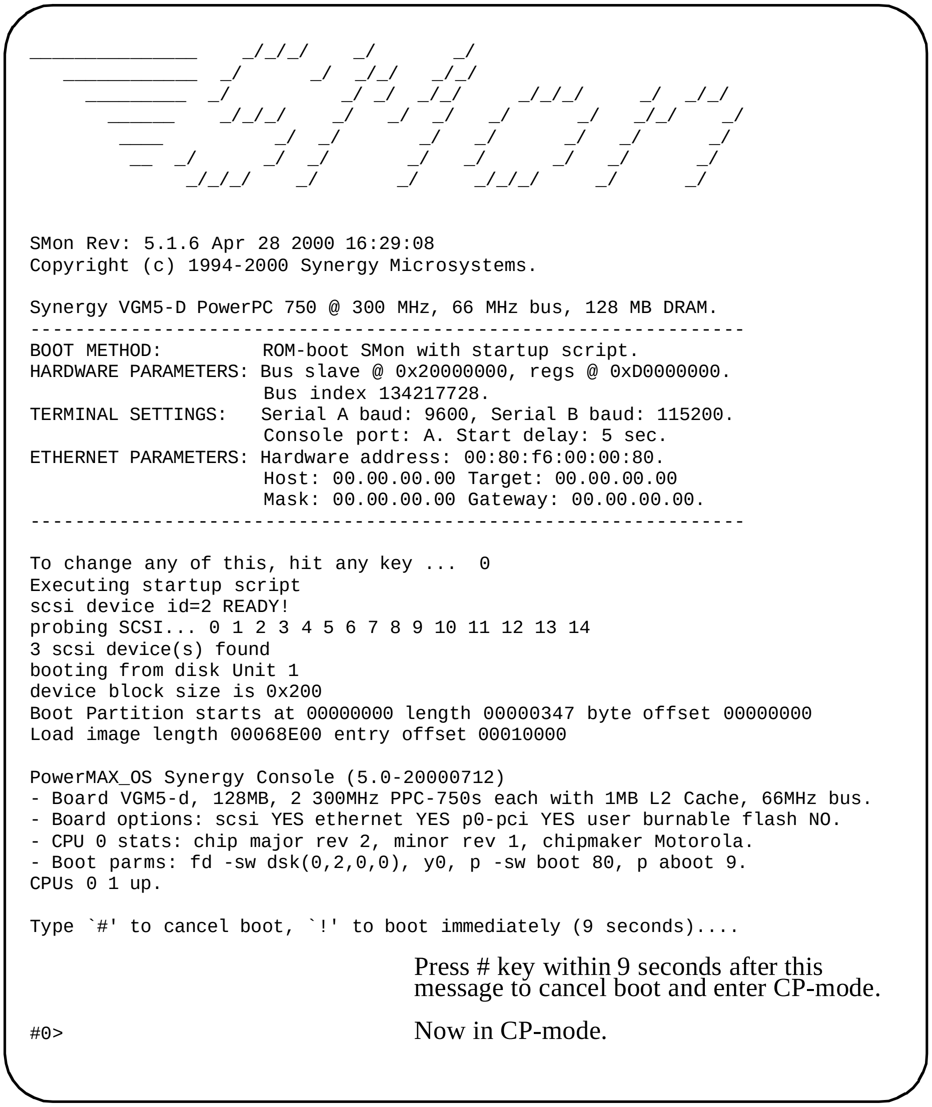


```
Copyright Motorola Inc. 1988 - 1995, All Rights Reserved

PPC1 Debugger/Diagnostics Release Version 1.2 - 03/31/95
COLD Start

Local Memory Found =08000000 (&134217728)

MPU Clock Speed =99Mhz

BUS Clock Speed =66Mhz

WARNING: Keyboard not connected

System Memory: 128MB, Parity NOT Enabled (Non-Parity-Memory Detected)
L2Cache:       NONE, Parity NOT Enabled (MPU Errata)

SelfTest/Boots about to Begin... Press <BREAK> at anytime to Abort ALL

SelfTest about to Begin... Press <ESC> to Bypass, <SPC> to Continue

AutoBoot about to Begin... Press <ESC> to Bypass, <SPC> to Continue

Booting from: NCR53C825, Controller 0, Drive 0
Loading: Operating System

IPL Loaded at: $03F19000
Residual-Data Located at: $03F84000

PowerMAX OS Console (950818-2.2)
Type ‘#’ to cancel boot               Press # key within 5-7 seconds after
                                     this message to cancel boot and enter
                                     CP-mode.

#>                                   Now in CP-mode.
```

Screen 3-6. Entering CP-Mode on Power Hawk Series 600 Systems

How to Enter CP-Mode (Power Hawk Series 700 Systems Only)

The examples in Screen 3-7 show how to bring the Power Hawk Series 700 system to the CP-mode during boot-up. This sequence is begun on power up, or by pressing the RESET switch on front of the CPU. If the boot process is not interrupted, the system will enter the multi-user state.



```
_______________    _/_/_/    _/       _/
   ____________  _/      _/  _/_/   _/_/
     _________  _/          _/ _/  _/_/     _/_/_/     _/  _/_/
       ______    _/_/_/    _/   _/  _/   _/      _/   _/_/    _/
        ____          _/  _/       _/   _/      _/   _/      _/
         __  _/      _/  _/       _/   _/      _/   _/      _/
              _/_/_/    _/       _/     _/_/_/     _/      _/


SMon Rev: 5.1.6 Apr 28 2000 16:29:08
Copyright (c) 1994-2000 Synergy Microsystems.

Synergy VGM5-D PowerPC 750 @ 300 MHz, 66 MHz bus, 128 MB DRAM.
----------------------------------------------------------------
BOOT METHOD:         ROM-boot SMon with startup script.
HARDWARE PARAMETERS: Bus slave @ 0x20000000, regs @ 0xD0000000.
                     Bus index 134217728.
TERMINAL SETTINGS:   Serial A baud: 9600, Serial B baud: 115200.
                     Console port: A. Start delay: 5 sec.
ETHERNET PARAMETERS: Hardware address: 00:80:f6:00:00:80.
                     Host: 00.00.00.00 Target: 00.00.00.00
                     Mask: 00.00.00.00 Gateway: 00.00.00.00.
----------------------------------------------------------------

To change any of this, hit any key ...  0
Executing startup script
scsi device id=2 READY!
probing SCSI... 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
3 scsi device(s) found
booting from disk Unit 1
device block size is 0x200
Boot Partition starts at 00000000 length 00000347 byte offset 00000000
Load image length 00068E00 entry offset 00010000

PowerMAX_OS Synergy Console (5.0-20000712)
- Board VGM5-d, 128MB, 2 300MHz PPC-750s each with 1MB L2 Cache, 66MHz bus.
- Board options: scsi YES ethernet YES p0-pci YES user burnable flash NO.
- CPU 0 stats: chip major rev 2, minor rev 1, chipmaker Motorola.
- Boot parms: fd -sw dsk(0,2,0,0), y0, p -sw boot 80, p aboot 9.
CPUs 0 1 up.

Type `#' to cancel boot, `!' to boot immediately (9 seconds)....


#0>
```

Press # key within 9 seconds after this message to cancel boot and enter CP-mode.

Now in CP-mode.

Screen 3-7. Entering CP-Mode on Power Hawk Series 700 Systems

Part I - Booting The SystemTo Single-User Mode

The examples in Screen 3-8 show how to bring the system from CP-mode to single-user mode. For more information refer to online manual page **boot(8)**.

```
 #> pboot 2. 00000000     Boot option to enter single-user mode.
                          (See boot(8) for valid values).
 #> fb                    fb requests the system boot single-user.

  .
  .
 Boot
 :/stand/unix
  .
  .
  .
  Booting messages...
  .
  .
  .
INIT: New run level: S
INIT: SINGLE USER MODE

Type <CTRL><d> to proceed with normal startup,
(or give root password for single-user mode
```

Screen 3-8. Sample Boot to Single-User

Note that single-user boot option overrides the default system state as specified by the init-default line in the file **/etc/inittab.** When this bit is set, the system always stops in single-user mode.

Part II - Booting a Different Kernel

If you want to boot a different kernel other than **/stand/unix,** a sample sequence to accomplish this is shown in Scree n3-9.

```
 # pboot 3. 00000000 Request the system to prompt for the file
                     to boot plus enter single-user mode.
 #> fb               Requests the boot.
  .
  .
  Booting messages...
  .
  .
  .
  .
Boot                 The system prompts for boot
                     information.

: /stand/unix.good    Boot different kernel.
```

Screen 3-9. Booting a Different Kernel

Part III - Booting From a Different Root Disk

Suppose you want to specify a boot file other than dsk(0,0,0) /stand/unix (assuming that the default boot device is dsk(0,0,0). A sample sequence to accomplish this is shown in Screen 3-10.

```
 # pboot 3. 00000000         Request the system to prompt for the file
                             to boot plus enter single-user mode.
 #> fb                       Requests the boot.
  .
  .
  Booting messages...
  .
  .
  .
  .
Boot                         The system prompts for boot
                             information.

: dsk(5,0,0,0)/stand/unix    Reply with the device name
                             which is in the form:
                                xxx(c,u,p,b)program.
                                xxx is dsk, or mt. c is a controller
                             or slot number (hex) for HVME
                             controllers. u is a unit or
                             drive number. p is a partition
                             (0-7) for disks, and a file number
                             on the tape for tapes. b is the bus
                             number. program is the name of
                             a standalone executable file on the
                             device specified.”
                                controller 5
                                drive 0
                                partition 0
                                bus 0
                             (Note: device specification not needed
                             if same disk partition.)
```

Screen 3-10. Booting From a Different Root Disk

Alternate Method to Change Default Boot Device

An alternate method to the previous example is to change the default device using the console **`fd`** command. (Refer to the appropriate *Console Reference Manual* for details on this command.) A sample sequence to accomplish this is shown in Screen 3-11.

```
#> pboot 2. 00000000          Boot option to enter single-user mode.
                              (See boot(8) for valid values).
#> fd dsk(5)                  Defines default boot device ( see Screen3-10).
#> fb                         Request the system boot single-user.
 .
 .
Boot
:/stand/unix                  The system prompts for boot
```

Screen 3-11. Changing the Default Boot Device

Part IV - Taking The System From Single-User To Multi-User Mode

Methods To Bring The System To Multi-User Mode

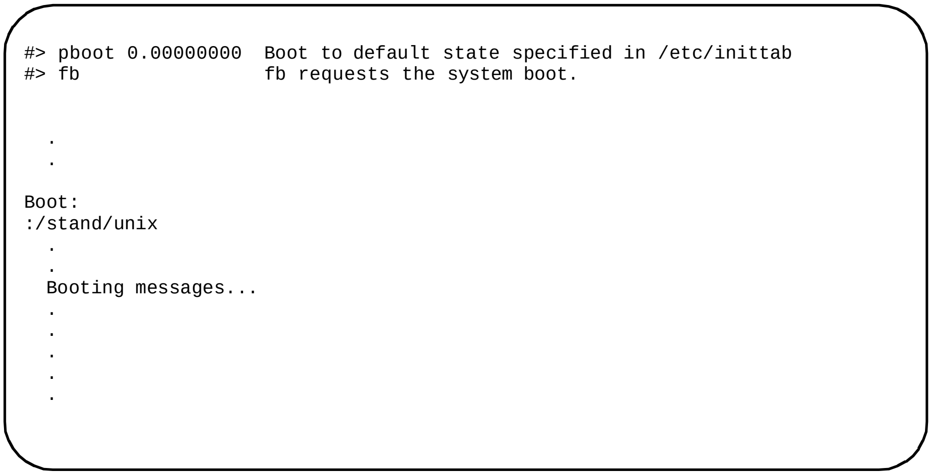
The various methods of bringing the system from single- to multi-user mode are summarized in Table 3-2.

Table 3-2. Boot Methods from Single-User to Multi-User Mode

Method	Recommended Use
Type telinit[2] or init[2]	From single-user mode, file switches to multi-user mode. Refer to section “Changing to Multi-User State”.
Press <CTRL><d> Type 2 to init’s prompt for a run level.	Same as above.
Create a null fastboot file in the root file system. Then execute one of the above.	Same as above, except file system checks are skipped. CAUTION: only do this if the file systems are in a consistent state.

Part V - Booting Direct to Multi-User Mode

The normal boot procedure is shown in Scree n3-12.



```
#> pboot 0.00000000  Boot to default state specified in /etc/inittab
#> fb                fb requests the system boot.


   .
   .

Boot:
:/stand/unix
   .
   .
   Booting messages...
   .
   .
   .
   .
   .
```

Screen 3-12. Boot to Multi-User Mode)

This boots to the default state as specified by the initdefault line in the file **`/etc/inittab`** (usually state 2 or 3).

Part VI - Shutting Down The System.

Try to schedule system downtime in advance and warn users that you plan to shut down the system. There are two methods of issuing the warning.

- Place a warning in the message-of-the-day (/etc/motd) file, which users see when they login, and
- Use the wall(1M) command to issue periodic warnings to the users.

The six methods of shutting down the system are:

- /sbin/shutdown
- /etc/halt
- /etc/reboot
- /etc/telinit s
- ~i
- /usr/ucb/fasthalt

Shutdown Method

Although the six methods of shutting down the system are described in the following paragraphs, consider the various side effects of each method to determine the best method for a given situation.

/sbin/shutdown

This method is described under section “Shutting Down Your System”.

/etc/halt

The /etc/halt command synchronizes the super blocks, kills all processes, and halts the CPU. /etc/halt does not warn users that you are taking the system down.

/etc/reboot

Similar to /etc/halt but automatically reboots.

/etc/telinit s

This command kills all processes and brings the system to single-user mode. It does not warn users, synchronize the super blocks, or unmount file systems.

<CR><~><i>

Typing <CR><~><i> from multi- or single-user mode causes the system to go to CP-mode. The <CR><~><i> command does not warn users that the system is going down. It does not synchronize the super block, or unmount file systems, it simply halts all CPUs.

This also occurs if the console wakeup switch on the console is depressed. This method is not recommended unless all other methods have failed.

Using /usr/ucb/fasthalt & /usr/ucb/fastboot

These commands are simple shell scripts that create the file /fastboot before executing halt or reboot, respectively. Use these commands only if you do not want to perform fsck checks automatically when the system boots up and you are sure the filesystems are in good shape.

How to Decrease Boot Time

As an administrator, there are several things you can do to decrease boot time. One method involves changing certain tunables values (explained below). The second method involves customizing your system. Refer to the section entitled “Customizing Your System”, immediately following the section on changing tunables, for a detailed description on how to customize your system to achieve reductions in boot time.

Tunables that Affect Boot Time

Boot time can be decreased by bypassing certain functions during system initialization. The functions that can be bypassed are listed in Tabl e3-3. The tunables that control whether these functions are performed are listed adjacent to the function(s) they control. Following the table is a description of each tunable.

Table 3-3. Functions That May Be Bypassed To Decrease Boot Time

Tunable	Function
CONSMSGDIS	console message output during system initialization
FASTFS	file system consistency checks performed by initialization scripts creation of security index and master files remounting of root file system
SCSISCAN	SCSI device lookup

The three tunables, CONSMSGDIS, FASTFS and SCSISCAN are described below. The tunable values can be changed as necessary by using the kernel configuration utility **`(config(1M)).`** Refer to Chapter 8 “Configuring and Building the Kernel” in Volume 2 for complete information on this utility. Note that the kernel must be rebuilt and rebooted before the modification takes effect. The config utility may also be used to rebuild the kernel.

CONSMSGDIS

This tunable controls whether console messages are output during system initialization. This tunable can take on one of three values:

`0` - console message output is enabled during system initialization.
`1` - console message output is disabled during system initialization.

`2` - console message output is determined by the presence of a console. When a console is present, console messages will be output. When a console is not present, console messages will not be generated. (**Note:** This feature is currently only available on the Power Hawk 610. It will not be implemented on the HN6200, HN6800 and PowerMAXION systems until the release of console firmware Revision 908 which is expected to be available in the fall of 1996.)

The default value is `0`, that is, console messages are always generated.

FASTFS

Several functions can be bypassed during system initialization if the file systems are guaranteed to be clean. This tunable controls whether the following functions are bypassed:

1. file system consistency checks performed by system initialization scripts -

When the system is booted, several scripts are run that check all file systems for consistency and perform repairs as needed. If the file systems are guaranteed to be in a clean state this process can be bypassed to reduce boot time.

2. creation of security index and master files -

The master and index files are authentication files which are created during each system boot by **`creatiadb(1M)`**. These files don't need to be created on every reboot if the file systems are guaranteed to be clean therefore reducing boot time.

3. remounting of root file system -

The **`root`** file system is initially mounted read-only by the kernel. The **`ckroot`** script executed from **`inittab`** takes care of remounting it read/write after it checks it for inconsistencies. If the **`root`** file system is guaranteed to be clean at boot time then it can be mounted read/write on the first mount. File system check and remount of **`/root`** done by **`ckroot`** can then be bypassed to reduce boot time.

The FASTFS tunable has one of two values:

`0` - file system checks are not bypassed on system boot.

`1` - file system checks are bypassed on system boot when the file systems are guaranteed to be clean.

The default value is 0, that is, file system checks are not bypassed on system boot. (See Note below.)

Note

The system needs to be booted once after initial system configuration for the master and index files to be created. The kernel can then be reconfigured with the FASTFS tunable set to 1 to bypass file system checks. If the kernel is a secure kernel, the master and index files will always be created whether this tunable is set or not.

SCSISCAN

The generic disk driver scans each adapter for SCSI devices which may be attached to it. Each device found is added to the device list for the adapter. This list is later used by: **`hwstat(1M)`**, a hardware configuration report generator, **`devcfg(1M)`** a configuration utility, and **`idmknod(1M)`** which updates device nodes to reflect kernel configuration and is run during system reboot by **`idmkenv`**.

This above step could be bypassed to reduce boot time in systems where: the configuration will not change, the kernel will not be rebuilt and where users will have no need to run **`hwstat(1M)`** to find out what disk drives are configured on the system.

This tunable can have one of two values:

`0` - don’t perform scan for SCSI devices; that is, device table for adapters will not be built.

`1` - perform scan for SCSI devices building the device table for adapters.

The default value is 1; that is, the scan will be performed.

Customizing Your System

As an administrator, there are several things you can do to further reduce boot time. What you do depends on how your system is being used, whether you have a defined hardware configuration that will not change, and whether you will need to reconfigure the kernel. In general, some of the things you can do are:

1. Create your own system state which bypasses some of the functions performed during the transition from system boot to multi-user state.
2. Make the **`/etc/inittab`** file smaller by eliminating entries which are not required to run for your state or, can run after the system has been booted.

3. Delay starting some utilities such as networking for example, until after the system has booted.
4. Forcing the kernel to be allocated contiguously on disk.
5. Use of a static kernel over a dynamic kernel. Refer to the chapter "Managing System Performance" in Volume 2, *System Administration*. Specifically, refer to the section entitled "Dynamic vs Static Kernels" for more information on this subject.

The following sections give a detailed description on how to make these changes.

System State 4

System state 4 is provided as a convenience to enable you to customize a system state. For example, you may want to come up to a state that has all the file systems mounted but the network is not initiated. Refer to the section on "System State Directories" presented earlier in this Chapter for an explanation on system state directories and the scripts located in these directories.

An **rc4** script and **rc4.d** directory are not delivered with the system, you will need to create these yourself. You will then include in the **rc4.d** directory only those scripts which must run when changing to state 4. The **rc4** script will be executed through **inittab** to invoke the scripts in the **rc4.d** directory.

To create your own **rc4** script, do the following:

1. Using one of the existing rc scripts as a model for example, **rc2**, copy the state script to a new file named **rc4**.
2. Edit **rc4** to invoke the scripts under **rc4.d** and modify as needed to perform functions required for your particular system.

There may be a utility or daemon typically started during system initialization through one of the initialization scripts which you may not need until after the system is up and running. In this case, you can start the particular function at some other convenient time by invoking the script (possibly through an application). You would not include this script in your **rc** directory.

For example, if you wish to delay initiation of the error daemon you may start it later by invoking the following:

```
/sbin/sh /etc/init.d/errdemon start
```

To initiate the networking utilities invoke the following:

```
/sbin/sh /etc/init.d/inetinit start
```

The following modifications need to be made to the **inittab** file to reflect the new run level:

1. The default run level entry must reflect a run level of "4":

   ```
   is:4:initdefault:
   ```

2. The following line needs to be inserted to specify a script to run specifically for run level 4:

```
r4:4:wait:/sbin/rc4  >/dev/sysmsg 2>&1 </dev/console
```

Inittab Entries Which May be Deleted

There are entries in **inittab** which may be removed depending on how your system is set up and the system’s function. These entries fall into the following categories:

1. Functions invoked through **inittab** that are only necessary when booting a development system after reconfiguring a kernel. These functions may not be necessary when booting your final production system assuming you have a set hardware configuration and the kernel does not have to be reconfigured.
2. Functions that can be delayed to start only after the system has been initialized, perhaps through a user application.
3. Functions applicable only to a secure operating system.
4. Entries that are only executed for a specified run level. If the system will never be changed to that level, the appropriate entries may be deleted to make the **inittab** file smaller therefore decreasing the time it takes **init** to scan this file.

The executable **/etc/conf/bin/idmkenv**, reconstructs **/etc/inittab** from files in **/etc/conf/init.d** whenever the system is rebooted after a new kernel has been reconfigured. Once the system has been reconfigured to your needs, a copy of **/etc/inittab** can be saved and the original **inittab** file edited to remove or modify entries as specified below. The original **inittab** file can then be restored if this action becomes necessary.

The following entries for security scripts may be deleted for a nonsecure kernel:

```
mi::sysinit:/sbin/sh -c ‘[ -x /sbin/macinit ] && /sbin/macinit’ > \
/dev/sysmsg 2>&1
```

```
xdc::sysinit:/sbin/sh -c ‘if [ -x /etc/rc.d/es_setup ] ; then \   /
etc/rc.d/es_setup ; fi’ >/dev/console 2>&1
```

The following entries for reconfiguring and rebuilding the kernel may be deleted. **(Note:** If the kernel is to be reconfigured, the following first two entries need to be restored in the **inittab** file before rebooting the system.)

```
me::sysinit:/etc/conf/bin/idmkenv >/dev/sysmsg 2>&1
```

```
bu::sysinit:/etc/conf/bin/idrebuild reboot </dev/console> \
/dev/sysmsg 2>&1
```

```
bd:56:wait:/etc/conf/bin/idrebuild </dev/console >/dev/sysmsg 2>&1
```

The following entries for registering dynamically loadable modules (DLMs) may be deleted if the kernel booted has been linked only with statically linked modules:

```
mm::sysinit:/etc/conf/bin/idmodreg -c \`/etc/conf/bin/idkname /
-c\` >/dev/sysmsg 2>&1
```

```
ldmd::sysinit:/etc/conf/bin/idmodload >/dev/sysmsg 2>&1
```

Any of the following entries for level 1-3 may be deleted if you don’t plan on changing the system to these levels:

```
    r1:1:wait:/sbin/rc1 >/dev/sysmsg 2>&1 </dev/console

    r2:23:wait:/sbin/rc2 >/dev/sysmsg 2>&1 </dev/console

    r3:3:wait:/sbin/rc3  >/dev/sysmsg 2>&1 </dev/console
```

The following entries for level 0, 5 and 6, which are used to bring the system down, may be deleted if you don’t plan on bringing the system down by invoking **init** or **shutdown**. See the section entitled, “Shutting Down the System Quickly” presented later in this Chapter, for procedures on how to shut down the system quickly.

```
    r0:0:wait:/sbin/rc0 off >/dev/sysmsg 2>&1 </dev/console

    r5:5:wait:/sbin/rc0 firm >/dev/sysmsg 2>&1 </dev/console

    r6:6:wait:/sbin/rc0 reboot >/dev/sysmsg 2>&1 </dev/console

    sd:0:wait:/sbin/uadmin 2 0 >/dev/sysmsg 2>&1 </dev/console

    fw:5:wait:/sbin/uadmin 2 2 >/dev/sysmsg 2>&1 </dev/console

    rb:6:wait:/sbin/uadmin 2 1 >/dev/sysmsg 2>&1 </dev/console
```

If the system does not have a console, or you can delay initiation of the console login, you may delete the following line which starts the console login:

```
co:12345:respawn:/usr/lib/saf/ttymon -g -v -p /
\\”Console Login:\\ ” -d /dev/console -l console
```

You can initiate the console login at a later time by invoking the following command:

```
/usr/lib/saf/ttymon -g -v -p \\”Console Login: \\” -d /dev/console /
-l console
```

The following line is not required since these devices are created as part of the system initialization when a system is rebooted after a reconfiguration:

```
     li:23:wait:/usr/bin/ln /dev/systty /dev/syscon >/dev/null 2>&1
```

If the system does not need the network to be up, or delaying initiation of the network until after the system is up is acceptable, the following entry which starts the service access controller (SAC), which in turn is responsible for starting the port monitor, may be deleted.

```
    sc:234:respawn:/usr/lib/saf/sac -t 300
```

To initiate the SAC at a later time invoke the following command in the background:

```
    /usr/lib/saf/sac -t 300 &
```

dinit

The **/sbin/dinit** script completes initialization of processes that can be delayed until after a login prompt is displayed. The directory **/etc/dinit.d** contains the initialization scripts for each of these processes. These include the following:

1. Starting the **cron** daemon.
2. Making the line printer service available, if installed.
3. Making **uucp** available for use, if installed.
4. Starting connection server for TLI and serial connections.
5. Creating **ttymap** file for use by **ttyname** library function.

You may not need to run all of these scripts on your system. Delete scripts from this directory for functions that are not required for your system. If you want **dinit** to be run for level 4, you must modify the **dinit** entry in **inittab** to execute at level 4:

original entry:

```
d2:23:wait:/sbin/dinit >/dev/sysmsg 2>&1 </dev/console
```

modified entry:

```
d2:234:wait:/sbin/dinit >/dev/sysmsg 2>&1 </dev/console
```

idrc.d

The **/etc/idrc.d** directory contains **rc** scripts from driver packages which are installed on the system. They would include, for example, **hps**, **rtc** and **pts**. The scripts in this directory are executed directly from **/sbin/rc2** when coming up to multi-user level. You may decide to delay execution of some files in this directory or bypass their execution completely. For example:

- The **pts** script sets up pseudo-terminals and needs to be executed only once when the system is initially booted after configuration. Therefore, it can be eliminated from the system initialization procedure for the production system if the configuration will not change.
- The **rtc** script sets up a real-time clock as a high resolution timer expiration source. This script needs to be executed only if a real-time clock will be used as a high resolution timer as is the case for POSIX timeout and nanosleep services. You may delay starting this script until after your system is up.

If you are customizing your state by using level 4 and you want to run the scripts in this directory, you must include code in the **rc4** script to do so.

Initprivs

The **initprivs** command initializes the system with privilege information. It reads this information from **/etc/security/tcb/privs**. During system initialization,

initprivs is always run after the **/usr** file system is mounted. This command is always executed on system initialization because the system's privilege information is always cleared when a file system is unmounted.

The following privileges are required on a non-secure kernel:

- P_CPUBIAS
- P_LOADMOD
- P_MOUNT
- P_OWNER
- P_PLOCK
- P_RTIME
- P_SHMBIND
- P_SYSOPS
- P_TSHAR
- P_USERINT

In systems not running a secure kernel this operation could be expedited by removing all entries in **/etc/security/tcb/privs**, except those used by applications which have any of the privileges listed above. It is recommended that you save a copy of the original file before you make any changes.

If an application invokes any service or library routine which requires any of the above listed privileges, then an entry will need to be added for the application in the **/etc/security/tcb/privs** file with the appropriate privileges. This can be accomplished by invoking the following line when the system is being configured (the argument **priv** is defined as a process privilege name):

```
filepriv -f priv,priv,... file
```

Refer to **filepriv(1M)** and **initprivs(1M)** for a more detailed description on file privileges.

How to Shut Down the System Quickly

Shut down time is affected by: how may processes are running, how much of the file buffers need to be flushed to disk, and the number of facilities and daemons that are running that need to be stopped in a clean and orderly fashion.

In your particular environment, the method of choice for bringing the system down may be to invoke the **uadmin(1M)** command directly. Bypassing **init** and **shutdown** speeds up the process by a significant amount of time. Although **uadmin** will flush the file buffers to disk, it is recommended that you either flush the file system buffers to disk periodically by issuing a **sync(2)**call after writes to files, or use direct I/O in which cases **syncs'** are not required. Doing this periodically will expedite the **uadmin** operation during shut down. When you are ready to bring the system down you will need to perform the following steps:

1. First, stop utilities and daemons that require to be stopped in a clean fashion. For example, to stop the error daemon, invoke the error daemon script with a stop argument:

   ```
   /sbin/sh /etc/init.d/errdemon stop
   ```

2. Invoke the **uadmin** command as follows:

```
uadmin 2 0
```

What to Do If the System Does Not Boot

There are several scenarios in which user modifications to the system (intentional or unintentional) may result in a system that won't boot.

- Hardware failure.
- Modification to the system kernel, such as adjustment to a tunable parameter or installation/removal of system software, such as a device driver.
- Accidental removal of a file or system utility needed by the system during the boot phase.

After the system kernel is modified, either as a result of software installation/removal or adjustment to a tunable parameter, a copy of the old kernel is automatically saved in the file **/stand/unix.old.** To recover from a new, unbootable kernel, you should first try to reboot the old operating system kernel.

Some administrators like to keep a copy of the kernel that they know to be bootable under another name, **unix.good,** for example. If you provide the name of that kernel, the boot code will load that kernel.

Refer to section “Part II - Booting a Different Kernel”, for procedure on booting a different kernel than the default kernel.

Displaying Information About the System

Displaying Summary Configuration Information

To print system configuration information, type:

```
hwstat
```

The system will display information about memory and peripheral configuration, such as the information shown in the following example:

```
Boot Time: Wed Apr 27 13:03:24 1994
Boot Data (vers 2): Device: dsk(4,0,0,0) Console Local Port Baud Rate: 9600

2 CPUs:
Logical  Slot  Physical  Flags
=======  ====  ========  ====================
      0     0      0x00  BOOT
      1     0      0x01  (none)

1 Memory Pools:
Id  Type        Size(Kb)  Free(Kb)  Start Addr  End Addr    Logical CPUs
==  ==========  ========  ========  ==========  ==========  ===============
 0  Local          32752     14484  0x00000000  0x01ffbfff  0 1

I/O Configuration:  Bus0: HVME  Bus1: (none)  Bus2: (none)  Bus3: (none).
7 Devices Configured:
Device                 Major/Minor     Bus   Bus I/O Addr  Std I/O Addr
=====================  ==============  ====  ============  ============
PG    0  FDDI          (    7, 13   )  Bus0    0xFFFF0200       --
EGL   0  Ethernet      (    7, 46   )  Bus0    0xFFFF0800    0xF2000000
HPS   0  Serial Ports  (   10, 12288)  Bus0    0xE0140000    0x00000000
HSA   0  Disk     0/0  (  101, 0    )  Bus0    0xC4040000    0x00000000
HSA   0  Disk     1/0  (  101, 256  )  Bus0    0xC4040000    0x00000000
HSA   0  Disk     2/0  (  101, 512  )  Bus0    0xC4040000    0x00000000
HSA   0  Tape     5/0  (  103, 1280 )  Bus0    0xC4040000    0x00000000
#
```

Displaying System Name and Operating System Release Number

To display your system name and the number of your operating system release, use the **uname** command with the **-s** and **-r** options, as follows:

```
$ uname -sr
Operating_System X.X  (where X.X = OS Release)
```

In this example, the name of the system is Operating System and the release number of the operating system being run is Version X.X. See **uname(1)** for a list of other information you can display and change with the **uname** command.

Displaying a List of Users Logged On to Your Machine

Before taking any action that would affect a system user, check to see who is logged on to the system. The **who** command displays a list of users logged on to your computer, along with the ID, terminal number, and login time of each user. The **-H** option supplies headers for each field. For example

:

```
$ who -H
NAME       LINE         TIME
rrusk      pts/0        Apr 15 17:01
joei       pts/1        Apr 15 17:52
allen      pts/3        Apr 15 13:47
$
```

The **`who`** command has a number of options that allow you access to more information than the previous example shows. The following list describes some of these options. For a complete listing and details about each option, see **`who(1)`**.

`-u` On each line, show the number of hours and minutes since activity last occurred. A dot (.) indicates the terminal has been active in the last minute and is therefore "current." The information is included as an additional field on the default display for the **`who`** command.

`-T` Show whether someone else can write to that terminal. A plus sign (+) appears if the terminal is writable by other users; a minus sign (`-`) appears if it is not. (A privileged process can write to all terminal lines.) If a bad line is encountered, a `?` is printed.

`-l` Show lines on which the system is waiting for someone to log on.

`-q` Show a "quick" listing, containing only the login name for those who are logged on to the system and the total number of users logged on.

`-b` Show the current system state of the init process and the date.

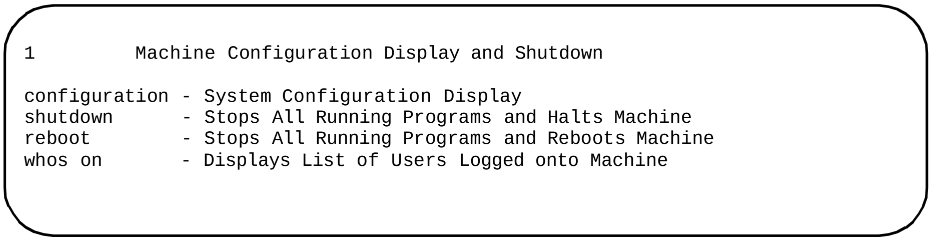
`-a` Run **`who`** with all options turned on.

Displaying Processor Information

To display the processor ID number or whether the processor is online or off line, use the **`psrinfo(1M)`** command. When you use the command without specifying any processors, information about all processors will be written to standard output.

Machine Management through OA&M Menus

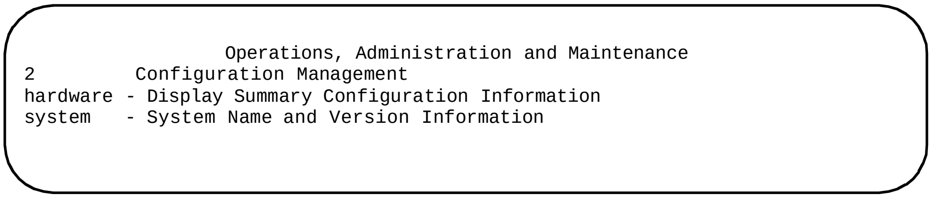
The system administration menus are only available if the Operations, Administration and Maintenance (OA&M) package is installed on your system. To access the menu for computer management type **`sysadm machine`**. The following menu will appear on your screen.



```
1         Machine Configuration Display and Shutdown

configuration - System Configuration Display
shutdown      - Stops All Running Programs and Halts Machine
reboot        - Stops All Running Programs and Reboots Machine
whos on       - Displays List of Users Logged onto Machine
```

Screen 3-13. Main Menu for Machine Management

If the “configuration” option is selected from the above menu, the following menu options are also displayed.



```
                   Operations, Administration and Maintenance
2         Configuration Management
hardware - Display Summary Configuration Information
system   - System Name and Version Information
```

The following table shows how the tasks listed on the `machine` menu correspond to the shell commands discussed throughout this chapter.

Task to Be Performed	**sysadm** Task	Shell Command
Shutting down your machine	shutdown	**shutdown(1M)**
Rebooting your system	reboot	**shutdown(1M)**
Displaying who is logged on	whos on	**who(1)**
Displaying configuration info.	hardware	**hwstat(1)**
Displaying system name	system	**uname(1)**

If you are not an expert administrator, you may find it easier to perform these tasks through the **sysadm** interface.

Quick Reference to Machine Management

- Shutting down your computer from multi-user state:

  ```
  shutdown -i0
  ```

  or

  ```
  shutdown
  ```

  shuts down the system

- Shutting down your computer from single-user state:

  ```
  shutdown -y -i0 -g0
  ```

  where the **-y** option means all questions should be answered by “yes,” **-i0** shuts down the system to state 0 (meaning off), and **-g0** defines the grace period as 0 seconds. You can use **-y** and **-g0** in this case because you're the only user on the computer.

- Rebooting your system:

  ```
  shutdown -i6
  ```

  where **-i6** shuts down the system to state 6, meaning stop the system and reboot.

- Displaying hardware configuration information:

  ```
  hwstat
  ```

  produces a display that includes memory and peripheral configuration information.

- Displaying the system name and operating system release number:

  ```
  uname -sr
  ```

  displays your system name (such as `unix`) and the operating system release number.

- Displaying who is logged on to your computer:

  ```
  who -H
  ```

  produces a list of users logged on to your computer; the ID, terminal number, and login time of each user is also shown. The **-H** option adds field headers to the display.

4
Creating and Managing User Accounts

4
Creating and Managing User Accounts

Introduction

The job of managing the accounts of individuals and groups that work on your computer includes several important responsibilities, which are described below.

- Maintain security by controlling access to the computer. To do this, assign login names with passwords to all users.

  System security can also be implemented on two other levels once a user logs in, these levels are: group management and Mandatory Access Control. Your computer uses a concept of “group membership” to control access to certain files and directories. Each file and directory is a member of a group (identified with a “permission code”). Those members belonging to the same group as the file or directory are allowed access. By establishing and maintaining user group assignments, you can control user access to specific directories and files on your system. (Details of these permissions are described later in this chapter.)

  Mandatory Access Controls restrict access to files, directories, and resources on the basis of security levels. A level is a combination of a hierarchical classification and a set of categories. The procedure for assigning levels to users is described later in this chapter. Mandatory access control is described in Chapter 17, “Administering Mandatory Access Control and Multilevel Directories”.

- Streamline your environment to suit the particular needs of your users. For example, you may want to organize system resources differently for a group of programmers than you would for a group of writers. See the “Managing System Performance” chapter in volume 2 of *System Administration* for information about how to contour system performance to meet these specialized user needs.

- Keep the users on your system informed about system status and service schedules. There are several ways to do this, and they are described in Chapter 1, “Introduction to Basic Administration”.

This chapter tells you how to use the operating system shell commands to set up and control accounts for users and user groups, file and directory access, and command authorization on your computer. The specific online manual pages provide detailed explanations of the shell commands.

If the Operations, Administration and Maintenance (OA&M) package (a non-graphical menu interface) is installed on your system you can use it to complete many of these tasks. (See *“Managing User Logins through OA&M Menus”* later in this chapter.)

Controlling Access to the System and Data

You can implement some basic procedures that will help ensure the security of your computer and the data stored in it. The first line of defense is to make sure that only authorized users can use the computer. Login names and passwords, together, function like a combination lock on a safe. To the extent that you enforce their use, you can control access to your computer system.

You can also protect the data on your computer by assigning permission codes to individual directories and files. Permission codes restrict access to only the groups and individuals you specify.

The Function of Login

When a user requests access to a computer, the `login:` prompt is displayed and the user must enter a string of characters called a "login name." Next, the `Password:` prompt is displayed and the user types his or her password. The computer verifies that the password entered is valid for the login name entered. If it is, the user is allowed access to the computer. If it isn't, the computer prompts the user again with the **`login`** prompt.

NOTE

Some administrative operations use commands which require privileges to override certain sensitive operations. See Chapte r9, "Administering Privilege" for information on using privileges.

CAUTION

Assign passwords to all logins. See Chapter 14, "User Account and Group Management" for information about passwords and other security precautions you should take with your computer.

Login names and passwords are maintained with the commands **`passwd`**, **`useradd`**, **`usermod`**, and **`userdel`**, as described in *"Creating and Maintaining User Logins",* later in this chapter.

The Function of Directory and File Permissions

In addition to controlling access to your computer, you can control access to particular files and directories by assigning permission codes to them. Each permission code is a ten-character string that shows who can access the data in question. It can be viewed, as part of a directory or file listing, by running the **`ls`** command with the **`-l`** option. For a detailed explanation of directory and file permissions, see the description of the **`ls`** command in the *User's Guide* and **`umask(1)`** in the online manual page.

File permissions provide a simple, easy-to-use mechanism for controlling access to files. If you need a more powerful control mechanism, you can use an Access Control List (ACL). An ACL allows the owner of a file to grant or deny access to a file on a per-user basis, for any number of specific users. For more information on ACLs, see Part 2, the "Security Administration" part of this book.

Creating and Maintaining User Logins

Create at least one login name for every user. You can assign groups to a login when the login is created, or, if the login already exists, you can change group assignment by using the **usermod** command.

If you have a large computer and you must create logins for new users frequently, you may want to schedule this work for a time when the system is not busy. If you have a small computer, the time needed to add one user login is negligible.

Adding User Logins

The simplest way to create a login for a new user is by typing

```
useradd -m login_name
```

where *login_name* is the login name of that user. This command defines and creates a default directory as the new user's home directory, referencing a set of default environment values to create the new user's environment.

It also creates entries for the new user in two files: **/etc/passwd** and **/etc/shadow**. These files contain user login credentials, that is, login name, password, UID, GID, and so on. [See **passwd(4)**in the online manual.]

You must also assign a temporary password to the login and tell the user what it is. A login cannot be used until a password has been assigned. (See *"Assigning and Changing Passwords"* later in this chapter.)

CAUTION

> Never edit the **/etc/passwd** or **/etc/shadow** files directly. Doing so will cause the files to become corrupted. Entries in these files can be modified with the **useradd, usermod, userdel, groupadd, groupmod,** or **passwd** commands.

To Display the Default Values for the useradd Command:

Type the following command to see the default values that are used by **useradd**:

```
defadm useradd
```

See the **`defadm(1M)`** online manual page for complete details on its use.

To Set Default Values for the useradd Command:

You can change the default values used by **`useradd`**. When you do, the values you define will apply to all future logins you create. Type

```
defadm useradd parameter=value
```

with any or all of the following parameters:

- `SHELL=`*default_shell*
- `HOMEDIR=`*default_home_dir*
- `SKELDIR=`*default_skel_dir*
- `FORCED_PASS=`*default_forced_pass*
- `GROUPID=`*default_group*
- `INACT=`*default_inactive_days*
- `EXPIRE=`*default_expire_date*
- `DEFLVL=`*default_login_level*
- `AUDIT_MASK=`*default_audit_mask*

For example,

```
defadm useradd SHELL=/usr/bin/ksh
```

will change the automatically-supplied default shell to **`ksh`**

To Add a User, Specifying Non-default Values:

The **`useradd`** command offers another way to override the default environment parameters when you are creating a new user account and the login name associated with it. The list that follows describes some of the options that are frequently used to define (or re-define) parameters for your users. See the **`useradd(1M)`** online manual page for complete descriptions of all available options.

NOTE

> The system file entries created when you run the **`useradd`** command have a limit of 512 characters per line. If you specify long arguments to several options, you may exceed this limit.

-c *comment*
Store information about the owner of each login in the **/etc/passwd** entry for that user. If the comment contains spaces, it must be enclosed in double quotes, as in the following example:

-c “John P. Sousa”

Do not include colons in your comment (in the **/etc/passwd** file they are interpreted as field separators).

-d *dir*
Make *dir* the home directory of the new user instead of the default. *dir* must be a full path (such as **/home2/login**).

-e *expire*
The login cannot be used after the date shown by *expire*. (This option is useful for creating temporary logins.)

-f *inactive*
The maximum number of days allowed between uses of a login name before that login is declared invalid.

-G *group*
An existing group's integer ID or character-string name. It defines the new user's supplementary group membership. Supplementary groups are those that can be accessed with the **newgrp** command. More than one *group* can be specified in a comma-separated list. (See also the **-g** option.)

-g *group*
An existing group's integer ID or character-string name. It defines the new user's primary group membership and defaults to the value defined by GROUPID. To change the default *group*, use **defadm(1M)**.

-h *level*
A security level at which a new user may log on. The level may be specified as either a level alias or a fully-qualified level. You may specify the **-h** option multiple times to allow the user to log on at more than one level.

-k *skel_dir*
Copy the files in the named “skeleton” directory (one containing files such as **.profile**), into a new user's home directory. The default skeleton directory is **/etc/skel.** If the **-k**, option is used, *skel_dir* must already exist.

-m
Create a home directory if one doesn't already exist. The new login remains locked until **passwd** is executed.

If **-m**, is not specified, you must create a default home directory and give the user ownership of it (see **chown(1M)** for details.) It must also have read, write, and execute permissions by *group*, where *group* is the user's primary group.

-s *shell*
Use *shell* as the user's shell when that user logs on instead of the default. *shell* must be the full path of a valid executable file (such as **/usr/bin/ksh**). The default shell used is defined in **/etc/default/useradd** when the **-s** option is not used.

-u *UID*
Make *UID* the user ID of the new user, instead of the default. *UID* must be a non-negative decimal integer below MAXUID as defined in **/usr/include/sys/param.h**, a file that contains default values for various system parameters. The UID defaults to

the next available (unique) number above the highest number currently assigned. For example, if UIDs 100 and 105 are assigned, the next default UID number will be 106. (UIDs from 0-99 are reserved.)

-v *def_level* Make *def_level* the initial default level of the new user. The level may be specified as either a level alias or a fully-qualified level. The user will be able to change the default level to any level at which the user is authorized to log on.

-w *level* Set the security level of the home directory to *level*.

You can use these options to change the default values that are used when you add a new user login with the **useradd** command:

a. For each user you want to add, type:

   ```
   useradd [option . . .] login_name
   ```

   where *option* may be any or all of the options described above or on the **useradd** online manual page.

b. For each user you have added, assign a temporary password to the account. Once an account has been added, it cannot be used until a password has been assigned. (See the next section.)

For example, the **/etc/skel** directory can contain files that provide a working environment for a user, including the **.profile** file. When you create a new login account with **useradd -m** and accept the default skeleton directory (by *not* using the **-k** option), then the files in **/etc/skel** will be copied into the new user's home directory.

In some cases, however, you may want to provide a different working environment than the one defined by the files in **/etc/skel.** To handle such cases, you can create another skeleton directory by copying the files from **/etc/skel** and edit them to suit your needs (or create new files). For example, if you have created an alternative skeleton directory in **/mylogin/newskel,** then the following command will create a new user account for user harvey and copy the files in **/mylogin/newskel** to Harvey's home directory:

```
useradd -m -k /mylogin/newskel harvey
```

Assigning and Changing Passwords

Passwords must be assigned to new logins before they can be accessed. To assign or change a password use the procedure which follows:

1. Create a temporary password for the user by running the **passwd** command, specifying the user's login name:

   ```
   passwd login_name
   ```

2. The following prompt appears:

   ```
   New password:
   ```

   Enter a temporary password.

3. The following prompt appears:

   ```
   Re-enter new password:
   ```

   Type the same password again.

4. To force the user to change the temporary password the next time he or she logs in, re-execute the **passwd** command with the **-f** option:

   `passwd -f` *login_name*

In addition, you may want to set up some parameters for the password you are creating. For example, for security reasons, you might want to restrict the amount of time a user is allowed to use a temporary password before replacing it with a permanent one. You can create restrictions such as this at any time.

- You can force a new user to replace a current (or temporary) password with a permanent one at the next login session. (Run **passwd -f** *login_name*.)
- You can assign a minimum number of days that the new user will be forced to keep a current (or temporary) password before being allowed to replace it. (Use the **-n** *min* option to **passwd**.)
- You can request a date (specified by the number of days before the expiration of the current password) on which a user will be warned about the impending expiration. (Use the **-w** *warn* option to **passwd**.)
- You can specify the maximum number of days a user will be allowed to keep a current password before being forced to replace it. (Use the **-x** *max* option to **passwd**.)

See the **passwd(1)** online manual page for complete details.

Inevitably, some users forget their passwords. Establish a policy for changing forgotten passwords that will help protect the security of your system. You may want to require users who forget their password to see you in person before you change their password.

Removing a User

There are two ways to remove a user's login name from your system: you can remove only the login entry, or you can remove the login entry and the user's files. If you remove only the login entry, the user's files remain. Preserving these files and directories may be a good idea in case any of them are needed by other users.

As you know, you can assign an expiration date for a password, or limit the time that can elapse between uses of an account, after which it is declared invalid and locked. For performance and security reasons, such logins should be removed from the password database. In addition, of course, whenever you know an account is no longer going to be used, you should remove it.

When you remove a user's login with **userdel**, that user's UID is recorded so the UID will not be reused for a period of time. The number of months the system will wait before reusing a UID can be specified with the **-n** option of **userdel**. The default value can be changed with the **defadm userdel** command.

In the procedure below, the first step presents the choice of removing only the user's login entry, or of removing the login entry and all files and directories associated with the login. If you remove the user's directories and files, they will be accessible only from backup tapes.

In the second step, the **`adminuser`** command prevents possible security breaches by removing all privileges that may have been assigned to that user.

1. Select the appropriate command to remove a user's login entry:

If you want to remove:	Then execute:
only the login entry	**`userdel`** *login_name*
the login entry and all associated files and directories	**`userdel -r`** *login_name*

2. To remove any privileges that may have been assigned to the login, execute:

   `adminuser` **`-d`** *login_name*

Not all users have privileges. If the user you are removing does not, the following message will be displayed:

```
UX:adminuser: ERROR: Undefined user “login_name”
```

Adding Groups

NOTE

> Groups must be added to your computer before users can be assigned to them.

The purpose of creating group IDs is to restrict access to special commands or to information about a special project. Those users (that is, their login IDs) who need access to the project or commands can be assigned to that group, either as their primary group or as one of their secondary groups. (Refer to the **`useradd(1M)`** and **`groupadd(1M)`** in the online manual pages for more information about group IDs.)

NOTE

> Because the names of users, groups, and home directories will be readable by any user on the system, the names themselves should not be sensitive to disclosure.

To add a group, complete the following procedure:

1. Type

   `groupadd` *group_name*

   The group will be assigned a GID automatically. To assign a particular GID, use the **`-g`** option and specify the desired GID:

   `groupadd` **`-g`** *GID group_name*

2. To add users to the group, run the **`useradd`** command.

The time needed to add one group is negligible. If you need to add groups to your system frequently, however, you may want to schedule these additions for a time when the system is not busy.

Removing Groups

To delete a group definition from the system, use the **`groupdel`** command:

`groupdel` *group*

This deletes a group definition from the system by removing the appropriate entry from the **`/etc/group`** file.

Modifying User and Group Attributes

Users' working lives change regularly: they transfer departments, receive promotions, change projects, and so on. When they do, the status (attributes) of these users may need to be updated. For example, users who change projects may no longer need access to the files available to them on their previous project. In addition, special projects may be created with information that must be restricted to a select few.

In these cases, you should create a new group (and possibly new login names) for the project. (See *"Adding User Logins"* earlier in this chapter.) Be sure the file creation mask is set appropriately in the **`.profile`** files for the users assigned to the new group. (See *"The File Creation Mask (umask)"* later in this chapter.)

To Modify User Attributes:

The **`usermod`** command changes the status or attributes of existing user logins. Most of the options available with **`usermod`** parallel the options available for defining user attributes with **`useradd.`** The following options, (described earlier in this chapter in the section *"Adding User Logins"),* are also available with **`usermod.`**

`-a` *event*
`-c` *comment*
`-d` *home_directory*
`-e` *expire*
`-f` *inactive*
`-G` *supp_group*

-g *primary_group*
-h *level*
-s *shell*
-u *UID*
-v *def_level*

The following two options have specific behavior that applies only to **usermod.**

-l *new_login* A string of printable characters that specifies the new login name for a user. It must not contain a colon (**:**) or a newline (**\f1**).

-m Move the home directory of the user to a new location (specified with **-d**). If this new location already exists, the user being added must have access permission to the directory. (This option takes no arguments.)

See the **usermod(1M)** online manual page for complete descriptions of all available options.

NOTE

Do not modify a user's account while that user is logged on.

1. For each user account you want to modify, type:

   `usermod [`*option . . .*`]` *login_name*

2. Notify the user that you have modified his or her account.

   Changes made with the **usermod** command do not take effect until the next time the user logs on. In most cases this is of no consequence. If the changes need to become effective immediately, however, the user must be made to log off and log back on again.

To Modify Group Attributes:

The **groupmod** command modifies the definition of a group that is stored in the **/etc/group** file. See the **groupmod(1M)** online manual page for complete details on its use.

Invoke the **groupmod** command, with the appropriate options, to do the following tasks:

- Change the group ID:

  `groupmod` **`-g`** *GID group*

- Override the unique group ID constraint so you can use a single group ID for more than one group:

  `groupmod` **`-o -g`** *GID group*

  Use this option only with caution; using the same group ID for multiple groups is not recommended.

- Rename a Group: As the needs of your users change, you may want to change the name of a group without changing its membership. To do so, type

  `groupmod` **`-n`** *new_group_name old_group_name*

  where *new_group_name* is the new name for the group and *old_group_name* is the current name.

  For example, to change a group name from **`tech`** to **`pub,`** type

  ```
  groupmod -n pub tech
  ```

Getting Information about Users and Groups

During normal administrative activities, it may be necessary to check up on user and group assignments. The following procedures allow you to do this.

- To List All Users and Groups:

  ```
  listusers
  ```

- To List Information About One or More Users:

  `listusers` **`-l`** *login_name1 login_name2*

- To List Users Belonging to a Specific Primary or Secondary Group:

  `listusers` **`-g`** *group_name_list*

  where the names of groups are separated by commas

- To List Logins with Duplicate Login Names:

  ```
  logins -d
  ```

- To List Logins with Supplemental Group IDs:

  ```
  logins -m
  ```

- To List Logins with Unassigned Passwords:

  ```
  logins -p
  ```

- To Display Extended Information About Logins:

  ```
  logins -x
  ```

Streamlining the Work Environment: System and User Profiles

Your computer does several routine tasks between the time a user logs on and the time a shell prompt appears. These tasks are listed in the two files that are executed during the login sequence: the “system profile” (**`/etc/profile`**) and the “user profile” (**`$HOME/.profile`**). This section describes these files and explains how you can change them to enhance the performance of your system so it best serves the needs of your users.

The System Profile

When a user enters a valid login name and password, the computer runs a program called the system profile (**`/etc/profile`**). This program is an executable ASCII file that performs several important actions:

- It defines and exports some environment variables. (For details, see *“Environment Variables,”* later in this chapter, and the “Managing System Performance” chapter in volume 2 of *System Administration.)*
- The message of the day is displayed. (See *“Message of the Day”* in Chapter 1, “Introduction to Basic Administration”.)
- A list of news items is displayed if the user is not **`root`**. (See *“News”* in Chapter 1, “Introduction to Basic Administration”.)
- A message about mail items is displayed if the user has mail. (See *“Communicating with Users”* in Chapter 1, “Introduction to Basic Administration”.)
- The default user mask is defined. (See *“The File Creation Mask (umask)”* later in this chapter.)

A sample **`/etc/profile`** is shown in Scree n4-1.

```
trap ““ 1 2 3
umask 022# set default file creation mask
export LOGNAME


PATH=$PATH:/usr/ccs/bin:/usr/ucb

case “$0” in
-jsh | -ksh | -rsh | -sh)
# issue message of the day
    trap : 1 2 3
    echo ““  # skip a line
    if [ -s /etc/motd ]; then
        cat /etc/motd;
    fi
    trap ““ 1 2 3

# Set up reasonable terminal behavior.
    if [ -x /sbin/stty ]; then
        /sbin/stty intr '^c' erase '^h' kill '^u' swtch '^`' echoe tab3
    fi

# Set up a reasonable TERM environment variable.
    if [ -x /usr/bin/tty ];; then
        tty | grep pts > /dev/null 2>&1
        if [ “$?“ != “0” -a “$TERM” = ““ ]; then
            TERM=`/usr/bin/tty`; TERM=`/bin/basename $TERM`
            TERM=`/usr/bin/grep $TERM /etc/ttytype`
            TERM=`/usr/bin/basename $TERM`
        fi
    fi

    if [ “$TERMCAP” = ““ ]; then
        TERMCAP=/usr/share/lib/termcap
    fi
    export TERM TERMCAP

# check mailbox and news bulletins
    mailcheck 2>/dev/null
    if [ $LOGNAME != root -a -d /var/news ]
    then news -n
    fi
#   Uncomment this script if you wish to use secure RPC facility
#
#   ps -e | grep rpcbind 1>/dev/null
#   if [ $? = 0 ]
#   then
#       ps -e | grep keyserv 1>/dev/null
#       if [ $? = 0 ]
#       then
#           echo “Please login to the network”

#           /usr/bin/keylogin
#       else
#           echo `date`: “secure rpc nonfunctional; keyserv is down” >>/var/
adm/log/rpc.log
#       fi
#   else
#       echo `date`: “secure rpc nonfunctional; rpcbind is down” >>/var/adm/
log/rpc.log
#   fi
#
#   ;;

esac
export PATH
trap 1 2 3
```

Screen 4-1. A Sample System Profile (`/etc/profile`)

You can change existing definitions by editing the system profile. For example, you may want to change the value assigned to your system mask from`022` to `077` to make files and directories created by users more secure. [See **`chmod(1)`** online manual page for information on file permissions.] Also, you may want to change the default `PATH` to provide access to locally developed commands.

You can also edit the system profile to automate other tasks. For example, you can add shell commands to this file to display the current time and date, inform a user that he or she has mail, and tell a user the number of people logged in on the computer, as shown in Figure 4-1.

.

```
echo `date`
mail -e
n=`who | wc -l`
echo “There are $n users on `uname -n`.”
```

Figure 4-1. Additional Shell Script for the System Profile (`/etc/profile`)

NOTE

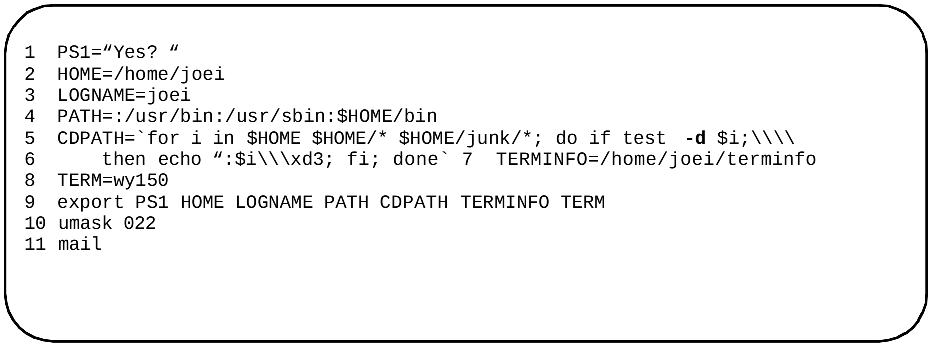
> You should change the system profile only when the change you make will be beneficial to all users of the computer.

The User's Profile

If you create a login for a new user with **`useradd -m`**, the contents of the directory **`/etc/skel`** are copied to the new user's home directory by default. The **`/etc/skel`** directory contains a standard user profile, **`.profile`**, that can list tasks and environment variables in addition to those listed in the system profile file. This user's profile allows you to further define the working environment that is set up when the user logs in. If you want to provide other directories and files (such as an **`rje`** directory or a **`.mailrc`** file), you can add them to the **`skel`** directory.

You can also create other skeleton directories, each having a customized **`.profile`** (and any other files and directories you want them to have). Then, whenever you add a login name to your system, you can use the **`-k`** *skel_dir* option to **`useradd`** to name the full pathname of the particular skeleton directory you want copied to that user's home directory. This mechanism eliminates the need to add directories and files, individually, to a new home directory.

After executing the system profile, the computer executes the user's profile. The user's profile executes commands and shell scripts in the same way the system profile does. The operating system shows a sample **`.profile`** file.



```
PS1=“Yes? “
HOME=/home/joei
LOGNAME=joei
PATH=:/usr/bin:/usr/sbin:$HOME/bin
CDPATH=`for i in $HOME $HOME/* $HOME/junk/*; do if test  -d $i;\\\\
    then echo “:$i\\\xd3; fi; done`  TERMINFO=/home/joei/terminfo
TERM=wy150
export PS1 HOME LOGNAME PATH CDPATH TERMINFO TERM
umask 022
mail
```

Screen 4-2. A Sample User's Profile (`$HOME/.profile`)

An individual user can add, remove, or redefine environment variables in the **`.profile`** file in his or her login directory. Many users like to do this to customize their work environment to an even more personal degree (to suit factors such as the type of terminal being used, their favorite text editor, or the type of work being done).

If you want to change the environment for new users, you can modify **`/etc/skel/.profile`** to include some or all of the environment variables shown in the operating system. These environment variables are defined below.

Environment Variables

This section describes the environment variables defined in the sample user profile shown in the operating system. See the “Managing System Performance” chapter in volume 2 of *System Administration* for details.

`PS1` The user's shell-level prompt.

`HOME` The user's home directory. Scripts and system programs that reference the user's home directory use this environment variable to find it.

If a user is moved to another file system, the only change needed in the user profile, to simulate the original working environment, is the definition of `HOME`. For example, if a user whose login name is `jean` is moved from the **`/home`** file system to the **`/home2`** file system, the only change needed in her user profile is a change in the definition of `HOME` from **`/home/jean`** to **`/home2/jean`**. Once `HOME` has been redefined, the user can access all the files and commands that were available in the environment defined by the previous value of `PATH`.

`LOGNAME` The user's login name. Scripts and system programs that reference the user's login name use this environment variable to find it.

`PATH` A list of directories and the order in which these directories are searched for a command requested by a user. This order may be important: when identically named commands exist in different locations, the first command found with that name is used. For example, assume PATH is defined as **PATH=/usr/bin:/usr/sbin:$HOME/bin**. If a user invokes a command called **sample** without specifying its full path, and this command resides in both **/usr/bin** and **$HOME/bin**, the version found in **/usr/bin** will be used.

CDPATH

Specifies the directories searched when a unique directory name is typed without a full path. This environment variable is used as an argument to the **cd** command. For example, **bin** and **rje** exist under **/home/jean**: If you are in **/home/jean/bin** and type cd **rje**, you will change directories to **/home/jean/rje** even though you haven't specified a full path.

TERMINFO

Definitions of supported terminals are found in the default terminal information database, **/usr/share/lib/terminfo**. When using an unsupported terminal, a user can create a definition for it in another terminfo directory and define TERMINFO as a path to this directory (such as **/home/jean/terminfo)** in his or her profile. The system first checks the TERMINFO path defined by the user. If a definition for a terminal is not found there, the system searches the default directory, **/usr/share/lib/terminfo,** for a definition. If a definition is not found in either location, the terminal is identified as “unknown.” [See “Directories and Files”, in this book, **tic(1M)** online manual page, **curses(3X)** online manual page, **term(4)** online manual page, and **terminfo(4)** online manual page for more information about the **terminfo** directory and the commands used with it.]

TERM

The terminal used by this user. When the user invokes an editor, the computer looks for a file with the same name as that defined by this environment variable. (For example, in the operating system the terminal was defined as a wy150 model.) The system searches the directory referenced by TERMINFO (in the operating system, **/home/jean/terminfo)** to determine the characteristics of the terminal.

New environment variables can be defined in a user's profile at any time. By convention, environment variables are defined as follows:

VARIABLE_NAME=value

The variable name appears in upper case, followed by an equals sign and the value. Once an environment variable is defined, it can be used globally by executing the **export** command with the environment variable as an argument to it.

The File Creation Mask (umask)

The default permissions (mask) used for files and directories created by a user are determined by the values assigned to the file creation mask. The**umask** command assigns values to the mask. The system profile may contain the **umask** command to establish these permissions. Default permissions may be redefined in a user's profile.

Specify the mask as a three-digit octal number. The system determines permissions by subtracting the value of the mask from 777.

For example, if the default mask is set by the system profile to mode `027`, the command line **`umask 022`** causes newly-created directories to be assigned mode `755` (`drwxr-xr-x`) and newly-created files to be assigned mode `644` (`-rw-r--r--`). If the default mask is set by the system profile to mode `027`, the command line **`umask 027`** causes newly-created directories to be assigned mode `750` (`drwxr-x---`) and newly-created files to be assigned mode `640` (`-rw-r-----`). The system checks the permission settings before allowing or denying access to a file or directory.

NOTE

> The value of the executable bit is ignored for the creation of text files but not for the creation of directories.

Access can be given to one or more of the following sets of users:

- the owner of the data
- people with the same group ID as the owner of the data
- all other people using the computer

For details, see **`umask(1)`** online manual page.

The **`umask`** provides a mechanism to restrict access to newly created files. If a more powerful mechanism is needed, Access Control Lists (ACLs) offer additional flexibility. For details, see Part 2, *Security Administration*.

User Privileges

The system administrator controls which users have access to commands and system services that require privilege. You can assign privileges to individual users or groups of users.

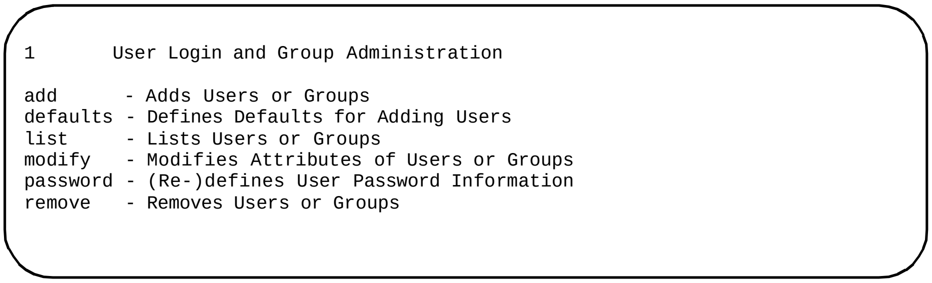
Administrators and privileged users need to perform sensitive tasks, but because privileges are associated with processes and executable files, not user IDs (except for the special case of UID=0 when using the SUM privilege policy module), it is not possible to grant privileges to users directly. The Trusted Facility Management tools (TFM) provide the means to maintain a database of users and the commands they may execute with privilege. If privileges are assigned to a user's shell, they will be inherited by all commands executed by that shell.

This eliminates the need to place fixed privileges on commands (via **`filepriv`**) for a user to execute commands with privilege.

Refer to Chapter 9 “Administering Privilege” and Chapte r10 “Trusted Facility Management” for additional information.

Managing User Logins through OA&M Menus

The system administration menus are only available if the Operations, Administration and Maintenance (OA&M) package is installed on your system. To access the system administration menu for user and group management, type **`sysadm users`**. The following menu will appear on your screen:0



```
1       User Login and Group Administration

add      - Adds Users or Groups
defaults - Defines Defaults for Adding Users
list     - Lists Users or Groups
modify   - Modifies Attributes of Users or Groups
password - (Re-)defines User Password Information
remove   - Removes Users or Groups
```

The following table shows how the tasks listed on the `User Login and Group Administration` menu correspond to the tasks discussed throughout this chapter.

Task to Be Performed	**`sysadm`** Task	Shell Command

Add users or groups	`add`	**`useradd(1M)`**
		`groupadd(1M)`

Change users passwords	`password`	**`passwd(1)`**
Define defaults for adding users	`defaults`	**`defadm(1M)`**

List users and groups	`list`	**`logins(1M)`**
		`listusers(1M)`

Modify user or group attributes	`modify`	**`usermod(1M)`**
		`groupmod(1M)`

Remove a user or group	`remove`	**`userdel(1M)`**
		`groupdel(1M)`

Quick Reference to Managing User Logins

- Adding a user with default options:

  `useradd` **`-m`** *login_name*

- Adding a user with non-default options:

  ```
  useradd | -h level | -v default_level | -a audit_mask \
  | -u user_ID | -g group_ID | -c comment \
  | -d home_directory | -m | -s shell login_name
  ```

  ```
      passwd -n min_days -x max_days -f login_name
  ```

- Adding a group:

  ```
  groupadd -g group_ID group_name
  ```

- Listing all users:

  ```
  listusers
  ```

- Listing user information:

  ```
  listusers -l login_name1 login_name2
  ```

- Listing users assigned to a group:

  ```
  listusers -g group_name_list
  ```

  Separate the names of groups by commas.

- Listing users with duplicate user IDs:

  ```
  logins -d
  ```

- Listing users with supplemental group IDs:

  ```
  logins -m
  ```

- Listing users with unassigned passwords:

  ```
  logins -p
  ```

- Displaying extended information about users:

  ```
  logins -x
  ```

- Modifying the user's comment field:

  ```
  usermod -c comment login_name
  ```

- Changing the user's home directory field:

  ```
  usermod -d home_directory login_name
  ```

- Modifying the user's group name:

  ```
  usermod -g existing_group login_name
  ```

- Adding the user to a supplementary group:

  ```
  usermod -G supp_group login_name
  ```

- Changing the user's login:

  ```
  usermod -l new_login_name old_login_name
  ```

  or

```
usermod -l new_login_name -d new_dir old_login_name
```

- Changing the user's home directory:

  ```
  usermod -m -d new_home_directory login_name
  ```

- Duplicating a user_ID:

  ```
  usermod -o -u UID login_name
  ```

- Changing the user's login shell:

  ```
  usermod -s shell login_name
  ```

- Changing the user's user_ID:

  ```
  usermod -u new_UID login_name
  ```

- Changing the user's login expiration status:

  ```
  usermod -e MM/DD/YY login_name
  ```

- Changing the inactive status of a user's login name:

  ```
  usermod -f number_of_days login_name
  ```

- Creating a new group:

  ```
  groupadd [-g group_id] group_name
  ```

- Making the group ID a duplicate group ID:

  ```
  groupmod -g current_GID -o group_name
  ```

- Changing a forgotten password for a user:

  1. passwd *login_name*
  2. passwd **-f** *login_name*

- Removing only a login entry:

  ```
  userdel login_name
  ```

- Removing a login entry and the user's home directory:

  ```
  userdel -r login_name
  ```

- Changing a group ID:

  ```
  groupmod -g new_group_ID group_name
  ```

- Changing a group name:

  ```
  groupmod -n new_group_name old_group_name
  ```

- Removing a group name:

  ```
  groupdel group
  ```

5
Managing Ports

5
Managing Ports

Introduction

This chapter is divided into six sections, describing the components used to set up and manage the ports that provide access across systems and in a multi-user environment. The port monitors supplied for this purpose and the commands used to administer system files are explained. A *"Quick Reference Guide to Managing Ports"* tells you how to invoke the `sysadm` menus for administration of system ports. The quick reference also provides a summary of the **`sacadm`**, **`pmadm`**, and **`sttydefs`** command lines discussed throughout the chapter. The first section, *"Overview of the Service Access Facility"* describes the basic files and facilities used in configuring ports and the types of port monitors that may be established. The second section, *"Port Monitor Management"*explains how you can use the **`sacadm`** command to manage the SAC. Specifically, it tells you how to

- print status reports about a port monitor
- add or remove a port monitor
- enable or disable a port monitor
- start or stop a port monitor

This section also describes the configuration scripts used to modify the SAC and port monitor environments.

For every port monitor on your system, you must have an administrative file. The third section, "*Service Management*," explains how to manage the information in this file through the **`pmadm`** command. These files contain information about the administration and status of each port and the service invoked by it. The **`pmadm`** command allows you to

- print information derived from the administrative file
- add (and remove) services
- enable (and disable) services
- print, install, and change per-service configuration scripts
- specify an authentication scheme for a service (See *Network Administration* for a description of authentication schemes.)

Following sections focus on the **`ttymon`** port monitor, terminal line settings in **`ttydefs`**, and the **`listen`** port monitor, known as "the listener."

The section on **`ttymon`** describes its role in monitoring asynchronous terminal devices, such as terminals, modems, and PCs being used as terminals. It also explains how to perform some of the administrative tasks described in the sections on port monitor management and service management.

The section on **`ttymon`** provides instructions on how to

- get a list of the **`ttymon`** port monitors that have been configured
- get a list of the services that have been configured under a given **`ttymon`** port monitor
- enable (and disable) **`ttymon`** ports and services
- add (and remove) a **`ttymon`** port monitor
- add a service under a **`ttymon`** port monitor
- specify an authentication scheme for a service under a **`ttymon`** port monitor

The management of terminal line settings is discussed in connection with **`ttymon`** management, since the **`ttydefs`** file (which replaces **`gettydefs`** as the database file for system terminal information) is used by **`ttymon`** and you'll have to modify it as part of **`ttymon`** administration. To set terminal modes and line speeds, you must learn how to

- print terminal line setting information
- modify terminal line settings
- set up hunt sequences
- add or remove terminal line settings for a terminal
- make terminal options available (using the **`stty`** command)

The section on the listener explains how to maintain and use the port monitor that oversees logical devices defined as layer 4 of the OSI Reference Model (that is, transport end points). Specifically, it describes how to

- get a list of configured **`listen`** port monitors
- get a list of services available through a given **`listen`** port monitor
- enable (and disable) **`listen`** ports and services
- add (and remove) **`listen`** port monitors
- add a service under a **`listen`** port monitor
- specify an authentication scheme for a service under a **`listen`** port monitor

Overview of the Service Access Facility

As an administrator, you're responsible for making your system accessible to those authorized to use it. These users may be local (logged on to your system directly) or remote (logged on to your system through a network). Both local and remote users gain access to a system through “ports”: physical and symbolic entry points to a computer for either users or processes.

What Is the Service Access Facility?

This release provided a new umbrella interface: the Service Access Facility (SAF). The SAF, which applies to systems operating in multi-user mode, provides a mechanism for uniform access to port monitor services. Each physical port on a computer running the current OS has a monitor associated with it that you can use to make the port active and otherwise manage its use. The monitor for a particular port controls use of the port with which it's associated: it grants (or denies) access, keeps track of usage, and provides information about availability to administrative programs that need to use the port.

Two port monitors, **`ttymon`** and **`listen`**, are delivered on newly installed systems running the operating system. These port monitors are provided for the following reasons:

`ttymon` runs ports used when local users log in.

`listen` runs ports used by remote users (through a network) to access accounts and software on the local system.

NOTE

You're not restricted to using these, however; if you like, you can develop your own. (For instructions on writing a port monitor, see *Network Programming Interfaces.)*

The SAF also provides a set of tools for installing, configuring, and querying port monitors through a utility called the Service Access Controller (SAC).

This chapter describes the **`ttymon`** and **`listen`** port monitors and the SAF tools provided for installing, using, and maintaining them.

The Service Access Facility consists of the following components:

- the Service Access Controller (SAC)
  - the SAC program (which runs the Service Access Facility)
  - a system-specific configuration script
  - an administrative file for the SAC
  - a command, **`sacadm,`** for administration of the SAC
- the **`ttymon`** and **`listen`** port monitors
  - the **`ttymon`** and **`listen`** programs
  - an administrative file for each port monitor
  - a configuration script (optional) for each port monitor

  - a command, **`pmadm`**, for administration of both port monitors

- service-specific configuration scripts (optional)

Administration of the ports on your system can be divided into two levels of responsibility (which reflect the two levels in the supporting directory structure). The top administrative level is concerned with general port monitor administration (through the **`sacadm`** command); the lower level is concerned with the administration of specific port monitors (through the **`pmadm`** command).

The Service Access Controller

The Service Access Controller (SAC) is the administrative point of control for all port monitors (and therefore for all ports on the system). Its job is to maintain the port monitors on the system in the state specified by you. It accomplishes this through three actions:

Customization — During initialization, the SAC customizes its own environment by invoking the per-system configuration script supplied with the SAF.

Start-up — Next, the SAC reads its administrative file to determine which port monitors are to be started. For each port monitor specified, the SAC interprets the corresponding configuration script (if one exists) and then starts the port monitor itself.

Polling — Once the SAC is running, it has two ongoing functions: (a) it polls its port monitors periodically, and (b) it initiates recovery procedures when necessary.

Because the SAC is started by the **`init(1M)`** command during system initialization (through an entry in **`/etc/inittab),`** it starts working as soon as you set up your system.

From time to time you'll want to query the SAC, for example, to check the status of the port monitors. You may also need to make changes to the port monitors. For example, you may want to enable a disabled port monitor or start a port monitor that has been killed. Two commands, **`sacadm`** and **`pmadm,`** allow you to issue commands to the SAC which, in turn, communicates with the port monitors.

Making Changes to Your Port Monitors

To get a report on port monitors known to the SAC, run the **`sacadm`** command with the **`-l`** option, or with an option that focuses on the details you wish to display. You may want to make changes in the group of port monitors on your system. For example, you may want to add or remove a port monitor from SAC supervision. The **`sacadm`** command allows you to make such changes.

If you decide to keep a port monitor under SAC supervision, but you want to assign a different state to it, run the **`sacadm`** command and specify one of the following states:

`STARTING` — An intermediate state (between `DISABLED` and `ENABLED`) when the port monitor is in the process of starting up.

ENABLED The port monitor is running and accepting connections. [See the **-e** option to **sacadm(1M)**.]

DISABLED The port monitor is running but is not accepting connections. [See NOTRUNNING and the **-d** option to **sacadm(1M)**.]

STOPPING An intermediate state on the way to NOTRUNNING; the port monitor has been manually terminated but has not completed its shutdown procedure.

NOTRUNNING The port monitor is not currently running. (See **sacadm** with **-k**.) This is the normal “not running” state. When a port monitor is killed, all ports it was monitoring are inaccessible. It's not possible for an external user to tell whether a port is not being monitored or the system is down. If the port monitor is not killed but is in the DISABLED state, it may be possible (depending on the port monitor being used) to write a message on the inaccessible port telling the user who's trying to access the port that it is disabled. This is the advantage of having the DISABLED state as well as the NOTRUNNING state.

FAILED The port monitor is not running. (It was not able to start and remain running. Regardless of what you specify, a port monitor will enter this state if the SAC can't start the port monitor after a specified number of tries.)

Ongoing Activities of the SAC

Once the port monitors are running, the SAC polls them periodically for status information. If during a poll, the SAC does not receive a response (such as ENABLED) from a particular port monitor, it assumes that monitor is not running. If it should be running, the SAC assumes it has failed and takes appropriate recovery action. The SAC also restarts a failed port monitor if a non-zero restart count was specified for the port monitor when it was created.

By default, the SAC polls all port monitors every 300 seconds (or five minutes). You can change this schedule, however, by running **sac -t** and specifying, in seconds, the interval after which you want polling to be repeated. For example, if you want polling done every ten minutes instead, enter

```
sacadm -t 600
```

[See **sac(1M)**, **sacadm(1M)**, and *“Port Monitor Management”*.]

Configuration Scripts for Individual Systems

If you want to customize the environment for services on your system, write a system-specific script that will be interpreted by the Service Access Controller when it's started (which occurs when the system enters multi-user mode). Write your script in **/etc/saf/_sysconfig,** an empty file delivered with the system for this purpose.

An ordinary shell script cannot be used; you must write this script in the interpreted language described on the **doconfig(3I)** online manual page and in *Network Programming Interfaces.*

Configuration Scripts for Individual Port Monitors

If you want to customize the environment for a given port monitor (and the services available through the access points for which it's responsible), you can write a configuration script for that particular port monitor. This script may override any defaults provided by a system-specific configuration script.

A port monitor-specific configuration script is interpreted when the relevant port monitor is started. (Port monitors are started after the SAC has been started and has run its own configuration script, **/etc/saf/_sysconfig.)**

Write your script in the same language used for system-specific configuration scripts (described above) and store it in **/etc/saf/***pmtag*/_config (where *pmtag* is the string by which the SAC recognizes the port monitor).

Configuration Scripts for Individual Services

You also have the option of writing service-specific configuration files. This capability is useful if you want to override the default values provided by the configuration files for your system and port monitor whenever a particular service is invoked. For example, you may want to have a set of STREAMS modules other than the default set used whenever a particular service is accessed.

Or, as another example, your system may offer a service that requires special privileges not available to general users. To avoid having to give your users all possible privileges, you can simply write a service-specific script that will grant or limit them when the service is accessed through a particular port monitor.

Like the configuration scripts described above, this type of script must be written in the language described on **doconfig(3I)** and in *Network Programming Interfaces.*

The SAC Administrative File

Information about all the port monitors for which the SAC is responsible is stored in a file that you maintain through the **sacadm** command: the SAC administrative file. You don't have to create this file; it exists on the delivered system. If you need to edit the file, you should not edit it directly. Its contents can and should be displayed and updated with the **sacadm** command. Initially, **/etc/saf/_sactab** is empty except for a single comment line that contains the version number of the Service Access Controller. Entries are added to this file to identify the port monitors on your system, that is, adding a port monitor is accomplished by adding an entry for it to this file.

Entries are added for each port monitor by executing the **sacadm** command with the **-a** option and appropriate information for the port monitor. This may be done by you as needed or automatically by an installation procedure for a network service or application that requires the setup of port monitors.

NOTE

> For a description of the complete command line syntax, see **`sacadm(1M)`**. This command permits you to add, update, remove, and display file entries for port monitors controlled by the SAC.

The first line of each SAC administrative file is typically a comment line indicating the version of the SAC.

Each entry in the SAC administrative file is a colon (`:`) delimited string that contains the following information:

`PMTAG` A unique tag that identifies a particular port monitor. The system administrator is responsible for naming a port monitor. This tag is then used by the Service Access Controller (SAC) to identify the port monitor for all administrative purposes.

`PMTAG` may consist of up to 14 alphanumeric characters.

`PMTYPE` The type of the port monitor. In addition to its unique tag, each port monitor has a type designator. The type designator identifies a group of port monitors that are different invocations of the same entity. **`ttymon`** and **`listen`** are examples of valid port monitor types. The type designator is used to facilitate the administration of groups of related port monitors. Without a type designator, the system administrator has no way of knowing which port monitor tags correspond to port monitors of the same type.

`PMTYPE` may consist of up to 14 alphanumeric characters.

`FLGS` The following flags are currently defined:

d	When started, do not enable the port monitor.
x	Do not start the port monitor.

If no flag is specified, the default action is taken. By default a port monitor is started and enabled.

`RCNT` The number of times a port monitor may fail before being placed in a failed state. Once a port monitor enters the failed state, the SAC will not try to restart it. If a count is not specified when the entry is created, this field is set to `0`. A restart count of `0` indicates that the port monitor is not to be restarted when it fails.

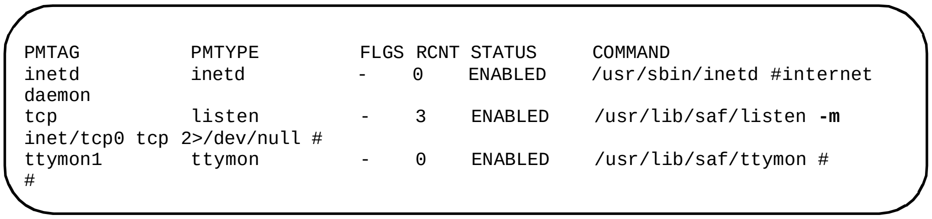
`COMMAND` A string representing the command that will start the port monitor. The first component of the string, the command itself, must be a full pathname.

Each string comprising an entry in the file is terminated by a pound sign (#), which indicates the start of an optional comment.

Screen 5-1 shows the output of

```
sacadm -l
```

invoked to display the complete list of entries in a sample SAC administrative file.



```
PMTAG          PMTYPE         FLGS RCNT STATUS     COMMAND
inetd          inetd          -    0    ENABLED    /usr/sbin/inetd #internet
daemon
tcp            listen         -    3    ENABLED    /usr/lib/saf/listen -m
inet/tcp0 tcp 2>/dev/null #
ttymon1        ttymon         -    0    ENABLED    /usr/lib/saf/ttymon #
#
```

Screen 5-1. Output of `sacadm -l`

The Port Monitor Administrative File

Each port monitor has its own administrative file, **`/etc/saf/`***pmtag*/_pmtab that you maintain through the **`pmadm`** command. Whenever you make changes to a port monitor on your system, the **`pmadm`** command records those changes by adding, deleting, or modifying the relevant entries. Each type of port monitor has separate administrative files. Changes to a file are made only by the appropriate port monitor (**`ttymon`**, **`listen`**, or one you created) which immediately rereads its file whenever a change is made.

Each entry in a port monitor administrative file indicates:

- The service to be invoked on a specific port
- How the port monitor should treat that port

Each entry within the file is uniquely identified by a service tag. The combination of a service tag and a port monitor tag is a unique string that defines an instance of a service.

NOTE

> A single service tag may be used in more than one file to identify the same service under multiple port monitors. In other words, for consistency and recognition, the same service tag may be used in more than one port monitor administrative file.

The entry must also contain port monitor-specific information that's meaningful to a particular port monitor. For example, entries for `ttymon-type` port monitors may include prompt strings.

To add information to a port monitor administrative file, execute the command for the appropriate port monitor type:

`ttyadm`	for a **`ttymon`** port monitor
`nlsadmin`	for a **`listen`** port monitor

NOTE

> If you're installing new software for a network service or application, it may include an installation procedure that automatically adds the appropriate entries to the port monitor administrative file.

Note also that additions to both the SAC and port monitor administrative files are made cooperatively, since there must be an entry in **`/etc/saf/_sacadm`** for each *pmtag* associated with a port monitor administrative file. See *"Adding a Port Monitor"* under *"Port Monitor Management"* for details.

The first line of each port monitor administrative file is typically a comment line indicating the version of the port monitor.

Each entry in the port monitor administrative file is a colon (`:`) delimited string that contains the following information:

`SVCTAG` A unique tag that identifies a service. This tag is unique only for the port monitor through which the service is available. Other port monitors may offer the same or other services with the same tag. A service requires both a port monitor tag and a service tag to identify it uniquely.

`SVCTAG` may consist of up to 14 alphanumeric characters.

`FLGS` Flags with the following meanings may currently be included in this field:

`x` Do not enable this port.
By default the port is enabled.

`u` Create an entry for this service in **`/var/adm/utmp`**.
By default no **`utmp`** entry is created for the service.

The **`utmp`** file is used by the **`who`** command, which reports a list of users currently logged in and the ports on which they're working. Note that port monitors may ignore the `u` flag if creating a **`utmp`** entry for the service is not appropriate to the manner in which the service is to be invoked. Some services may not start properly unless **`utmp`** entries have been created for them. For example, services using the login scheme require a **`utmp`** entry.

`ID` The identity under which the service is to be started. The identity has the form of a login name as it appears in **`/etc/passwd.`** If this field is empty, the identity is supplied by the authentication scheme. When an ID and an authentication scheme are both specified, the port monitor performs the authentication under the scheme-supplied identity and invokes the service under the identity specified in the ID field. If neither ID nor authentication scheme is supplied, an error is returned when the service is executed.

`SCHEME` The authentication scheme for the service. If the scheme field is empty, no authentication is done by the port monitor.

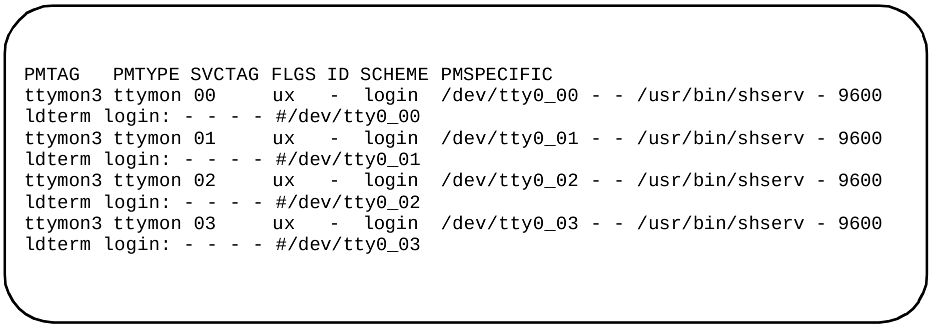
`PMSPECIFIC` Examples of port monitor-specific information are addresses, the name of a process to execute, or the name of a STREAMS pipe to pass a connection through.

Each string comprising an entry in the file is terminated by a pound sign (#), which indicates the start of an optional comment.

Screen 5-2 shows the output of

```
    pmadm -l -p ttymon3
```

invoked to display the list of entries in a sample administrative file, **`ttymon3`**. Note that everything in the `PMSPECIFIC` column is specific to a `ttymon` port monitor. The listing for a **`listen`** administrative file, for example, will contain a different set of entries in this column. Port-monitor specific information is formatted by the port monitor's administrative command, in this case **`ttyadm`**. The **`ttyadm`** command is included as part of the **`pmadm`** command when **`pmadm`** is used to add a port monitor. See *"Adding a Service"* under *"Service Management"*.



```
PMTAG   PMTYPE SVCTAG FLGS ID SCHEME PMSPECIFIC
ttymon3 ttymon 00     ux   -  login  /dev/tty0_00 - - /usr/bin/shserv - 9600
ldterm login: - - - - #/dev/tty0_00
ttymon3 ttymon 01     ux   -  login  /dev/tty0_01 - - /usr/bin/shserv - 9600
ldterm login: - - - - #/dev/tty0_01
ttymon3 ttymon 02     ux   -  login  /dev/tty0_02 - - /usr/bin/shserv - 9600
ldterm login: - - - - #/dev/tty0_02
ttymon3 ttymon 03     ux   -  login  /dev/tty0_03 - - /usr/bin/shserv - 9600
ldterm login: - - - - #/dev/tty0_03
```

Screen 5-2. Output of `pmadm -l -p` ttymon3

CAUTION

To maintain the integrity of the system, it is strongly recommended that changes in the SAC and port monitor administrative files be made with the **`sacadm`** and **`pmadm`** commands, not by editing the files. The SAC does not recognize changes in some of the fields in these files unless they are made using the appropriate administrative command. Editing the file directly can lead to unexpected results.

Port Monitor Management

The Service Access Facility administrative model is hierarchical. The higher level is concerned with port monitor administration. At this level, port monitors may be added, removed, started, stopped, enabled, or disabled. The lower level is concerned with service administration. Functions performed at this level include requesting port monitor status information, replacing a per-system configuration file, installing or replacing a per-port monitor configuration file, and requesting that a port monitor read its administrative file. These functions are discussed under *“Service Management”*.

The SAC Administrative Command sacadm

sacadm is the administrative command for the upper level of the Service Access Facility hierarchy, that is, for port monitor administration. [See **sacadm(1M)**.] Under the Service Access Facility, port monitors are administered by using the **sacadm** command to make changes in the SAC's administrative file. **sacadm** performs the functions listed below. Each function is discussed in one of the following sections.

- print the requested port monitor information from the SAC administrative file
- add or remove a port monitor
- enable or disable a port monitor
- start or stop a port monitor
- install or replace a per-system configuration script
- install or replace a per-port monitor configuration script
- ask the SAC to reread its administrative file

Printing Port Monitor Status Information

```
sacadm -L [ -p pmtag | -t type ]
sacadm -l [ -p pmtag | -t type ]
```

pmtag is the tag associated with the port monitor that is being listed. *type* specifies the port monitor type, for example, **listen**. Unless the system administrator already knows the type of a port monitor, it may be necessary to use the most general form of the command (**sacadm -l**) to find out what the valid type and tag names are.

The **-l** and **-L** options request port monitor status information and may be invoked by any user on the system. The **-l** by itself lists status information for all port monitors on the system. The **-l** option with a **-p** option lists status information for port monitor *pmtag*. A **-l** with **-t** lists status information for all port monitors of type *type*. Any other combination of options with the **-l** option is invalid.

The **-L** option is provided for use in scripts; the format of its output (a condensed version of **-l** output without headers) lends itself well to online parsing, but is much less legible than the output of **-l**.

Options that request information write the requested information to the standard output. A request for information using the **-l** option prints column headers and aligns the information under the appropriate headings. A request for information in the condensed format using the **-L** option prints the information in colon-separated fields. If the **-l** option is used, empty fields are indicated by a hyphen. If the **-L** option is used, empty fields are indicated by two successive colons.

The following sample output shows the differences between some of the options described above. The command

```
sacadm -l
```

lists status information for all port monitors:

```
PMTAG       PMTYPE         FLGS RCNT STATUS     COMMAND
inetd       inetd           -    0    ENABLED    /usr/sbin/inetd #internet dae
mon
tcp         listen          -    3    ENABLED    /usr/lib/saf/listen -m inet/t
cp0 tcp 2>/dev/null #
```

Screen 5-3. Sample Output of sacadm -l (Most General Form of the list Option)

Note that if **ttymon1** is enabled (**sacadm -e -p** ttymon1), the entry in the STATUS field changes from DISABLED to ENABLED, but the entry in the FLGS field does not change. The d flag indicates that the port monitor goes immediately to DISABLED state when it is started. After it has been started, the system administrator can put it in ENABLED state. The flags field conveys information about the state in which a port monitor *starts*, not about its current state.

The command

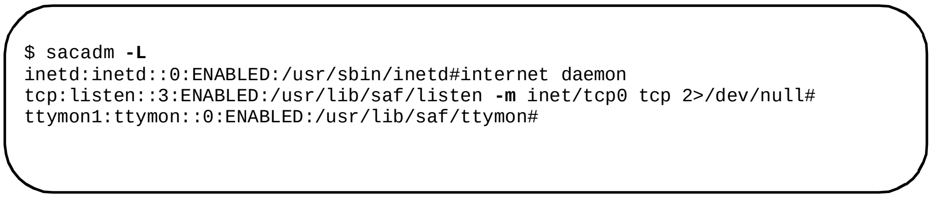
```
sacadm -l -p tcp
```

lists status information only for port monitor tcp:

```
PMTAG    PMTYPE   FLGS   RCNT  STATUS      COMMAND
tcp  listen   d      3     DISABLED    /usr/lib/saf/listen -m tcp #
```

Screen 5-4. Sample Output of sacadm -l When a Port Monitor Is Specified

The same command using **-L** instead of **-l** will produce:



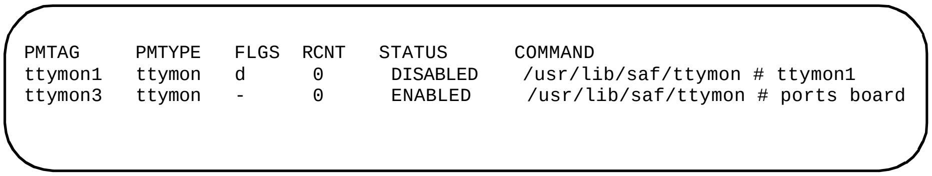
```
$ sacadm -L
inetd:inetd::0:ENABLED:/usr/sbin/inetd#internet daemon
tcp:listen::3:ENABLED:/usr/lib/saf/listen -m inet/tcp0 tcp 2>/dev/null#
ttymon1:ttymon::0:ENABLED:/usr/lib/saf/ttymon#
```

Screen 5-5. Sample of Condensed Output of sacadm -L

The command

```
sacadm -l -t ttymon
```

lists status information for all **ttymon** port monitors.



```
PMTAG     PMTYPE   FLGS  RCNT   STATUS      COMMAND
ttymon1   ttymon   d      0      DISABLED    /usr/lib/saf/ttymon # ttymon1
ttymon3   ttymon   -      0      ENABLED     /usr/lib/saf/ttymon # ports board
```

Screen 5-6. Status Information for Port Monitors of a Single Type.

Adding a Port Monitor

sacadm **-a -p** *pmtag* **-t** *type* **-c** *"cmd"* **-v** *ver* [**-f** dx]\
[**-n** *count*][**-y** *"comment"*][**-z** *script*]

sacadm with a **-a** option is used by the system administrator or by a package that is being installed to create new instances of a port monitor. Because of the complexity of the options and arguments that follow the **-a** option, it may be advisable for the system administrator to use a command script or the menu system to add port monitors. To use the menu system, type **sysadm ports** and then choose the **port_monitors** option.

When **sacadm** creates a port monitor, it creates the supporting directory structure in **/etc/saf** and **/var/saf** for the new port monitor *pmtag* and the port monitor administrative file. It also adds an entry for the new port monitor to the SAC's administrative file.

The options following the **-a** option have the following meanings:

The **-c** option is followed by a command enclosed in double quotes. This is the command the SAC executes to start the port monitor.

The **-v** option is followed by the version number of the port monitor. The version number may be given to **sacadm** by the port monitor's special administrative command, as an argument to the **-v** option. For example:

```
-v `2ttyadm -V`
```

The port monitor-specific command is **ttyadm** for **ttymon** and **nlsadmin** for **listen** [see **ttyadm(1M)** and **nlsadmin(1M)**]. The version stamp of the port monitor is known by the command and is returned when the port monitor administrative command is invoked with the **-V** option. The version number is added to the new administrative file as a comment line of the form

# **VERSION=***value*

where *value* is an integer that represents the port monitor version number. The version number defines the file format. It provides a means of synchronizing software releases of port monitors with their properly formatted administrative files.

The **-f** option specifies one or both of the two flags d and x. The flags have the following meanings:

d	Do not enable the port monitor
x	Do not start the port monitor.

If the **-f** option is not included in the command line, no flags are set and the default conditions prevail. By default, a port monitor is started and enabled.

The **-n** option sets the restart count to *count*. If a restart count is not specified when adding a port monitor, *count* is set to 0. A count of 0 indicates that the port monitor is not to be restarted if it fails.

The **-y** option includes ”*comment*” in the SAC administrative file entry for the port monitor being added.

The **-z** option names a file whose contents are installed as the per-port monitor configuration script, _config.

The command line in Screen 5-7 adds a tcp port monitor of type **listen.**

```
$ sacadm -a -p tcp -t listen -c “/usr/lib/saf/listen -m inet/tcp0 \
tcp 2>/dev/null” -v `nlsadmin -V` -n 3 2>/dev/null
```

Screen 5-7. Adding a listen Port Monitor

```
sacadm -a -p ttymon1 -t ttymon -c “/usr/lib/saf/ttymon” -v `yadm -V`
```

Screen 5-8. Adding a ttymon Port Monitor

Port Monitor sacadm Options

The following options to the **sacadm** command allow you to perform the task indicated. (These options are used in commands which, when received by the SAC, are forwarded to the specified port monitor.)

-e Enable *pmtag.*

-d Disable *pmtag.*

-s Start *pmtag.*

-k Kill *pmtag.* (The SAC sends the signal SIGTERM to *pmtag.*)

NOTE

The above are **sacadm** options, not command lines. For a description of the complete command line syntax, see **sacadm(1M)**.

Removing a Port Monitor

To remove port monitor *pmtag* from the system, execute:

```
sacadm -r -p pmtag
```

The port monitor entry is removed from the SAC's administrative file and the SAC rereads the file. If the removed port monitor is not running, no further action is taken. If the removed port monitor is running, the Service Access Controller sends it SIGTERM to indicate it should shut down. Note that the port monitor's directory structure remains intact but is no longer referenced by anything.

Configuration Scripts for Systems and Port Monitors

Configuration scripts tailored to individual systems and port monitors are administered using **sacadm**; configuration scripts written for individual services are administered using **pmadm** and are described under *"Service Management"*. Configuration scripts specific to systems and port monitors allow a system administrator to modify the system and port monitor environments. They are written in the interpreted language described on the **doconfig(3I)** online manual page and in *Network Programming Interfaces.* Sample configuration scripts are shown below.

The system-specific configuration script _sysconfig, is interpreted when the SAC is starting. A port monitor's per-port monitor configuration script is interpreted by the SAC just before the SAC starts the port monitor.

Configuration scripts specific to systems and port monitors may be printed by any user on the system. Only the system administrator may install or replace them.

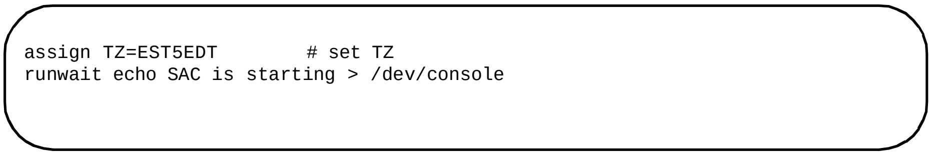
Configuration Scripts for Individual Systems

sacadm **-G** [**-z** *script*]

The system-specific configuration script **/etc/saf/_sysconfig** customizes the environment for all services on the system. When it starts up, the Service Access Controller interprets the system-specific configuration script, using the doconfig library routine. A default **_sysconfig** containing only a comment line is part of the delivered system.

The **-G** option is used to print or replace the system-specific configuration script. The **-G** option by itself prints the system-specific configuration script. The **-G** option in combination with a **-z** option replaces **/etc/saf/_sysconfig** with the contents of the file *script*. Other combinations of options with a **-G** option are invalid.

The **_sysconfig** file in the figure sets the time zone variable, TZ.



```
assign TZ=EST5EDT        # set TZ
runwait echo SAC is starting > /dev/console
```

Screen 5-9. Sample System-specific Configuration Script

NOTE

The **-z** option is also used with the **-a** option to specify the contents of the port monitor-specific configuration file when a port monitor is created.

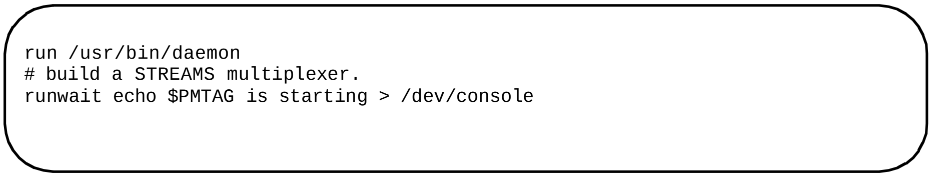
Configuration Scripts for Individual Port Monitors

sacadm **-g -p** *pmtag* [**-z** *script*]

The port monitor-specific configuration script **/etc/saf/***pmtag*/_config customizes the environment for services that are available through the specific collection of access points for which port monitor *pmtag* is responsible. When the SAC starts a port monitor, the port monitor-specific configuration script is interpreted, if it exists, using the **doconfig(3I)** library routine.

The **-g** option is used to print, install, or replace a port monitor-specific configuration script. A **-g** option requires a **-p** option. The **-g** option with only a **-p** option prints the port monitor-specific configuration script for port monitor *pmtag*. When run with the **-p** and **-z** options, the **-g** option installs the file *script* as the port monitor-specific configuration script for port monitor *pmtag*, or, if **/etc/saf/***pmtag*/_config exists, it replaces **_config** with the contents of *script*. Other combinations of options with **-g** are invalid.

In the hypothetical **_config** file in Screen 5-10, the command **/usr/bin/daemon** is assumed to start a daemon process that builds and holds together a STREAMS multiplexer. By installing this configuration script, the command can be executed just before starting the port monitor that requires it.



```
run /usr/bin/daemon
# build a STREAMS multiplexer.
runwait echo $PMTAG is starting > /dev/console
```

Screen 5-10. Sample Port Monitor-Specific Configuration Script

Reading the Administrative Files

`sacadm` **`-x`** `[`**`-p`** *pmtag*`]`

When changes are made to the SAC's administrative file, the SAC needs to be notified of the change. When changes are made to a port monitor's administrative files, the port monitor needs to be notified. When **sacadm** and **pmadm** are used to make changes, this notification takes place automatically. If the files are edited by the system administrator, the SAC and the port monitors are not notified. In this case, **sacadm** must be called with the **-x** option to notify the SAC or port monitor of the changes.

sacadm with the **-x** option tells the SAC to update its internal copy of the information in the SAC administrative file. **sacadm** with the **-x** and **-p** options causes the SAC to send a READ message to the designated port monitor, *pmtag*.

System administrators are advised against editing these files directly.

Service Management

The top level of the Service Access Facility is concerned with port monitor administration and is discussed in *"Port Monitor Management"*. The lower level is concerned with service administration and is discussed in this section.

At this level there are two distinct administrative functions. The first is the administration of the port itself. The information needed to administer a port is found on the online manual page for **ttymon's** port monitor-specific command, **ttyadm(1M)**. The information needed to administer a network address monitored by a **listen** port monitor is found on the online manual page for **listen's** port monitor-specific command, **nlsadmin(1M)**.

The second level is the administration of the service associated with a port. By definition, there is only one service associated with a port. All ports on the system are peers and their services are administered through the same command interface, the Service Access Facility's administrative command **pmadm(1M).** At the level of service administration, services may be added, removed, enabled, and disabled. Other functions performed at this level include installing, removing, or replacing a service-specific configuration script, requesting service status information, and specifying the authentication scheme to be used for the service. Identification and authentication schemes are discussed in *Network Administration.*

The Port Monitor Administrative Command pmadm

`pmadm` is the administrative command for the lower level of the Service Access Facility hierarchy, that is, for service administration. A port may have only one service associated with it although the same service may be available through more than one port. By executing **`pmadm`** you can

- print information from the port monitor's administrative file
- add or remove a service
- enable or disable a service
- install, remove, or replace a per-service configuration script
- specify an authentication scheme

Note that in order to identify an instance of a service uniquely, the **`pmadm`** command must identify both the service (**`-s`**) and the port monitor or port monitors through which the service is available (**`-p`** or **`-t`**).

Printing Service Status Information

```
pmadm -l [ -t type | -p pmtag1 ] [ -s svctag ]
pmadm -L [ -t type | -p pmtag ] [ -s svctag ]
```

The **`-l`** and **`-L`** options request service status information. They may be invoked by any user on the system. Used either alone or with the options described below they provide a filter for extracting information in several different groupings.

`-l` By itself, the **`-l`** option lists status information for all services on the system.

`-l -p` *pmtag* Lists status information for all services available through port monitor *pmtag*.

`-l -s` *svctag* Lists status information for all services with the tag *svctag* available through any port monitor on the system.

`-l -p` *pmtag* **`-s`** *svctag* Lists status information for service *svctag* available through port monitor *pmtag*.

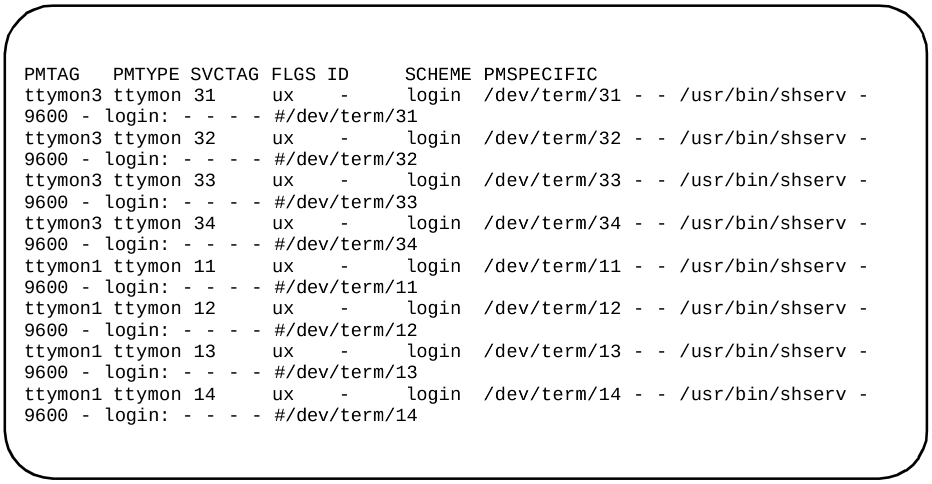
`-l -t` *type* Lists status information for all services available through port monitors of type *type*.

`-l -t` *type* **`-s`** *svctag* Lists status information for all services with the tag *svctag* offered through a port monitor of type *type*.

Other combinations of options with **`-l`** are invalid.

The **`-L`** option is identical to the **`-l`** option except that output is printed in a condensed format.

Options that request information write the requested information to the standard output. A request for information using the **`-l`** option prints column headers and aligns the information under the appropriate headings. A request for information in the condensed format using the **`-L`** option prints the information in colon-separated fields. If the **`-l`** option is used, empty fields are indicated by a hyphen. If the **`-L`** option is used, empty fields are indicated by two successive colons.



```
PMTAG   PMTYPE SVCTAG FLGS ID     SCHEME PMSPECIFIC
ttymon3 ttymon 31     ux    -     login  /dev/term/31 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/31
ttymon3 ttymon 32     ux    -     login  /dev/term/32 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/32
ttymon3 ttymon 33     ux    -     login  /dev/term/33 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/33
ttymon3 ttymon 34     ux    -     login  /dev/term/34 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/34
ttymon1 ttymon 11     ux    -     login  /dev/term/11 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/11
ttymon1 ttymon 12     ux    -     login  /dev/term/12 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/12
ttymon1 ttymon 13     ux    -     login  /dev/term/13 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/13
ttymon1 ttymon 14     ux    -     login  /dev/term/14 - - /usr/bin/shserv -
9600 - login: - - - - #/dev/term/14
```

Screen 5-11. Output of `pmadm -l`

Adding a Service

> `pmadm` **`-a`** `[` **`-p`** *pmtag* **`|`** **`-t`** *type* `]` **`-s`** *svctag* `[` **`-i`** *id* `]` **`-m`** **`\`**
> *"pmspecific"* **`-v`** *ver* `[`**`-f`** `xu] [`**`-S`** *"scheme"*`][`**`-y`** *"comment"*`]\`
> `[`**`-z`** *script* `]`

`pmadm` with an **`-a`** option adds a service by making an entry for the new service in the port monitor's administrative file. It is important to be aware that a service implies a port and that there is a one-to-one mapping between ports and instances of services. Because of the complexity of the options and arguments that follow the **`-a`** option, it may be advisable to use a command script or the menu system to add services. If you use the menu system, type **`sysadm ports`** and then choose the **`port_services`** option.

The following paragraphs describe the components of the command line for adding a service.

`-p` specifies the tag associated with the port monitor through which a service (specified as **`-s`** *svctag*) is available. *pmtag* and *svctag* are names chosen by the system administrator. When a service is added, the command line must contain either a **`-p`** or a **`-t`** option.

The **-t** option specifies the port monitor type. Port monitors are specified either by a **-t** or by a **-p** but not both. If **-p** is used, a service is added to a single port monitor, port monitor *pmtag*. If **-t** is used, instances of a service are added to all port monitors of type *type*.

The **-s** option specifies the service tag.

The **-i** option specifies the identity that is to be assigned to the service when it is started. *id* must be an entry in **/etc/passwd.** The **-i** argument is optional when a service is being added. If the **-i** option is omitted, the port monitor determines the user ID from information supplied by an authentication scheme (see **-S**, below). If the **-i** option is omitted and no authentication scheme is specified, an error is returned when the service is executed. When the user ID is specified using **-i** and an authentication scheme is also specified, the port monitor performs the authentication using the scheme-supplied identity. The identity specified by the **-i** option takes precedence when the service is invoked.

The **-m** option allows port monitor-specific options to be included on the **-a** command line. This information should be generated by using a port monitor-specific command, with whichever of its options are appropriate.

The **-v** option specifies the version number of the port monitor administrative file. For a port monitor of type **listen**, for example, the version number may be given as

```
-v `sadmin -V`
```

The port monitor-specific command for **ttymon** is **ttyadm(1M)**. The port monitor-specific command for **listen** is **nlsadmin(1M)**. The version stamp of the port monitor is known by the port monitor-specific command and is returned when the command is invoked with a **-V** option.

The **-f** option specifies one or both of two flags which are then included in the flags field of the port monitor administrative file entry for the new service. The flags have these meanings:

x	Do not enable the service.
u	Create a **utmp** entry for the service.

If the **-f** option is not included on the **-a** command line, no flags are set and the default conditions prevail. By default, a new service is enabled and no **utmp** entry is created for it.

The **-S** option specifies the authentication scheme to be associated with the service. An authentication scheme verifies that a potential user is authorized to log in on the system. A password for a login name is an example of an authentication scheme name.

scheme may be a simple authentication scheme name or an authentication scheme command line with arguments. If *scheme* includes arguments or options, it must be enclosed in double quotes.

-y precedes a comment enclosed in double quotes. *comment* is included in the comment field for the service entry in the port monitor administrative file.

-z installs *script* as a configuration file.

The example adds a service with service tag 105 to all port monitors of type **listen.**

```
pmadm -a -s 105 -t listen -i root -v `nlsadmin -V` \
      -m `nlsadmin -a 105 -c /usr/net/servers/rfsetup`
```

Enabling or Disabling a Service

```
pmadm -e -p pmtag -s svctag
pmadm -d -p pmtag -s svctag
```

pmadm with the **-e** option enables a service. x is removed from the flags field in the entry for service *svctag* in the port monitor administrative file.

The **-d** option disables a service. x is added to the flags field in the entry for service *svctag* in the port monitor administrative file.

Removing a Service

```
pmadm -r -p pmtag -s svctag
```

pmadm with a **-r** removes service *svctag*. The entry for the service is removed from the port monitor administrative file.

Authentication Schemes and User IDs

The **pmadm** command may be used to change or remove authentication schemes and user IDs.

```
pmadm -c -S “scheme” [-i id] -p pmtag -s svctag
pmadm -c -i id [-S “scheme”] -p pmtag -s svctag
```

Used with the **-c** option, the **-S** and **-i** options manipulate the contents of the *scheme* and *id* fields in the port monitor administrative file. For example,

```
pmadm -c -S “cr1 -suucico” -p tcp -s uucico
```

If either *scheme* or *id* is the NULL string, the corresponding field will be empty and the authentication scheme or user ID will be effectively removed from the service line in the file. For example,

```
pmadm -c -S ““ -p ttymon1 -s 11
```

will remove the authentication scheme for service 11 from the administrative file for port monitor **ttymon1.**

For a given service, there may be non-NULL entries in either of these fields, in both fields, or in neither. Since authentication schemes can provide a user ID, it is important to understand when the user ID determined by the authentication scheme is used and when the user ID specified in the *id* field is used. The following table describes the four possible cases.

scheme	*id*	Description
specified	`NULL`	Authentication is performed by the port monitor using the specified scheme. If authentication succeeds, the service is started with the ID determined by the scheme. If authentication fails, the service is not started.
`NULL`	specified	No authentication is performed by the port monitor. The service is started with the ID specified in the *id* field.
specified	specified	The port monitor invokes the specified authentication scheme. If authentication succeeds, the service is started with the ID from the *id* field. If authentication fails, the service is not started.
`NULL`	`NULL`	This is an error. The service will not start.

Configuration Scripts for Specific Services

```
pmadm -g -p pmtag -s svctag [ -z script ]
pmadm -g -s svctag -t type    -z script
```

Configuration scripts for individual services are command scripts written in the interpreted language described in the **doconfig(3I)** online manual page and in the *Network Programming Interfaces.* They allow the system administrator to modify the environment in which a service is executed. For example, the values of environment variables may be changed, STREAMS modules may be specified, or commands may be run.

Service-specific configuration scripts are interpreted by the port monitor before the service is invoked.

NOTE

> The SAC interprets both its own configuration file, **`_sysconfig`**, and the port monitor configuration files. Only service-specific configuration files are interpreted by the port monitors.

Service-specific configuration scripts may be printed by any user on the system. Only a system administrator may install or replace them.

The **`-g`** option is used to print, install, or replace a service-specific configuration script. The **`-g`** option with a **`-p`** option and a **`-s`** option prints the service-specific configuration script for service *svctag* available through port monitor *pmtag*. The **`-g`** option with the **`-p`** option, the **`-s`** option, and the **`-z`** option installs the service-specific configuration script contained in the file *script* as the service-specific configuration script for service *svctag* available through port monitor *pmtag*. The **`-g`** option with the **`-s`** option, the **`-t`** option, and the **`-z`** option, installs the file *script* as the service-specific configuration script for service *svctag* available through any port monitor of type *type*. Other combinations of options with **`-g`** are invalid.

The following service-specific configuration script does two things:

- It specifies the maximum file size for files created by a process by setting the process's `ulimit` to 4096.
- It specifies the protection mask to be applied to files created by the process by setting `umask` to 077.

```
runwait ulimit 4096
runwait umask 077
```

Screen 5-12. Sample Configuration Script for an Individual Service

The Port Monitor ttymon

`ttymon` is a port monitor invoked by the Service Access Controller (SAC). The Service Access Controller is the Service Access Facility's controlling process. It is started by **`init`** when the system enters multi-user state. One of the SAC's functions after it is started is to start all port monitors the system administrator has configured.

Beginning with the operating system, **`ttymon`** performs the functions that **`getty`** and **`uugetty`** performed in previous releases. Like **`getty`** and **`uugetty, ttymon`** sets terminal modes and line speeds for the port the user is connected to, allowing communication with the service associated with that port.

ttymon differs from **getty** and **uugetty** in several important ways:

- **ttymon** provides any service (such as the shell or a database) configured by the system administrator. **getty** and **uugetty** provided only login service.
- Each invocation of **ttymon** can support multiple ports. **getty** and **uugetty** supported only one port per invocation.
- **ttymon** is a persistent process that continues to run after the service process is initiated. The **getty** and **uugetty** processes were replaced by the process of the service invoked.
- **ttymon** can take advantage of all STREAMS I/O capabilities.
- Line disciplines are configurable on a per-port basis.
- **ttymon** provides an optional autobaud facility that automatically determines the line speed of the hardware connected to any port monitored by a **ttymon** port monitor.

What ttymon Does

ttymon has the following functions:

- It initializes and monitors terminal ports.
- It sets terminal modes and line speeds for each port it monitors.
- It identifies and authenticates users.
- It invokes the service associated with a given port whenever it receives a connection request on that port.

Each instance of **ttymon** has its own administrative file that specifies the ports to monitor and the services associated with each port. The file contains a *ttylabel* field that refers to a speed and TTY definition in the **/etc/ttydefs** file. See **ttyadm(1M)** for a description of the information specific to **ttymon** that is contained in a **ttymon** administrative file.

When **ttymon** is started, it initializes all ports specified in its administrative file. First, it constructs a tp (trusted path) multiplexer which splits access to the physical port into two channels. One channel provides login access to the user; the other is used by **ttymon**. The multiplexer provides security protection not previously possible. **ttymon** then pushes the specified STREAMS modules on the user-accessible channel, sets speed and initial **termio(7)** settings, and writes the prompt. It then waits for user input.

If the user indicates that the speed is inappropriate by pressing the BREAK key, **ttymon** hunts to the next *ttylabel* in the **/etc/ttydefs** file [see **ttydefs(4)**], adjusts termio values, and writes the prompt again. When valid input is received, that is, one or more non-break keys followed by a new line, **ttymon** interprets the per-service configuration file for the port, if one exists, invokes the identification and authentication scheme specified in the port monitor's administrative file, updates the **utmp** and **wtmp** files, sets and initializes the environment variables that create the service environment, and invokes a service. The **login** authentication scheme is the scheme most commonly associated with **ttymon** [see **login(1)**]. The service invoked will normally be **/usr/bin/shserv**, which will invoke the shell specified in the **/etc/passwd** file.

The Autobaud Option

Autobaud allows the system to set the line speed of a given TTY port to the line speed of the device connected to the port without the user's intervention. Each time a service to be monitored by a **ttymon** port monitor is added, a *ttylabel* must be supplied (see *“Adding a Service”*, below). If this *ttylabel* points to an entry in the **/etc/ttydefs** file that has an “A” in the autobaud field, **ttymon** will try to determine the proper line speed before printing the prompt.

After receiving a carrier-indication on one of its TTY ports, but before printing a prompt, **ttymon** does the following:

- **ttymon** reads the next character received from the port. Provided the character read is a newline character and that it is transmitted at a line speed autobaud can support, **ttymon** will reliably determine this line speed and change the port's line speed to that speed.
- If a baud rate cannot be determined from the character that is read (for example, if the user entered a character other than a newline), or if a break is received rather than a character, **ttymon** considers this to be an autobaud failure and the character is discarded. If after five opportunities, a newline is not recognized, the search proceeds to the next **ttydefs** entry in the hunt sequence. If an autobaud flag is encountered again, the prompt will not be written and the procedure just described is repeated. If no autobaud flag is set, the search again proceeds to the next **ttydefs** entry in the hunt sequence.

ttymon and the Service Access Facility

The Service Access Facility (SAF) provides a generic interface to which all port monitors must conform. **ttymon** is a port monitor under the Service Access Facility's controller, the Service Access Controller. (See *“Overview of the Service AccessFacility”*, *“Port Monitor Management”*, and *“Service Management”*, for a description of the Service Access Facility, the administrative files it maintains, and the commands used for port monitor and service administration.) Figure 5-1 shows how a service, which is usually a shell service, is invoked using **ttymon**.

There can be multiple invocations of **ttymon** port monitors, each identified by a unique *pmtag*. Each of these port monitors can monitor multiple ports for incoming connection requests.

A port has one and only one service associated with it. Each port, and its associated service, is identified by a service tag, *svctag*. Service tags for any given port monitor are unique.

When the Service Access Controller starts a port monitor, the port monitor reads its administrative file, which contains information about which ports to monitor and what service (that is, process) is associated with each port.

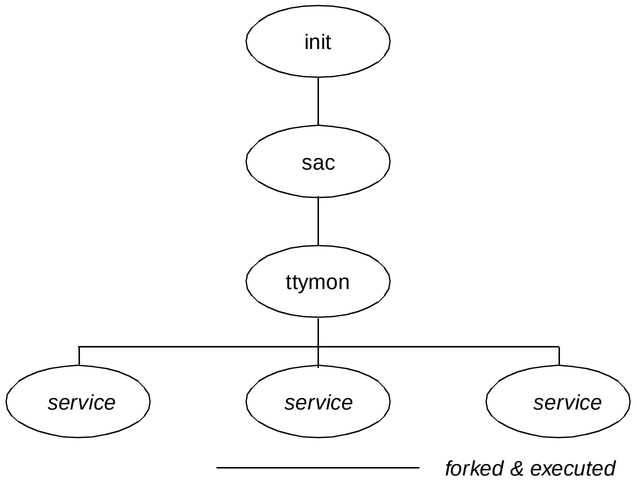


Figure 5-1. TTY Service Invocation

The Default ttymon Configuration

Some **ttymon** port monitors may be set up automatically when the system goes to multi-user mode. To find out if your system has been automatically configured, enter the command

```
sacadm -l
```

after the system is in multi-user mode. To see a listing of all services available under the configured **ttymon** port monitors, enter the command

```
pmadm -l -t ttymon
```

Services are not defined for the console port under any **ttymon** port monitor. Instead, there is an entry for it in the **/etc/inittab** file. This entry contains a call to **ttymon** in "express" mode. (See "**ttymon** Express," below.)

The ttyadm Command

The Service Access Facility requires each type of port monitor to provide an administrative command. This command must format information derived from command-line options so that it is suitable for inclusion in the administrative files for that port monitor type. The command may also perform other port monitor-specific functions.

ttyadm is **ttymon**'s administrative command. The **ttyadm** command formats information based on the options with which it is invoked and writes this information to the standard output.

The output of **ttyadm** is one of the arguments used by **pmadm -a** to format information in a way suitable for inclusion in a **ttymon** administrative file. **ttyadm** presents this information (as standard output) to **pmadm**, which places it in the file. This use of **ttyadm** is described below under *“Adding a ttymon Port Monitor.” pmspecific* information in a port monitor administrative file will be different for different port monitor types.

`ttyadm` is also included on the **`sacadm`** command line when a port monitor is added to the system. It is used to supply the **`ttymon`** version number for inclusion in a port monitor's administrative file.

NOTE

The port monitor administrative file is updated by the Service Access Facility's administrative commands, **`sacadm`** and **`pmadm`**. **`ttyadm`** merely provides a means of presenting formatted port monitor-specific (that is, **`ttymon`**-specific) data to these commands.

The **`sacadm`** command line uses **`ttyadm`** only with the **`-V`** option. **`ttyadm -V`** returns the version number of the **`ttymon`** command being used.

Managing TTY Ports

Finding Out Which ttymon Port Monitors Are Configured

sacadm **-l** [**-p** *pmtag* | **-t** *type*]

The **`sacadm`** command with only a **`-l`** option lists all port monitors currently defined for the system. The following is an example of its output:

```
$ sacadm -l
PMTAG         PMTYPE       FLGS RCNT STATUS     COMMAND
inetd         inetd         -    0    ENABLED    /usr/sbin/inetd #internet
daemon
tcp           listen       -    3    ENABLED    /usr/lib/saf/listen  -m inet/
tcp0 tcp #
ttymon1       ttymon       -    0    ENABLED    /usr/lib/saf/ttymon #
ttymon3       ttymon       -    0    ENABLED    /usr/lib/saf/ttymon #
```

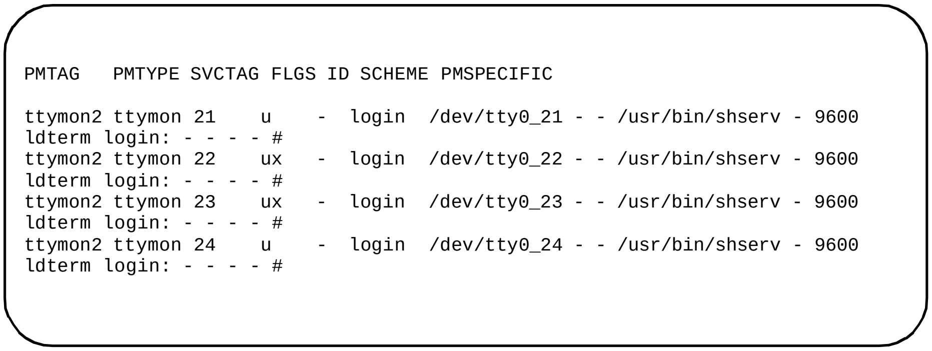
`sacadm` can also be used to list a single port monitor (**`-p`**) or to list only port monitors of a single type (**`-t`**), for example, all port monitors of type **`ttymon`**. For a complete description of these options, see *“Printing Port Monitor Status Information”* (under *“Port Monitor Management”*) or see **`sacadm(1M)`**.

Finding Out Which Services Are Configured for a ttymon Port Monitor

pmadm **-l** [**-p** *pmtag* | **-t** *type*] [**-s** *svctag*]

pmadm with only a **-l** will list all services for all port monitors on the system. If a port monitor is specified (**-p**), all services for that port monitor will be listed. The following is a sample listing for the command

```
pmadm -l -p ttymon2
```



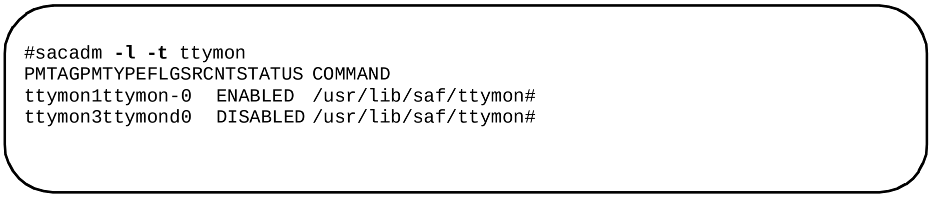
```
PMTAG   PMTYPE SVCTAG FLGS ID SCHEME PMSPECIFIC

ttymon2 ttymon 21    u    -  login  /dev/tty0_21 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
ttymon2 ttymon 22    ux   -  login  /dev/tty0_22 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
ttymon2 ttymon 23    ux   -  login  /dev/tty0_23 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
ttymon2 ttymon 24    u    -  login  /dev/tty0_24 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
```

In the above table, the *pmspecific* fields include the device (for example, **/dev/term/21**), the service to be invoked (**/usr/bin/shserv**), and the prompt (login:). See the **ttyadm(1M)** online manual page for a description of the *pmspecific* fields.

Finding Out Which TTY Ports Are Accessible

To find out which ports are accessible to users, first identify all enabled **ttymon** port monitors:

```
sacadm -l -t ttymon
```



```
#sacadm -l -t ttymon
PMTAGPMTYPEFLGSRCNTSTATUS COMMAND
ttymon1ttymon-0  ENABLED  /usr/lib/saf/ttymon#
ttymon3ttymond0  DISABLED /usr/lib/saf/ttymon#
```

In the listing, port monitor **ttymon1** is enabled. This means that it is accepting service requests for any of its services that are enabled.

To identify which services are enabled, run

```
pmadm -l -p ttymon1
```

This will list all configured TTY services for port monitor **ttymon1**.

```
#pmadm -l -p ttymon1
PMTAG   PMTYPE SVCTAG FLGS ID SCHEME PMSPECIFIC
ttymon1 ttymon 11    u    -  login  /dev/term/11 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
ttymon1 ttymon 12    ux   -  login  /dev/term/12 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
ttymon1 ttymon 13    u    -  login  /dev/term/13 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
ttymon1 ttymon 14    ux   -  login  /dev/term/14 - - /usr/bin/shserv - 9600
ldterm login: - - - - #
```

In the listing, enabled services are those that do *not* have an `x` in the `FLGS` column. The ports corresponding to these services (**`/dev/term/11`** and **`/dev/term/13)`** are accessible to users.

NOTE

On the operating system **`who -l`** lists all running port monitors, not the accessible TTY ports. Follow the procedure described above to find out which TTY ports are accessible.

Adding a ttymon Port Monitor

```
sacadm -a -p pmtag -t type -c "cmd" -v `2pmspecific -V`e
   -n count [ -f dx ] [ -z script ] [ -y "comment" ]
```

The following command line will add a **`ttymon`**-type port monitor named **`ttymon1`**:

```
sacadm -a -p ttymon1 -t ttymon -c /usr/lib/saf/ttymon \
       -v `yadm -V`
```

The command adds a line to the SAC's administrative file. The options that may be used with **`sacadm -a`** are described under *"Port Monitor Management"* and in the **`sacadm(1M)`** and **`ttyadm(1M)`** online manual pages.

Removing a ttymon Port Monitor

```
sacadm -r -p pmtag
```

The following command line removes the port monitor added in the previous example:

```
sacadm -r -p ttymon1
```

The SAC removes the line for port monitor **`ttymon1`** from its administrative file. The port monitor directory will remain in **`/etc/saf`** but will be removed and recreated when a new port monitor with the same name is added. (If a service is already defined for the existing port monitor, it will be suspended unless it's subsequently provided through another port monitor.)

To make changes to a port monitor entry, always remove the entry and add a new entry using the **sacadm** command. Do not edit the SAC administrative file.

Adding a Service

```
pmadm -a -p pmtag -s svctag [ -i id ] [ -S “scheme” ] \
     -f ux ] \ -v `ttyadm -V`e
      -m “`ttyadm [ -b ] [ -r count] [ -c ] [ -h ] \
          [-i msg] [-m modules] [-p prompt] [-t timeout] \
           -d device -l ttylabel -s service`“
```

The following command line adds a shell service with **login** as the authentication scheme to be monitored by the **ttymon** port monitor **ttymon2**:

```
pmadm -a -p ttymon2 -s 21 -S login -fu -v `ttyadm -V` \
      -m “`ttyadm -d /dev/term/21 -l 9600 \
      -s /usr/bin/shserv -m ldterm -p ” tty21:”`“
```

The options that may be used with **pmadm -a** are described in *“Service Management”* and on the **pmadm(1M)** and **ttyadm(1M)** online manual pages. If the Enhanced Security Utilities are installed, the Secure Attention Key (SAK) must be defined before a port is accessible. See **ttyadm(1M)** and *“The Secure Attention Key”* in Chapter 15, “Administering Printers, Terminals and Services” of the System Administration book.

The **ttyadm -m** option may be used for pushing STREAMS modules (for example the line discipline module) ldterm. If autopush has pushed modules on the stream, **ttymon** pops them before pushing its own.

By using the **ttyadm -i** option, we could also have specified a message to be printed whenever someone tries to log in on a disabled port.

The following command defines a service that permits both incoming and outgoing calls. The service is put under port monitor **ttymon2**. The **-b** option defines the port as bi-directional.

```
pmadm -a -p ttymon2 -s 21 -S login -fu -v `ttyadm -V` \
    -m “`ttyadm -b -h -r0 -t 60 -d /dev/term/21 \
    -l 9600H -s /usr/bin/shserv -m ldterm -p ” tty21:”`“
```

The **ttyadm -r** option with count=0 is assumed when the **ttyadm -b** bi-directional option is used; the **-r0** could therefore have been omitted.

NOTE

The Basic Networking Utilities package must be installed for bi-directional services.

For example, you may want to use **cu** with the cr1 authentication scheme (instead of the login scheme). To do so, you must first set up **ttymon** with the following command:

```
pmadm -a -p ttymon -s 22 -v `ttyadm -C` -f u -S “cr1 -s cu” \
-m “`yadm -b -c -t 60 -d \
/dev/term/22 -l 9600H -s /usr/bin/shserv -m ldterm -p ” tty22:”`s+1
```

Removing a Service

```
pmadm -r -p pmtag -s svctag
```

The following example deletes the service that was added in the previous example.

```
pmadm -r -p ttymon2 -s 21
```

Authentication Schemes and User IDs

The **pmadm** command may be used to change or remove authentication schemes and user IDs.

```
pmadm -c -S “scheme” [-i id] -p pmtag -s svctag
pmadm -c -i id [-S “scheme”] -p pmtag -s svctag
```

Used with the **-c** option, the **-S** and **-i** options manipulate the contents of the *scheme* and *id* fields in the port monitor administrative file. If either *scheme* or *id* is the NULL string, the corresponding field will be empty and the authentication scheme or user ID will be effectively removed from the service line in the file. For example,

```
pmadm -c -S ““ -p ttymon -s 21
```

will remove the authentication scheme for service 21 under the **ttymon2** port monitor.

For a given service, there may be non-NULL entries in either of these fields, in both fields, or in neither. It is important to understand when the user ID determined by an authentication scheme is used and when the user ID specified in the *id* field is used. The following table shows the four possible cases.

scheme	*id*	Description
specified	`NULL`	Authentication is performed by the port monitor using the specified scheme. If authentication succeeds, the service is started with the ID determined by the scheme. If authentication fails, the service is not started.
`NULL`	specified	No authentication is performed by the port monitor. The service is started with the ID specified in the *id* field.
specified	specified	The port monitor invokes the specified authentication scheme. If authentication succeeds, the service is started with the ID from the *id* field. If authentication fails, the service is not started.
`NULL`	`NULL`	This is an error. The service will not start.

Enabling a Service

```
pmadm -e -p pmtag -s svctag
```

To enable a service on a specific port, first find out which port monitor is monitoring the port. Enter

```
pmadm -l -t ttymon
```

This lists all services defined for ttymon-type ports.

Now look in the PMSPECIFIC column for the device file that corresponds to the port you are interested in, for example, **/dev/term/23.** If the port monitor is **ttymon2** and the service tag is 23, the command

```
pmadm -e -p ttymon2 -s 23
```

will enable the service on port **/dev/term/23.**

To verify that the port has been enabled, enter

```
pmadm -l -p ttymon2 -s 23
```

The `x` will have been removed from the FLGS column in the entry for this service. If the Enhanced Security Utilities are installed, you will have to define a Secure Attention Key (SAK) before a port is available. See *"The Secure Attention Key"* in Chapter 15, "Administering Printers, Terminals, and Devices" of the System Administration book.

Disabling a Service

```
pmadm -d -p pmtag -s svctag
```

When a service is disabled, all subsequent connection requests for the service will be denied. Using the same example,

```
pmadm -d -p ttymon2 -s 23
```

will disable the service and restore the `x` to the **`FLGS`** field in the entry for service 23.

Disabling All Services Monitored by a ttymon Port Monitor

```
sacadm -d -p pmtag
```

To disable all services defined for the port monitor **`ttymon2`**, enter

```
sacadm -d -p ttymon2
```

Any future connection requests for services managed by this port monitor will be denied until the port monitor is enabled.

The command

```
sacadm -e -p ttymon2.
```

will re-enable port monitor **`ttymon2`**.

ttymon "Express"

Services are not defined for the console port under any **`ttymon`** port monitor. Instead, there is an entry for it in the **`/etc/inittab`** file. This entry contains a call to **`ttymon`** in "express" mode. **`ttymon`** express is a special mode of **`ttymon`** that permits **`ttymon`** to be invoked directly. It uses the **`login`** authentication scheme and starts the **`/usr/bin/shserv`** service. **`ttymon`** in express mode is not managed by the Service Access Controller nor is an administrative file associated with any invocation of **`ttymon`** in this mode.

`ttymon` express is described in greater detail on the **`ttymon(1M)`** online manual page under the heading *"Invoking a Stand-Alone ttymon Process"*.

Configuration Files

As a port monitor under the Service Access Facility, **ttymon** can customize the environment of each service it starts. It does this by interpreting a per-service configuration script, if one exists, immediately before starting the service. Per-service configuration scripts are optional. Configuration scripts are installed by the system administrator, using the **pmadm** command with **-g** and **-z** options. [See **pmadm(1M)**].

It is also possible to customize the environment of a **ttymon** port monitor. A per-port monitor configuration script is defined using the **sacadm** command with **-g** and **-z** options. [See **sacadm(1M)**]. The environment modifications made by a port-monitor configuration script are inherited by the port monitor and all the services it invokes. The environment of any particular service can then be customized further by using a per-service configuration script.

The **doconfig(3I)** online manual page describes the language in which configuration scripts are written.

Configuration scripts are not normally needed for basic operations. You may use a configuration script if you want to customize your environment, debug a service, and so on.

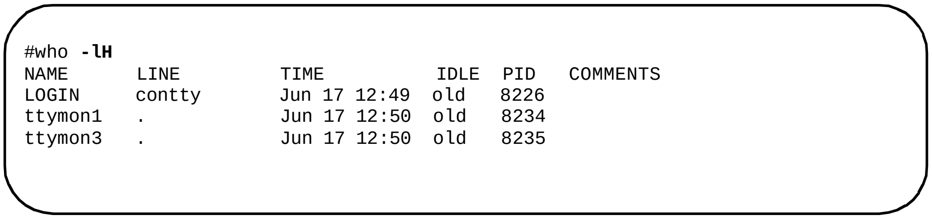
For example, suppose your system has a `ulimit` of 1MB; that is, you can't create files larger than 1MB. You want to send a file larger than 1MB to the system via **uuto,** but **uuto** rejects the file. To solve the problem, you can define a configuration script (through the **pmadm** command) that specifies a larger `ulimit` for the **uucp** service.

The who Command

The **who** command examines the **/var/adm/utmp** file. It's used to find out who is on the system. The command

```
who -lH
```

lists all `RUNNING` port monitors. When **ttymon** in express mode is monitoring a line, the name field is `LOGIN` as it is in the entry for `contty` in the following example.



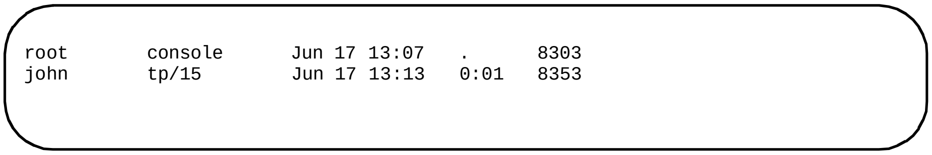
```
#who -lH
NAME      LINE         TIME          IDLE  PID   COMMENTS
LOGIN     contty       Jun 17 12:49  old   8226
ttymon1   .            Jun 17 12:50  old   8234
ttymon3   .            Jun 17 12:50  old   8235
```

Screen 5-13. who -lH Output

The command

```
who -u
```

lists all users who are currently logged in.



```
root       console      Jun 17 13:07   .      8303
john       tp/15        Jun 17 13:13   0:01   8353
```

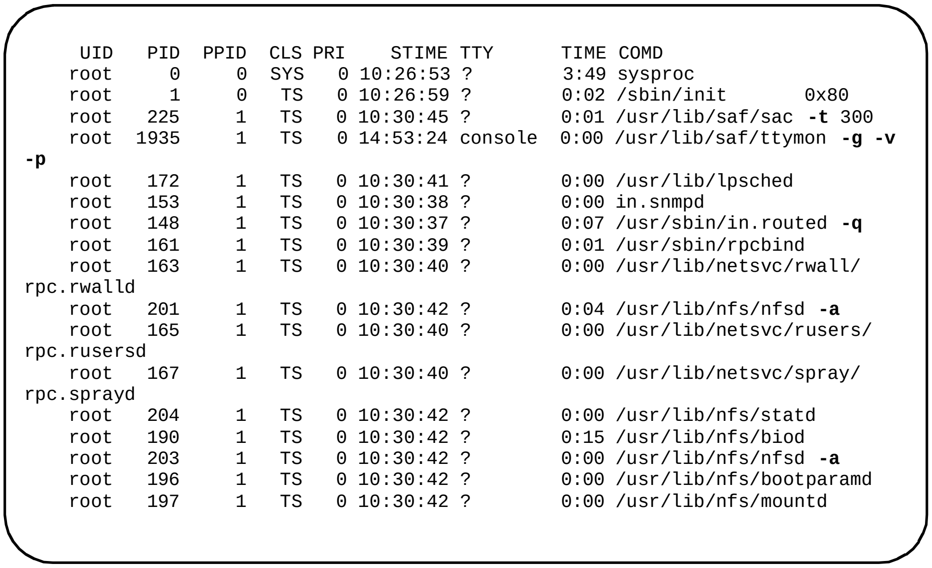
Screen 5-14. `who -u` Output

NOTE

Because the connection is multiplexed, only the tp (trusted path) device accessible to the user's shell is displayed. For a description of trusted path, see Chapter 10, “Trusted Facility Management” in *System Administration*.

Identifying ttymon Processes

The **`ps`** command lists all running processes, including **`ttymon`** processes. Because **`ttymon`** port monitors fork a process to handle each connection request, the number of `ttymon-related` entries that appear in the output of a **`ps`** listing may be greater than the number of running **`ttymon`** port monitors.



```
     UID   PID  PPID  CLS PRI    STIME TTY      TIME COMD
    root     0     0  SYS   0 10:26:53 ?        3:49 sysproc
    root     1     0   TS   0 10:26:59 ?        0:02 /sbin/init       0x80
    root   225     1   TS   0 10:30:45 ?        0:01 /usr/lib/saf/sac -t 300
    root  1935     1   TS   0 14:53:24 console  0:00 /usr/lib/saf/ttymon -g -v
-p
    root   172     1   TS   0 10:30:41 ?        0:00 /usr/lib/lpsched
    root   153     1   TS   0 10:30:38 ?        0:00 in.snmpd
    root   148     1   TS   0 10:30:37 ?        0:07 /usr/sbin/in.routed -q
    root   161     1   TS   0 10:30:39 ?        0:01 /usr/sbin/rpcbind
    root   163     1   TS   0 10:30:40 ?        0:00 /usr/lib/netsvc/rwall/
rpc.rwalld
    root   201     1   TS   0 10:30:42 ?        0:04 /usr/lib/nfs/nfsd -a
    root   165     1   TS   0 10:30:40 ?        0:00 /usr/lib/netsvc/rusers/
rpc.rusersd
    root   167     1   TS   0 10:30:40 ?        0:00 /usr/lib/netsvc/spray/
rpc.sprayd
    root   204     1   TS   0 10:30:42 ?        0:00 /usr/lib/nfs/statd
    root   190     1   TS   0 10:30:42 ?        0:15 /usr/lib/nfs/biod
    root   203     1   TS   0 10:30:42 ?        0:00 /usr/lib/nfs/nfsd -a
    root   196     1   TS   0 10:30:42 ?        0:00 /usr/lib/nfs/bootparamd
    root   197     1   TS   0 10:30:42 ?        0:00 /usr/lib/nfs/mountd
```

Screen 5-15. Sample Output of ps `-ef`

When a **`ttymon`** port monitor forks a child to process a connection request (that is, to do baud rate searching, set final `termio` options, and so on, before invoking the service), the controlling terminal is identified in the TTY field for this child process, as in the example above. (In the case of **ttymon**, the controlling terminal is the port. The question marks indicate there's no controlling terminal associated with the process. The **ps** command does not report any information about the service access controller.) For the parent **ttymon** port monitor process, this TTY field will be empty. In the above example, there are two ttymon port monitors running, with process IDs 8234 and 8235. Both were started by the SAC.

The output of the **ps** command does not identify the port monitors' *pmtag*s. The *pmtag* and process ID of a specific port monitor can be obtained using the **who** command (see the previous section).

Log Files

Problems often arise when a single port is monitored by more than one process. If a port (for example, **/dev/term/11**) is used by an enabled service under a **ttymon** port monitor running under the Service Access Facility, and the same port is also monitored by a **ttymon** process running in **ttymon** express mode, (that is, started by **init** when it reads **inittab**, not by **sac** when it reads its administrative file) then the port will behave unpredictably. The system administrator is must examine the system for such ambiguously configured ports.

There are also two log files that can be examined for clues to problems related to **ttymon** port monitors or ports monitored by **ttymon** port monitors: The Service Access Controller records aberrant port monitor behavior in **/var/saf/_log**; and each **ttymon** port monitor has its own log file, **/var/saf/***pmtag***/log**, where it records messages it receives from the SAC, services it starts, and so on.

The command

```
    tail -25 /var/saf/_log
```

will list the most recent 25 entries in the **_log** file.

Periodically, log files should be cleared or truncated. If you want **cron** to do the cleanup for you, add the appropriate commands to the file **/var/spool/cron/crontabs/root.**

Terminal Line Settings

init is a general process spawner that is invoked as the last step in the boot procedure. It starts the SAC. The SAC then looks in its administrative file to see which port monitors to start. Each **ttymon** port monitor started by the SAC looks in its own administrative file for the TTY ports to initialize. For each TTY port initialized, **ttymon** searches the **ttydefs** file for the information it needs to set terminal modes and line speeds. **ttymon** then waits for service requests. When a service request is received, and after the requester is authenticated, **ttymon** executes the command (usually **/usr/bin/shserv)** associated with the port that received the request.

From the system administrator's point of view, the key elements in managing terminal line settings are the **ttydefs** file and the **sttydefs** command, which maintains the **ttydefs** file.

The ttydefs File

/etc/ttydefs is an administrative file used by **ttymon**. It defines the speed and terminal settings for TTY ports. It contains five fields (*ttylabel*, *initial-flags*, *final-flags*, *autobaud*, and *nextlabel*), each of which is described below.

ttylabel
When **ttymon** initializes a port, it searches the **ttydefs** file for the entry that contains the `termio` settings for that port. The correct entry is the one whose *ttylabel* matches the *ttylabel* for the port. The *ttylabel* for the port is part of the *pmspecific* information included in **ttymon**'s administrative file. By convention, *ttylabel* identifies a baud rate (for example, `1200`), but it need not.

initial-flags
Contains the **termio** options to which the terminal is initially set. *initial-flags* must be specified using the syntax recognized by the **stty(1)** command.

final-flags
Contains the **termio** options set by **ttymon** after a connection request has been made and immediately before invoking a port's service. Final flags must be specified using the syntax recognized by **stty**.

autobaud
Autobaud is a line-speed option. When autobaud is used instead of a baud rate setting, **ttymon** determines the line speed of the TTY port by analyzing the first carriage return entered and sets the speed accordingly. If the autobaud field contains the character `A`, the autobaud facility is enabled. Otherwise, autobaud is disabled.

nextlabel
If the user indicates (by sending a BREAK) that the current **ttydefs** entry does not provide a compatible line speed, **ttymon** will search for the **ttydefs** entry whose *ttylabel* matches the *nextlabel* field. **ttymon** will then use that field as its *ttylabel* field. A series of speeds are often linked together in this way into a closed set called a hunt sequence. For example, `4800` may be linked to `1200,` which, in turn, is linked to `2400`, which is finally linked to `4800`.

All **termio** settings supported by the **stty** command are supported as options in the **ttydefs** file. For example, the system administrator will be able to specify the default erase and kill characters.

Figure 5-2f shows the relationship between the *ttylabel* and *nextlabel* fields in the **ttymon** administrative files and **ttydefs** files.

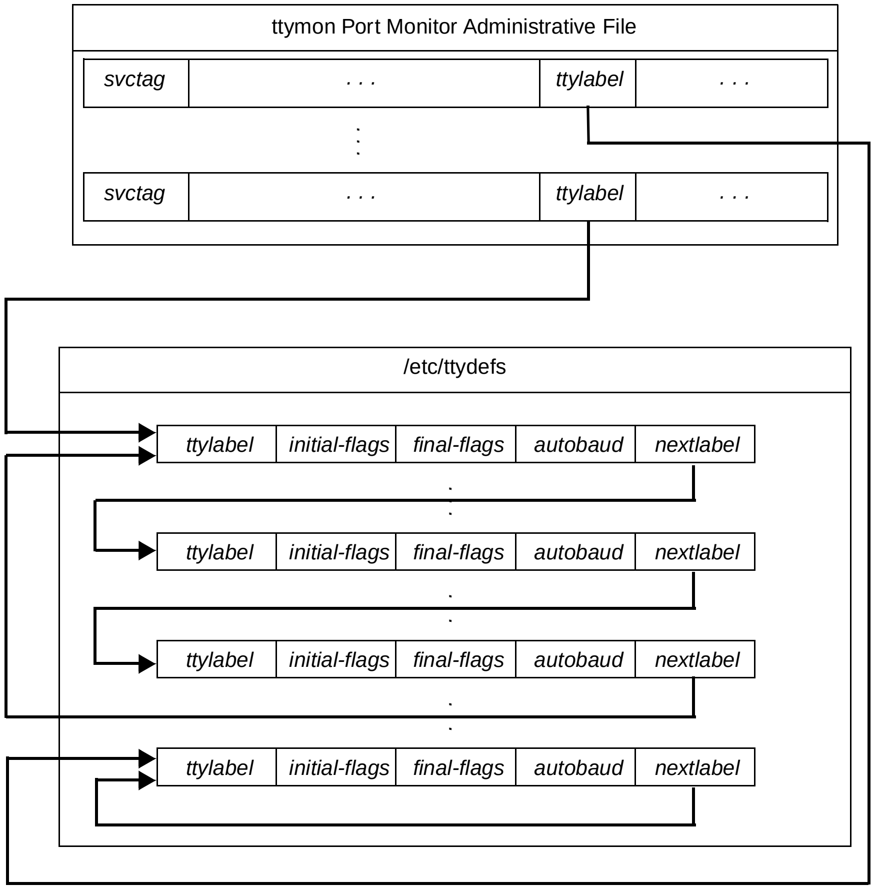
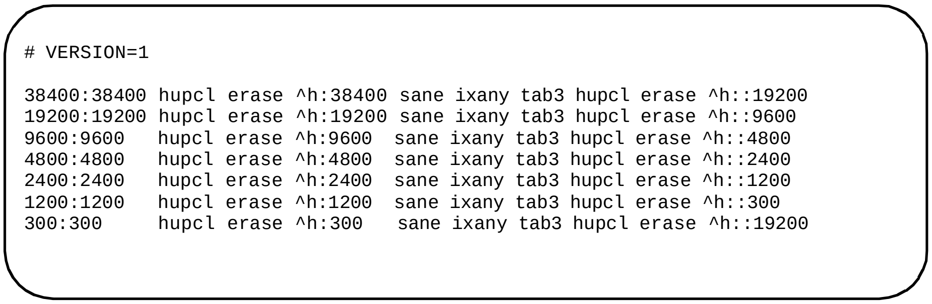


Figure 5-2. Links between the Port Monitor Administrative File and the `ttydefs` File

NOTE

> The format of the **`/etc/ttydefs`** file may change in future releases. For continuity across releases, use the **`sttydefs(1M)`** command to access this file.

The following screen shows a sample **`ttydefs`** file.



```
# VERSION=1

38400:38400 hupcl erase ^h:38400 sane ixany tab3 hupcl erase ^h::19200
19200:19200 hupcl erase ^h:19200 sane ixany tab3 hupcl erase ^h::9600
9600:9600   hupcl erase ^h:9600  sane ixany tab3 hupcl erase ^h::4800
4800:4800   hupcl erase ^h:4800  sane ixany tab3 hupcl erase ^h::2400
2400:2400   hupcl erase ^h:2400  sane ixany tab3 hupcl erase ^h::1200
1200:1200   hupcl erase ^h:1200  sane ixany tab3 hupcl erase ^h::300
300:300     hupcl erase ^h:300   sane ixany tab3 hupcl erase ^h::19200
```

Screen 5-16. Sample `ttydefs` File

The sttydefs Command

`sttydefs(1M)` is an administrative command that maintains the **`ttydefs`** file. The **`ttydefs`** file contains information about line settings and hunt sequences for the system's TTY ports. The **`sttydefs`** command and the **`ttydefs`** file together provide the facilities for managing terminal modes and line settings. The **`sttydefs`** command is used to

- print information contained in **`ttydefs`**
- add records for terminal ports to the **`ttydefs`** file
- remove records from the **`ttydefs`** file

Printing Terminal Line Setting Information

`/usr/sbin/sttydefs` **`-l`** `[`*ttylabel* `]`

If a *ttylabel* is specified, **`sttydefs`** prints the **`ttydefs`** record that corresponds to this *ttylabel.* If no *ttylabel* is specified, **`sttydefs`** prints this information for all records in the **`/etc/ttydefs`** file. **`sttydefs`** verifies that each entry it displays is correct and that the entry's *nextlabel* field refers to an existing *ttylabel.* An error message is printed for each invalid entry detected.

Adding Records to the ttydefs File

`/usr/sbin/sttydefs` **`-a`** *ttylabel* `[`**`-b`**`] [`**`-n`** *nextlabel*`] \`
  `[`**`-i`** *initial-flags*`] [`**`-f`** *final-flags*`]`

`sttydefs` with a **`-a`** option adds a record to the **`ttydefs`** file. *ttylabel* identifies the record. The following paragraphs describe the effect of the **`-b`**, **`-n`**, **`-i`**, or **`-f`** options when used with the **`-a`** option. The **`-a`** option is valid only when invoked by a privileged user.

The **`-b`** option enables autobaud.

The **-n** option specifies the value to be used in the *nextlabel* field. If *nextlabel* is not specified, **sttydefs** will set *nextlabel* to *ttylabel*.

The **-i** option specifies the value to be used in the *initial-flags* field. The argument to this option must be presented in a format recognized by the **stty** command. If *initial-flags* is not specified, **sttydefs** will set *initial-flags* to the **termio** flag 9600.

The **-f** option specifies the value to be used in the *final-flags* field. The argument to the **-f** option must be presented in a format recognized by the **stty** command. If *final-flags* is not specified, **sttydefs** will set *final-flags* to the **termio(7)** flags 9600 and sane.

The following command line creates a new record in **ttydefs**:

```
    sttydefs -aNEW -nNEXT -i“1200 hupcl erase ^h” \
    -f“1200 sane ixany hupcl erase ^h echo”
```

The flag fields shown have the following meanings:

300-19200 The baud rate of the line.

hupcl Hang up on close.

sane A composite flag that stands for a set of normal line characteristics.

ixany Allow any character to restart output. If this flag is not specified, only DC1 <CTRL><q> will restart output.

tab3 Send tabs to the terminal as spaces.

erase ^h Set the erase character to <CTRL><h>. On most terminals a <CTRL><h> is the backspace.

echo Echo erase character as the string backspace-space-backspace. On most terminals this will erase the erased character.

Creating a Hunt Sequence

The following sequence of commands adds records with labels 1200, 2400, 4800, and 9600 to the **ttydefs** file and puts them in a circular list or hunt sequence. In the example, the **nextlabel** field of each line is the *ttylabel* of the next line. The *nextlabel* field for the last line shown points back to the first line in the sequence.

The object of a hunt sequence is to link a range of line speeds. Entering a BREAK during the baud rate search causes **ttymon** to step to the next entry in the sequence. The hunt continues until the baud rate of the line matches the speed of the user's terminal.

```
sttydefs -a1200 -n2400 -i 1200 -f “1200 sane”
sttydefs -a2400 -n4800 -i 2400 -f “2400 sane”
sttydefs -a4800 -n9600 -i 4800 -f “4800 sane”
sttydefs -a9600 -n1200 -i 9600 -f “9600 sane”
```

The **ttydefs** file containing these records will look like this:

```
# VERSION=2
1200:1200:1200 sane::2400
2400:2400:2400 sane::4800
4800:4800:4800 sane::9600
9600:9600:9600 sane::1200
```

Removing Records from the ttydefs File

```
/usr/sbin/sttydefs -r ttylabel
```

The record for the *ttylabel* specified on the command line is removed from the **ttydefs** file.

The **-r** option is valid only when invoked by a privileged user.

NOTE

If a record you remove is part of a hunt sequence, be sure the sequence is repaired. It may be useful to run **sttydefs** with the **-l** option after a record has been removed. **sttydefs -l** will check for incorrect field values and broken hunt sequences and will print error messages.

Setting Terminal Options with the stty Command

The **stty(1)** command may be used to set or change terminal options after a user has logged in. A **stty** command line may also be added to a user's **.profile** to set options automatically as part of the login process. The following is an example of a simple **stty** command:

```
stty echo -tabs erase ^h
```

The options in the example have the following meanings:

echo Erase characters as you backspace.

-tabs Expand tabs to spaces when printing.

`erase ^h`	Change the character-delete character to <`CTRL`><`h`>. The default character-delete character is the pound sign (#). Most terminals transmit a <`CTRL`><`h`> when the `backspace` key is pressed.

Serial Line Access Issues

The following sections contain a comprehensive discussion of several serial line access issues including:

- port monitoring
- modem access
- modem settings

Changes to the ttymon Port Monitor

As an administrator, you should be aware of the following changes concerning **`ttymon(1M)`** (the system port monitor for detecting connection requests from other systems) and to the Basic Networking Utilities (BNU commands).

Enhancements and refinements have been made to **`ttymon`**, and you should notice only an improvement in the way the facility now works. The set of BNU commands has also undergone an upgrade, largely to support the changes made to **`ttymon`**.

This release is provided with a single installed version of **`ttymon`**, which is accessed as a front-end executable. In turn, the front-end **`ttymon`** then executes the port monitor, based on the value of the `TP_DEFAULT` parameter found in a new system file, **`/etc/default/tpath`**. For this release, the `TP_DEFAULT` parameter has a value of `NO`.

You should be aware that other operating system releases may have installed an **`/etc/default/tpath`** file with a `TP_DEFAULT` value of `YES`, which will invoke alternate **`ttymon`** processing.

If you have the need or opportunity to connect to an operating system that has TP, and if the TP version of **`ttymon`** has been installed, you should be aware that:

- **`ttymon`** processing on TP systems may use the TP version of **`ttymon`**.
- Devices that are accessed directly by the default version of **`ttymon`** are accessed indirectly by the TP version.
- You may need to contact the system administrator about the connect procedures and data transfer packages you can successfully use.

Modem Access

Many users prefer to manage communication settings through one of the many communication packages available for configuration and dialing. These packages allow you to select parity, terminal emulation, baud rate, phone number, transfer protocols, and other options available during a communications session.

However, you may access your modem directly without the aid of a communications package, and in those cases when you are experiencing connect problems, direct modem access may be preferable.   System references to your modem can be made as long as the system recognizes the port your modem is attached to.

Referencing the Port

Your modem may be accessed directly under your operating system by doing the following:

1. Look in the **`Dialers`** file (**`/etc/uucp/Dialers`**) for an entry to be used for a "direct" connection. The label will be `direct` or `direct_modem`. Note which label is used, since it must be the same in the **`Devices`** file.
2. Check **`/etc/uucp/Devices`** to be sure that there is a "direct" entry for the port where your modem is attached. The correct device name for a device is:

   ```
   /dev/ttyn_zz
       where n = adapter and zz = port.
   ```

   Note that the logical device names point to separate drivers, which differ based on the baud rate and `UART` values. See the documentation for **`asyc(7)`** for information on the driver you need and how to establish the appropriate reference in the **`Devices`** file. If there is no entry for your requirements, add one. Add the port as a `direct` line in **`/etc/uucp/Devices`** using either the `direct` or `direct_modem` label, depending on the label found in the **`Dialers`** file. For example:

   ```
   Direct /dev/tty0_00 - Any direct_modem
   ```
3. Use **`cu(1C)`** to access the modem. For example:

   ```
   cu -l /dev/tty0_00
   ```
4. When the direct connection is established (**`cu`** executes successfully), you may enter `AT` (Attention) data communications commands as well as modem configuration commands specific to your modem software.

Changes to Modem Settings

To ensure that the modem is optimally configured on your operating system, do the following:

1. Look in the **Dialers** file (**/etc/uucp/Dialers**) for an entry to be used for your modem. Try to locate an entry that is specifically for your modem (check the entry labels for your modem name). If you cannot find one, select the one that matches most closely. Normally, an acceptable entry is hayes. Most, if not all, available modems support Hayes compatibility.
2. Most modems have "dip" switches or other selectable settings already set to the defaults needed for standard network communication. Modems also accept configuration commands that address firmware settings. Consult your modem online manual for the correct settings and the appropriate commands for making changes if there is a need.
3. Make sure that the modem settings (defaults) do not conflict with the Dialers entry you select. This is especially important if someone else has used the modem before you got it. If you run into problems, or if the stated settings are apparently in conflict, make the change that is appropriate. To do this, do not modify the Dialers entry. Change the switches or use modem commands to change the firmware settings.
4. Note that firmware automatically reverts to the default settings each time you power on and off. You may set new defaults by using the "configuration set" feature available on most modems. Consult your manual if you need to reset defaults. Configuration setting is a chore often performed by communications packages. However, even if you set your modem directly (referencing the port), you have the ability to call up this feature.

What to Set

1. If you are using switched carrier, ensure that the **tty** port's jumpers are not set to carrier detect (CD) - always high.
2. If the modem is configured for 9600 bps or faster, configure the modem for hardware flow control.
3. Also, if the modem is configured for hardware flow control, specify a hardware flow controlled port.
4. The following guidelines should be used in configuring your modem and system for bi-directional use (that is, for both dialing in and dialing out).

   A. First, be sure you have reliable dial-out capability before you do any of the following:

      - Enable the system to answer calls.
      - Enable the modem to answer calls.
      - Enable error correction logic (if available on your modem).

NOTE

It is best to have dial-out capability working properly before calls are received.

B. Be sure to run **`ttyadm`** with the **`-b`** flag set so that **`ttymon`** is properly configured for the bi-directional line.

C. If you plan to run a BNU command (rather than **`ttymon`**), be sure to run **`uugetty`** instead of **`getty`**.

NOTE

These steps ensure that the proper locks and communications signals will be honored when the line is in use.

D. Set the default configuration parameters to specify local echo off. This will avoid runaway echo when waiting for a call.

E. Enable bi-directional use by doing the following:

- Set the value of the S0 register to be greater than 0. This will enable auto-answer mode.
- Activate the **`ttymon`** daemon or run **`uugetty`**.

F. If the label for the modem in **`/etc/uucp/Devices`** is `hayes`, it is recommended that the following parameters be set:

```
&C1 &D2
```

G. After configuration, add the port as an `ACU` line in **`/etc/uucp/Devices`**. For example:

```
ACU term/00 - Any hayes
```

H. The modem may be configured to ignore `DTR` (the “Data Terminal Ready” signal). If this is done, be sure that the phone line disconnects after use. You may disconnect at command level by using **`stty(1)`**. For example, use it to set baud rate to 0, which causes a disconnect.

```
stty 0
```

You may also issue **`AT`** commands directly to the modem. For example, use the command that causes a hang-up:

```
ATH0
```

Examples of ttyadm Flag Settings

The following flag settings are recommended for **`ttyadm`**.

`-b -r` 0 where the option **`-r`** *count* has been set to `0` (zero). Setting *count* to `0` means **`ttymon`** will display a prompt after receiving any character. This setting is often used with a modem configured for switched carrier, where bi-directional operation is required.

-b -r 8 where the option **-r** *count* has been set to `8`. Setting *count* to `8` means **ttymon** will display a prompt only after eight newlines have been received. This setting is often used when the modem has carrier detect set permanently high (`ON`).

no flags This setting is often used with a single, dial-in port monitor.

NOTE

The **getty** program still exists in the current OS. These applications, however, should be modified to use the administration interfaces available under the Service Access Facility. In particular, for serial port monitoring, ttymon in Daemon Mode is recommended.

In addition to the standard port monitors provided by the OS, the SAF architecture allows new port monitors to be installed by users and applications.

For more information about port monitors, see the *System Administrator's Guide* and the **ttymon(1M)** and **listen(1M)** online manual pages.

ttymon Timeouts

-t *timeout* This option specifies how long a port will wait for a service to be started, after an initial connect indication. *timeout* is specified in seconds.

internal_timeout_1 Internal to **ttymon** is a timeout used in bidirectional mode. The 60 second timeout monitors when a dial-out connection has been dropped for the purpose of raising `DTR` and restarting port monitoring.

internal_timeout_2 Again, internal to **ttymon** is a 4 second timeout which comes into play when a line is dropped. It prevents a new connection from being established for this interval. This timeout exists so that **ttymon** will not confuse a modem disconnect message with a new connect indication.

How to Monitor Ports

There are a several ways that you can monitor a port. These are the main configurations:

1. For logging in only (no uucp dial out):

   A. Over a directly connected terminal or machine to machine link.

   B. Over a modem using switched carrier:

- A modem that does not recognize commands from the DCE and does not send unsolicited characters to the DCE.

- A smart modem. One way this mode is different from bidirectional mode is in the way the **`login`** prompt is written. The prompt is written out immediately when **`ttymon`** begins to monitor a port. This can be echoed back by certain devices, such as a smart modem, causing a premature connection indication in the polling **`ttymon`**. Because of the prompt mechanism described previously, it is recommended that all *answering* by the modem be turned off using the following **`AT`** commands: `ATE0` and `ATQ1`. See section describing how to send commands to the modem. Alternatively, the **`CR/LF`** count used with the **`ttyadm -r`** flag, or bidirectional mode should be used.

C. Over a modem using carrier always high:

- A modem that does not recognize commands from the DCE.

- A smart modem which must use the **`ttyadm -r`** flag. This is to ensure that canonical mode with its associated echoing behavior is not set until the modem has successfully connected. Infinite echo is seen when the tty's line discipline and smart modem lock into a loop where they echo characters back and forth to each other. Because of the prompt mechanism described under Step B, adjust the **`CR/LF`** count by `+1` the value seen with bidirectional mode because the prompt is not written out immediately in that case. An alternative is for all *answering* by the modem to be turned off using the following **`AT`** commands: `ATE0` and `ATQ1`. Also note that if the **`-r`** count is too low, **`ttymon`** will change to canonical mode before the modem is connected resulting in infinite echo because the modem is still in command mode at this point.

D. Datakit. Mentioned for completeness, but is similar to the first bullet item in Step B.

2. For bidirectional use, using the **`ttyadm -b`** flag, for logging in or dialing out.

A. Over a directly connected terminal or machine to machine link.

B. Over a modem using switched carrier:

- A modem that does not recognize commands from the DCE and does not send unsolicited characters to the DCE.

- A smart modem.

C. Over a modem using carrier always high:

- A modem that does not recognize commands from the DCE.

- A smart modem which must use the **`ttyadm -r`** flag. This is to ensure that canonical mode is not set up on the tty before it has successfully connected.

D. Datakit. Mentioned here for completeness but similar to the first bullet item in Step B.

There is a delay from **`ttymon`**'s sensing of a connection attempt on its control channel and the creation of a data channel to the port. This results in lost characters. These characters are typically command mode messages from an intelligent modem and can thus be lost without serious consequences, unless the **`-r`** flag is used, in which case some of the anticipated **`CR/NL`** will be lost.

The listen Port Monitor

NOTE

> If your system is being run in compliance with the security criteria described in the "Security Administration" part of this book, network connections to the system are outside the evaluated configuration; the facilities described in this section should not be used to conform to the security guidelines.

`listen` is a port monitor invoked by the Service Access Controller (SAC). The Service Access Controller is the Service Access Facility's controlling process. It is started by **`init`** when the system enters multi-user mode. One of the SAC's functions after it is started is to start all port monitors the system administrator has configured.

`listen` monitors a connection-oriented transport network, receiving incoming connection requests, accepting them, and invoking the services that have been requested. The listener may be used with any connection-oriented transport provider that conforms to the Transport Interface (TLI) specification. The Transport Interface is documented in the *Network Programming Interfaces*. Beginning with the operating system, the listener conforms to the Service Access Facility model.

What listen Does

`listen` performs functions common to all port monitors:

- It initializes and monitors **`listen`** ports,
- it identifies and authenticates users, and
- it invokes the service associated with a port in response to requests.

The listener differs from previous listeners in several ways.

- It allows private addresses for services,
- passes connections (file descriptors) to standing servers,
- supports socket-based services, and

- supports RPC-based services and dynamic addressing.

Private Addresses for Services

Each **listen** service may have a transport address in addition to its service code (*svctag*). This private address is included in the port monitor's administrative file. The inclusion of private addresses for services allows a single **listen** process to monitor multiple addresses. This functionality might be useful for purposes such as balancing loads and making services, such as a name server, work more efficiently. The number of addresses on which the listener can listen is determined by the number of file descriptors available to the process.

Passing Connections to Standing Servers

By default, a new instance of a service is invoked for each connection. Beginning with the operating system, the listener has the ability to pass an incoming connection (file descriptor) to a standing server, eliminating the fork and exec overhead for each call [see **nlsadmin(1M)**]. This feature is useful for server processes that need to maintain state information.

NOTE

> A standing server is a server process or service that runs continuously and accepts connections through a FIFO or a named STREAM instead of being forked and execed.

Socket-based Services

The listener supports services that use sockets as their interface to the transport provider. Socket-based services are registered with the listener in the same way TLI-based services are, using the Service Access Facility's administrative commands. **listen** supports STREAMS; sockets is implemented as a STREAMS module and a library.

A socket-based service

- may or may not be an RPC service
- may have a statically or dynamically assigned address, or no private address
- may be invoked on each connection or may be a standing server, to which new connections are passed by a FIFO or a named STREAM

RPC-based Services and Dynamic Addressing

Dynamic addressing is most useful with RPC. RPC transport addresses may be either specified or dynamically assigned. In either case, the listener tells **rpcbind** what the address is and monitors it for incoming connections.

In the case of a dynamically assigned address, **`listen`** asks the transport provider to select a transport address each time the listener begins listening on behalf of the service.

When service addresses are dynamically assigned, the assigned address is written to the listener's log file.

listen and the Service Access Facility

The Service Access Facility (SAF) provides a generic interface to which all port monitors must conform. The port monitor used under the Service Access Facility's controller, "Service Access Controller" is **`listen`**. (See *"Overview of the Service Access Facility"*, *"Port Monitor Management"*, and *"Service Management"* for a description of the Service Access Facility, the administrative files it maintains. Also, for the commands used for the port monitor and service administration.)

There can be multiple invocations of **`listen`** port monitors, each identified by a unique *pmtag*. Each of these port monitors can monitor multiple ports for incoming connection requests.

A port has one and only one service associated with it. Each port, and its associated service, is identified by a service tag, *svctag*. Service tags for any given port monitor are unique.

When the Service Access Controller starts a port monitor, the port monitor reads its administrative file, which contains information about which ports to monitor and what service (that is, process) is associated with each port.

The nlsadmin Command

The Service Access Facility requires each type of port monitor to provide an administrative command. This command must format information derived from command-line options so that it is suitable for inclusion in the administrative files for that port monitor type. The command may also perform other port monitor-specific functions.

`nlsadmin` is the listener's administrative command. The **`nlsadmin`** command formats information based on the options with which it is invoked and writes this information to the standard output.

`nlsadmin` is one of the arguments used by **`pmadm -a`** to format information in a way suitable for inclusion in a **`listen`** administrative file. **`nlsadmin`** presents this information (as standard output) to **`pmadm`**, which places it in the file. This use of **`nlsadmin`** is described below under *"Adding a listen Port Monitor."* *pmspecific* information in a port monitor administrative file will be different for different port monitor types.

`nlsadmin` is also included on the **`sacadm`** command line when a port monitor is added to the system. It is used to supply the **`listen`** version number for inclusion in a port monitor's administrative file.

NOTE

> The port monitor administrative file is updated by the Service Access Facility's administrative commands, **`sacadm`** and **`pmadm`**. **`nlsadmin`** merely provides a means of presenting formatted, port monitor-specific (that is, **`listen`**-specific) data to these commands.

> The **`sacadm`** command line uses **`nlsadmin`** only with the **`-V`** option. **`nlsadmin -V`** returns the version number of the **`listen`** command being used.

Earlier versions of **`nlsadmin`** allowed the system administrator to add and delete services, start and stop the listener, and query the status of services using the **`nlsadmin`** command directly. Although this use of **`nlsadmin`** is retained for compatibility, these functions are now provided by the SAF administrative commands. This use of the SAF commands is described in the sections that follow. All uses of **`nlsadmin`**, including its use with the SAF administrative commands, are described on the **`nlsadmin(1M)`** online manual page.

Under the SAF, it is possible to have multiple listen processes for a single transport provider; each is identified by a unique *pmtag*. Before the current release of the operating system, only one listen process could be associated with a transport provider. This unique listen process was identified by its *net_spec* (see **`nlsadmin[1M]`**). If a listen process is added using the **`nlsadmin`** command, then its *pmtag* and *net_spec* will be identical. It is suggested that the SAF commands, **`sacadm(1M)`** and **`pmadm(1M)`**, instead of the **`nlsadmin`** command, be used to administer listen processes and the services managed by them.

NOTE

> For compatibility, a listen port monitor's *pmtag* and its *net_spec* are identical, although this does not have to be the case. In the sections that follow, the term *pmtag* is used where either the *pmtag* or a *net_spec* may be used.

Managing listen Ports

Finding Out Which listen Port Monitors Are Configured

```
sacadm -l [-t listen]
```

The **`sacadm`** command with only a **`-l`** option lists all port monitors currently defined for the system. For example:

```
PMTAG   PMTYPE  FLGS  RCNT    STATUS          COMMAND
tcp     listen  dx     5     NOTRUNNING  /usr/lib/saf/listen-m inet/tcp0 tcp #
ttymon1 ttymon  -      0     ENABLED     /usr/lib/saf/ttymon                 #
ttymon2 ttymon  -      0     ENABLED     /usr/lib/saf/ttymon                 #
```

Finding Out Which Services Are Configured for a listen Port Monitor

```
pmadm -l [-p pmtag] [-s svctag]
```

pmadm with only a **-l** will list all services for all port monitors on the system. If a port monitor is specified (**-p**), all services for that port monitor will be listed. The following is a sample listing for the command:

```
pmadm -l -p tcp
```

```
PMTAG   PMTYPE  SVCTAG       FLGS ID   SCHEME   <PMSPECIFIC>

tcp     listen  reportscheme -    root   -      - c -\/usr/sbin/reportscheme #
tcp     listen  rexec        u    -     crl -srexec - - c tirdwr /usr/lib/rexec/
rxserver \#remote execution
```

The following command line lists the general **listen** service (0).

```
pmadm -l -p pmtag -s 0
```

NOTE

By definition, service code 0 is for the **nlps_server**, which is a service that provides compatibility with **listen** service requests.

Adding a listen Port Monitor

```
sacadm -a -p pmtag -t type -c “cmd” -v `2pmspecific -V`e
[-n count] [ -f dx ] [ -z script ] [ -y “comment” ]
```

The following example shows how the listener's administrative command, **nlsadmin**, can be used to obtain the current version number of the listener's administrative file when used with **sacadm** to add a **listen** port monitor.

```
sacadm -a -p tcp -t listen \
      -c “/usr/lib/saf/listen -m inet/tcp0 tcp” \
      -v `sadmin -V`
```

This command line adds a line to the SAC's administrative file. The options that may be used with **`sacadm -a`** are described under *"Port Monitor Management"* and in the **`sacadm(1M)`** and **`nlsadmin(1M)`** online manual pages.

NOTE

> If the port monitor being added has the same name as an existing port monitor, the system administrator must remove the old one before adding the new one.

Removing a listen Port Monitor

```
sacadm -r -p pmtag
```

For example,

```
sacadm -r -p tcp
```

The SAC removes the line for port monitor `tcp` from its administrative file. The port monitor directory will remain in **`/etc/saf`** but will be removed and recreated when a new port monitor with the same name is added. To make changes to a port monitor entry, always remove the entry and add a new entry using the **`sacadm`** command. Do not edit the SAC administrative file.

Adding a Service

```
pmadm -a -p pmtag -s svctag [ -i id ] [ -S “scheme” ] \
-m “`sadmin options`-v `sadmin -V`[-y “comment”]
```

The following example adds a new service, **`tty`**, to a listener with *pmtag* `tcp` running as a port monitor under the Service Access Facility.

```
pmadm -a -p tcp -s tty -i -S login \
-m “`nlsadmin -c /usr/bin/shserv \
-p ntty,tirdwr,ldterm -A
\\\x00020ACF810b8b6e0000000000000000`“\
-v `nlsadmin -V` -y “Server-side for cu”
```

CAUTION

> The same address cannot be monitored by more than one **`listen`** port monitor at any given time. The first attempt to listen on an address will bind successfully; subsequent attempts will fail to bind. If both static and dynamic addresses are monitored by more than one listener, the static addresses are bound first, then the dynamic addresses. Mixing multiple listeners — each of which has static and dynamic addresses specified — may result in unpredictable behavior.

See *"Adding a Service"* under *"Service Management,"* or the **pmadm(1M)** and **nlsadmin(1M)** online manual pages for a full description of the **pmadm** command line options.

Removing a Service

```
    pmadm -r -p pmtag -s svctag
```

For example,

```
    pmadm -r -p tcp -s 23
```

removes service `23` from the `tcp` listener.

Enabling and Disabling Services

```
    pmadm -e -p pmtag -s svctag
    pmadm -d -p pmtag -s svctag
```

To enable a service on a specific port, first find out which port monitor is monitoring the port. Enter

```
    pmadm -l -t listen
```

This lists all services defined for listen-type ports.

If the port monitor is `tcp` and the service tag is `101`, the command

```
    pmadm -e -p tcp -s 101
```

will enable service `101`.

To verify that the port has been enabled, enter

```
    pmadm -l -p tcp -s 101
```

The `x` will have been removed from the **FLGS** column in the entry for this service.

When a service is disabled, all subsequent connection requests for the service will be denied. Using the same example,

```
    pmadm -d -p tcp -s 101
```

will restore the `x` to the **FLGS** field in the entry for service 101.

Authentication Schemes and User IDs

The **pmadm** command may be used to change or remove authentication schemes and user IDs.

```
    pmadm -c -S "scheme" [-i id] -p pmtag -s svctag
    pmadm -c -i id [-S "scheme"] -p pmtag -s svctag
```

Used with the **-c** option, the **-S** and **-i** options manipulate the contents of the *scheme* and *id* fields in the port monitor administrative file. To add the cr1 authentication scheme to service, `svc1`, under the **`listen`** port monitor, `tcp`.

```
pmadm -c -S cr1 -p tcp -s svc1
```

Now the `tcp` listener will invoke the cr1 authentication scheme before starting the `svc1` service. If either *scheme* or *id* is the NULL string, the corresponding field will be empty and the authentication scheme or user ID will be effectively removed from the service line in the file. For example,

```
pmadm -c -S "" tcp -s 101
```

will remove the authentication scheme for the service 101 from the administrative file for the `tcp` port monitor.

For a given service, there may be non-NULL entries in either of these fields, in both fields, or in neither. It is important to understand when the user ID determined by an authentication scheme is used and when the user ID specified in the *id* field is used. The following table describes the four possible cases.

scheme	*id*	Description
specified	`NULL`	Authentication is performed by the port monitor using the specified scheme. If authentication succeeds, the service is started with the ID determined by the scheme. If authentication fails, the service is not started.
`NULL`	specified	No authentication is performed by the port monitor. The service is started with the ID specified in the *id* field.
specified	specified	The port monitor invokes the specified authentication scheme. If authentication succeeds, the service is started with the ID from the *id* field. If authentication fails, the service is not started.
`NULL`	`NULL`	This is an error. The service will not start.

Disabling All Services Monitored by a listen Port Monitor

```
sacadm -d -p pmtag
```

To disable all services defined for the port monitor `tcp,` enter

```
sacadm -d -p tcp
```

Services are then temporarily unavailable; any future connection requests for services managed by this port monitor will be denied until the port monitor is enabled.

The command

```
sacadm -e -p tcp
```

will re-enable port monitor `tcp`.

Configuration Files

As a port monitor under the Service Access Facility, **`listen`** can customize the environment of each service it starts. It does this by interpreting a per-service configuration script, if one exists, immediately before starting the service. Per-service configuration scripts are optional. Configuration scripts are installed by the system administrator, using the **`pmadm`** command with **`-g`** and **`-z`** options [see **`pmadm(1M)`**].

It is also possible to customize the environment of a **`listen`** port monitor. A per-port monitor configuration script is defined using the **`sacadm`** command with **`-g`** and **`-z`** options [see **`sacadm(1M)`**]. The environment modifications made by a port-monitor configuration script are inherited by the port monitor and all the services it invokes. The environment of any particular service can then be customized further by using a per-service configuration script.

The **`doconfig(3I)`** online manual page describes the language in which configuration scripts are written.

Configuration scripts are not normally needed for basic operations.

Log Files

The listener creates and manages the log files **`/var/saf/`***pmtag*/log and **`/var/saf/`***pmtag*/o.log. Log file entries are in the following format:

> *date time; PID; message*

date and *time* show when the entry was made. *PID* is the ID of the process that made the log entry. *message* gives a description of the event or error that caused the log message.

The following events are logged, each:

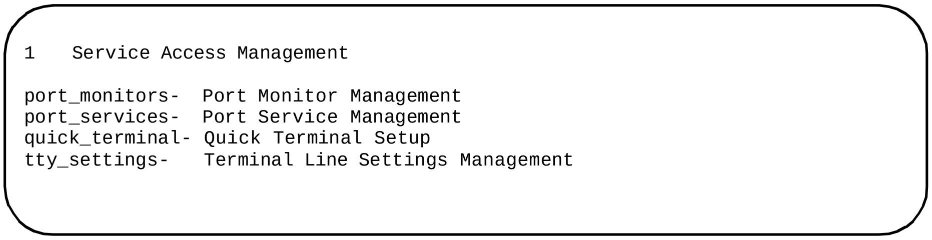
- connection that arrives
- service that is started
- file descriptor passed over a pipe
- state change that occurs

- error and unusual condition

The log files are held open by the listener process. Entries are made by two types of processes: the listener process (**listen**) and the NLPS server process (**nlps_server**). **nlps_server** is a service that provides compatibility with other service requests.

Quick Reference Guide to Managing Ports

The administrative procedures in this chapter are based on shell commands. If you prefer, however, you can do many tasks through a set of administration menus, instead. (Using the menus may be convenient if you're administering a system for the first time.) To use these menus, enter **sysadm ports**. The main menu for ports administration will appear on your screen as follows:



```
1   Service Access Management

port_monitors-  Port Monitor Management
port_services-  Port Service Management
quick_terminal- Quick Terminal Setup
tty_settings-   Terminal Line Settings Management
```

Screen 5-17. Main Menu for Managing Ports

When you select one of these entries, a series of menus and instructions will appear on the screen, prompting you to provide information and make selections needed for the task you've chosen.

To create a new connection between your system and a terminal, use the Quick Terminal Setup menu. Do ongoing administration with the Port Monitor Management and Port Service Management menus. The Terminal Line Setting Management menu concerns only tty ports; it's helpful in defining the default **termio** settings.

The rest of this section is a handy index to the shell commands that make up four categories of port manager administration:

- general management of sets of port monitors
- management of the port monitor administrative software (the Service Access Controller or SAC)
- management of the **ttymon** port monitor and terminal line settings
- management of the **listen** port monitor

Managing Sets of Port Monitors

- Add a port monitor entry to the SAC's administrative file.

  ```
  sacadm -a -p pmtag -t type -c "cmd" -v ver [-f dx]\
   [-n count][-y "comment"][-z script]
  ```

- Print port monitor status information.

  ```
  sacadm -l [-p pmtag | -t type]
  ```

- Print port monitor status information in condensed format.

  ```
  sacadm -L [-p pmtag | -t type]
  ```

- Print or replace the per-system configuration script **`/etc/saf/sysconfig.`**

  ```
  sacadm -G [-z script]
  ```

- Print or replace the per-port monitor configuration script **`/etc/saf/`***pmtag*/config.

  ```
  sacadm -g -p pmtag [-z script]
  ```

- Enable the port monitor *pmtag*.

  ```
  sacadm -e -p pmtag
  ```

- Disable the port monitor *pmtag*.

  ```
  sacadm -d -p pmtag
  ```

- Start the port monitor *pmtag*.

  ```
  sacadm -s -p pmtag
  ```

- Stop the port monitor *pmtag*.

  ```
  sacadm -k -p pmtag
  ```

- Remove the entry for port monitor *pmtag* from the SAC administrative file.

  ```
  sacadm -r -p pmtag
  ```

- Tell the SAC (and, optionally, the specified port monitor) to read its administrative file.

  ```
  sacadm -x [-p pmtag]
  ```

Managing Services

- Add a service entry to the port monitor administrative file.

  ```
  pmadm -a [-p pmtag | -t type] -s svctag [-i id] -m "pmspecific" \
       -v ver [-f xi] [-S "scheme"] [-y "comment"] [-z script]
  ```

- Change a service's authentication scheme or user ID. An authentication scheme or user ID may be removed by specifying the NULL string. For example, **-S** ““

  ```
  pmadm -c -S “scheme” [-i id] -p pmtag -s svctag
  pmadm -c -i id [-S “scheme”] -p pmtag -s svctag
  ```

- List service status information.

  ```
  pmadm -l [-t type | -p pmtag] [-s svctag]
  ```

- List service status information in condensed format.

  ```
  pmadm -L [-t type | -p pmtag] [-s svctag]
  ```

- Print, install, or replace the service-specific configuration script for service *svctag* associated with port monitor *pmtag*.

  ```
  pmadm -g -p pmtag -s svctag [-z script]
  ```

- Install or replace the service-specific configuration scripts for all services *svctag* associated with port monitors of type *type*.

  ```
  pmadm -g -s svctag -t type -z script
  ```

- Enable the service *svctag* associated with port monitor *pmtag.*

  ```
  pmadm -e -p pmtag -s svctag
  ```

- Disable the service *svctag* associated with port monitor *pmtag*.

  ```
  pmadm -d -p pmtag -s svctag
  ```

- Remove the entry for service *svctag* from the port monitor administrative file.

  ```
  pmadm -r -p pmtag -s svctag
  ```

Managing the ttymon Port Monitor and Terminal Line Settings

- List all port monitors (**-l** alone), all port monitors of a given type (**-t** *type*), or a single port monitor (**-p** *pmtag*).

  ```
  sacadm -l [-t type | -p pmtag]
  ```

- List all services for all port monitors (**-l** alone), all services for all port monitors of a given type (**-t** *type*), all services for a specific port monitor (**-p** *pmtag*), or a single service (**-s** *svctag*).

  ```
  pmadm -l [-t type | -p pmtag] [-s svctag]
  ```

- Add a **ttymon** port monitor. **ttyadm** used with **sacadm -a** or **pmadm -a** as an argument to the **-v** option provides the comment line containing the **ttymon** version number for the new port monitor administrative file.

  ```
  sacadm -a -p pmtag -t ttymon -c “cmd” -v `ttyadm -V`
  ```

- Remove a port monitor.

```
sacadm -r -p pmtag
```

- Add a service. **ttyadm** used with **pmadm -a** as an argument to the **-m** option provides the *pmspecific* fields for inclusion in the port monitor's administrative file.

```
pmadm -a -p pmtag -s svctag[-S “scheme”] [-i id][-f ux] \
    -v `ttyadm -V` -m “`ttyadm [-b] [-r count] [-c] [-h] \
    [-i msg] [-m modules] [-p prompt] [-t timeout] \
    -d device -l ttylabel -s service`“
```

- Change a service's authentication scheme or user ID. An authentication scheme or user ID may be removed by specifying the NULL string.

```
pmadm -c -S “scheme” [-i id] -p pmtag -s svctag
pmadm -c -i id [-S “scheme”] -p pmtag -s svctag
```

- Remove a service.

```
pmadm -r -p pmtag -s svctag
```

- Enable a service.

```
pmadm -e -p pmtag -s svctag
```

- Disable the service *svctag*, available through port monitor *pmtag*.

```
pmadm -d -p pmtag -s svctag
```

- Enable all services defined for port monitor *pmtag*.

```
sacadm -e -p pmtag
```

- Disable all services defined for port monitor *pmtag*.

```
sacadm -d -p pmtag
```

- Add an entry to the **/etc/ttydefs** file.

```
sttydefs -a ttylabel [-b] [-n nextlabel] [-i initial-flags] [-f final-flags]
```

- Print terminal line setting information from the **/etc/ttydefs** file for entries with the label *ttylabel*. If no *ttylabel* is specified, print terminal line setting information for all entries in the file.

```
sttydefs -l [ttylabel]
```

- Remove entries for the *ttylabel*.

```
sttydefs -r ttylabel
```

Managing the listen Port Monitor

- List status information for all port monitors (**-l** alone) or for all port monitors of a given type (**-t** *type*).

```
sacadm -l [-t type]
```

- If *svctag* is supplied, list status information for the service. If no service is specified, list status information for all services under port monitor *pmtag*.

  ```
  pmadm -l -p pmtag [-s svctag]
  ```

- Add a **listen** port monitor.

  ```
  sacadm -a -p pmtag -t listen -c \
  “/usr/lib/saf/listen pmtag”-v `nlsadmin -V`
  ```

- Remove a **listen** port monitor.

  ```
  sacadm -r -p pmtag
  ```

- Add a service under a **listen** port monitor.

  ```
  pmadm -a -p pmtag -s svctag [-i id] [-S “scheme”] \
  -m “`nlsadmin options`“ -v `nlsadmin -V` [-y “comment”]
  ```

- Change a service's authentication scheme or user ID. An authentication scheme or user ID may be removed by specifying the NULL string. For example, **-S** ““

  ```
  pmadm -c -S “scheme” [-i id] -p pmtag -s svctag
  pmadm -c -i id [-S “scheme”] -p pmtag -s svctag
  ```

- Remove a service under a **listen** port monitor.

  ```
  pmadm -r -p pmtag -s svctag
  ```

- Enable service *svctag* under port monitor *pmtag*.

  ```
  pmadm -e -p pmtag -s svctag
  ```

- Disable service *svctag* under port monitor *pmtag*.

  ```
  pmadm -d -p pmtag -s svctag
  ```

- Disable all services under port monitor *pmtag*.

  ```
  sacadm -d -p pmtag
  ```

- Enable all services under port monitor *pmtag*.

  ```
  sacadm -e -p pmtag
  ```

6
Collecting Data on System Use

6
Collecting Data on System Use

Introduction

NOTE

> If your system is being run in compliance with the security criteria described in Chapter 10, “Trusted Facility Management” of this book, the Accounting Utilities will not be installed; see Chapter 12, “Installing Software on an Enhanced Security System” for a list of the software packages that should be installed on a B2-compliant system.

The operating system accounting utilities are a set of tools (C language programs and shell scripts) that collect data on system usage (by CPU usage, users, and processes) and organize this data into summary files and reports. By using these tools to keep track of connect sessions and disk usage, you can

- charge for usage
- troubleshoot performance problems
- tune the performance of applications
- manage installation security

CAUTION

> Do not run commands associated with the Accounting Utilities while your system is running in single-user mode. If you do, files will be created that will not be accessible afterward.

Once the accounting system has been set up (see *“Setting Up Accounting”* below), it runs, for the most part, automatically.

- Between system start-up and shutdown, it collects (in accounting files) raw data about system use, such as records of the logins used, processes run, and data stored.
- Once a day, **`cron`** invokes the **`runacct`** program, which does three tasks: (a) it processes the various accounting files; (b) it produces both cumulative summary files and daily accounting reports; and (c) it prints the daily reports (via 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.

Four types of accounting are available: connect accounting, process accounting, disk accounting, and fee calculations. This chapter describes how these types of tracking are done and how you can use the accounting utilities to do the work needed on your system.

Types of Accounting

The data collected daily (as described above) can help you do four types of accounting: connect accounting, process accounting, disk accounting, and fee calculations. The rest of this introduction defines these four types of accounting.

Connect Accounting

Connect accounting enables you to determine how long a user was logged in, and to obtain information about the usage of tty lines, the number of reboots on your system, and the frequency of the stopping and starting of the accounting software. To obtain this information, the system stores records of time adjustments, boot times, the turning on or off of the accounting software, 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 **/var/adm/wtmp.** Entries in the **wtmp** directory 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 allows 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 (divided between users and the system), the amount of memory used, the commands run, and the controlling tty during 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 over 500 blocks, it moves the file to **/var/adm/pacct**? where? is the next unused increment (expressed as a number).

Disk Accounting

Disk accounting allows you to gather (and format) the following data about the files each user has on disks: the name and ID of the user, and the number of blocks used by the user's files. This data is collected by eight programs in the accounting package: a shell script called **dodisk** and five C programs that it invokes **sfsdiskusg**, **ufsdiskusg**,

vxdiskusg, **acctdusg**, and **acctdisk**). **sfsdiskusg**, **ufsdiskusg**, and **vxdiskusg** gather the file data by reading file inodes directly from **sfs** and **ufs** file systems, respectively. (Note that no **xfsdiskusg** program exist to read inodes directly, therefore, the **dodisk** program defaults to using **acctusg** for the **xfs** file system.) **acctdusg** does stat calls for each file in the file system tree to gather data and works for any file system type. **sfsdiskusg**, **ufsdiskusg**, and **vxdiskusg** are faster than **acctdusg**. **acctdisk** formats the data gathered by **sfsdiskusg**, **ufsdiskusg**, **vxdiskusg**, and/or **acctdusg**, and saves the information in **/var/adm/acct/nite/disktacct.**

The **dodisk** script can be used in either of two ways: in fast mode or in slow mode. Fast mode uses **acctdusg** The fast mode syntax is:

```
/usr/lib/acct/dodisk file_systems
```

File systems are specified by their special device names (such as **/dev/dsk/1s0).** If file systems are not specified, then the file systems used are those found in **/etc/vfstab** for which the value of **fsckpass** is 1. Note that when **dodisk** uses **/etc/vfstab,** it skips remote resources.

When run in slow mode, **dodisk** invokes **acctdusg** to gather all the disk accounting information. The slow mode syntax is:

```
/usr/lib/acct/dodisk -o mountpoints
```

If no mountpoints are specified, the root mountpoint is used.

One note of caution: information gathered by running **dodisk** in either fast mode or slow mode is stored in the **/var/adm/acct/nite/disktacct** file. This information is overwritten the next time **dodisk** is used. Therefore, avoid using both modes on the same day. This allows **runacct** to use the information in the **/var/adm/acct/nite/disktacct** file before it is overwritten by new output from **dodisk**.

Fee Calculations

If you charge your users for special services, such as file restores 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

The **acctcom** program is stored in **/usr/bin;** all other binary programs are stored in **/usr/lib/acct.** These programs, which are owned by **bin** (except **accton**, which is owned by root), do various functions. For example, **/usr/lib/acct/startup** helps initiate the accounting process when the system enters multi-user state. The **chargefee** program is used to charge a particular user for a special service, such as performing a file restore 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.

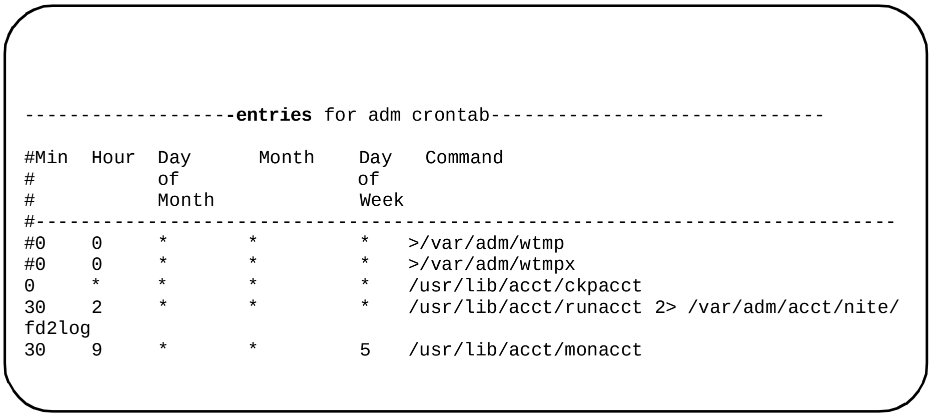
Setting Up Accounting

To set up system accounting so it will be running while the system is in multi-user state (system state 2), you need to create and/or modify four files: **/etc/rc0.d/K22acct, /var/spool/cron/crontabs/adm, /etc/rc2.d/S22acct,** and **/var/spool/cron/crontabs/root.**

If you want accounting to be shut off during shutdown, link **/etc/init.d/acct** to **etc/rc0.d/k22acct.**

If you want accounting to be turned on when the system is in multi-user state (system state 2), link **/etc/init.d/acct** to **/etc/rc2.d/S22acct.**

Most of the **cron** entries needed for accounting are put into a database called **/var/spool/cron/crontabs/adm.** The entries in this database allow **ckpacct** to be run periodically, **runacct** to be run daily, and **monacct** to be run on a fiscal basis. Screen 6-1 shows several sample entries; your entries may vary. (Note that the comment lines in the file shown here may not appear in your file; they're included to clarify the meaning of the fields shown.)



```
-------------------entries for adm crontab------------------------------

#Min  Hour  Day      Month    Day   Command
#           of                of
#           Month             Week
#------------------------------------------------------------------------------
#0    0     *       *         *    >/var/adm/wtmp
#0    0     *       *         *    >/var/adm/wtmpx
0     *     *       *         *    /usr/lib/acct/ckpacct
30    2     *       *         *    /usr/lib/acct/runacct 2> /var/adm/acct/nite/
fd2log
30    9     *       *         5    /usr/lib/acct/monacct
```

Screen 6-1. Sample cron Entries for Accounting

In a delivered system, **/var/spool/cron/crontab/adm** already contains two entries for controlling the growth of **wtmp** files on systems without the accounting utilities:

```
0 0 * * *  >/var/adm/wtmp
0 0 * * *  >/var/adm/wtmpx
```

If you're running the accounting utilities you won't want these entries, but we recommend against deleting them in case the accounting utilities are later removed from your system. To make these entries inactive (that is, to neutralize their effect on accounting), prepend a # (pound) sign to each. If you find any other entries in **/var/spool/cron/crontab/adm,** we recommend leaving them alone.

For the **adm** crontab, assign root as the owner, **sys** as the group, and 644 as the permissions mode.

The entry for **`dodisk`** needs to be appended 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 will pretty much run on its own.

Daily Accounting

Here is a step-by-step summary of how operating system accounting works:

1. When the operating system is switched into multi-user state, the **`/usr/lib/acct/startup`** program is executed. The **`startup`** program executes several other programs that invoke accounting:
   - 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 6-1 presents a summary of how the raw accounting data is gathered and where it is stored.

Table 6-1. Raw Accounting Data

File	Information	Written By	Format
`/var/adm/wtmp`	connect sessions	**`login`**, **`init`**	**`utmp.h`**
	date changes	**`date`**	
	reboots	**`acctwtmp`**	
	shutdowns	`shutacct` shell	
`/var/adm/pacct?`	processes	kernel (when process ends)	**`acct.h`**

Table 6-1. Raw Accounting Data (Cont.)

File	Information	Written By	Format
		`turnacct` switch creates new file when old one reaches 500 blocks	
/var/adm/fee	special charges	**chargefee**	
/var/adm/acct/nite/disktacct	disk space used	**dodisk**	**tacct.h**

- The **turnacct** program, invoked with the **on** option, begins process accounting. Specifically, **turnacct on** executes the **accton** program with the argument **/var/adm/pacct.**
- The **remove** shell script "cleans up" the saved **pacct** and **wtmp** files left in the **sum** directory by **runacct**.

2. The **login** and **init** programs record connect sessions by writing records into **/var/adm/wtmp.** Any date changes (made by running **date** with an argument) are also written to **/var/adm/wtmp.** Reboots and shutdowns (via **acctwtmp**) are also recorded in **/var/adm/wtmp.**

   When a process ends, the kernel writes one record per process, in the form of **acct.h**, in the **/var/adm/pacct** file.

   One program, **acctdusg**, tracks disk usage by login. It is invoked by the shell script **dodisk.**

   cron executes the **ckpacct** program every hour to check the size of **/var/adm/pacct.** If the file grows past 500 blocks (default), `turnacct switch` is executed. (The `turnacct switch` 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 then turns off process accounting.

   If you provide services on a request basis (such as file restores), you can keep billing records by login, by 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.

3. **runacct** is executed via **cron** each night. It processes the accounting files **/var/adm/pacct?, /var/adm/wtmp, /var/adm/fee,** and **/var/adm/acct/nite/disktacct** to produce command summaries and usage summaries by login.

4. 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*.

5. 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.

The runacct Program

The main daily accounting shell procedure, **runacct**, is normally invoked by **cron** during non-prime time hours. The **runacct** shell script processes connect, fee, disk, and process accounting files. It also prepares daily and cumulative summary files for **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 end 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**.

If the files **lock** and **lock1** exist when **runacct** is invoked, **runacct** will complain. These files are used to prevent simultaneous execution of **runacct**. 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.

Reentrant States of the runacct Script

To allow **runacct** to be restartable, processing is broken down into separate reentrant states. A file is used to remember the last state completed. When each state completes, **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/pacct?** (except the **pacct** file) are moved to **/var/adm/Spacct?.***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 correctness. Because some date changes 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 two steps called CONNECT1 and CONNECT2.

PROCESS

The **acctprc** program is used to convert the process accounting files **/var/adm/Spacct?.***MMDD*, into total accounting records in **ptacct?.*MMDD***. The **Spacct** and **ptacct** files are correlated by number so if **runacct** fails, **Spacct** files are not reprocessed unnecessarily. One precaution should be noted: when restarting **runacct** in this state, remove the last **ptacct** file because it will not be complete.

MERGE

Merge the process accounting records with the connect accounting records to form **daytacct**.

FEES

Merge any ASCII **tacct** records from the file **fee** into **daytacct**.

DISK

If the **disktacct** file has been produced (by running the **dodisk** procedure), merge that file into **daytacct** and move **disktacct** to **/tmp/disktacct.***MMDD*.

MERGETACCT

Merge **daytacct** with **sum/tacct**, the cumulative total accounting file. Each day, **daytacct** is saved in **sum/tacct.*MMDD***, so **sum/tacct** can be recreated if it is corrupted or lost.

CMS

The program **acctcms** is run several times. It is first run to generate a command summary (using the **Spacct?** files), which it writes to **sum/daycms. acctcms** 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 **/var/adm/acct/sum/loginlog,** the report of when each user last logged on. (If **runacct** is run after midnight, records of the date and time some users last logged on will be incorrect by one day.)

USEREXIT

Any installation-dependent (local) accounting program can be included here .**runacct** expects it to be called **/usr/lib/acct/runacct.local.**

CLEANUP

Clean up temporary files, run **prdaily** (saving its output in **sum/rprt*MMDD***), remove the locks, and exit.

runacct Error Messages

The **runacct** procedure can fail for a variety of reasons, the most common of which are **/var** running out of space and a corrupted **wtmp** file. If the **active** ***MMDD*** file exists, check it first for error messages. If the **active** file and **lock** files exist, check **fd2log** for any mysterious messages. The following is a list of error messages produced by **runacct** and recommended recovery actions for each situation.

ERROR: locks found, run aborted

> The files **lock** and **lock1** were found. These files must be removed before **runacct** can restart. 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.

ERROR: acctg already run for date: check /var/adm/acct/nite/lastdate

> The date in **lastdate** and today's date are the same. Remove **lastdate.**

ERROR: turnacct switch returned rc=?

> Check the integrity of **turnacct** and **accton**. The **accton** program must be owned by **root** and have the setuid bit set.

ERROR: Spacct?.*MMDD* already exists

> File setups have probably already run. Check the status of the files; then run setups manually, if necessary.

ERROR: /var/adm/acct/nite/wtmp.*MMDD* already exists, run setup manually

> **/var/adm/wtmp** has already been copied to **/var/adm/acct/nite/wtmp.***MMDD*

ERROR: wtmpfix errors see /var/adm/acct/nite/wtmperror

> **wtmpfix** detected a corrupted **wtmp** file. Use **fwtmp** to correct the corrupted file.

ERROR: invalid state, check /var/adm/acct/nite/statefile

> The file **statefile** is probably corrupted. Check **statefile** and read **active** before restarting.

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 logoffs to logins exceeds about 3:1, 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 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 entirely foolproof; occasionally, a file is corrupted or lost. Some files can be restored from backup archives; the loss of others can be ignored. However, certain files are crucial to the integrity of the accounting system. If these files are corrupted, they must be fixed.

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 multi-user state, 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 may slip through **wtmpfix** and cause **acctcon** to fail. The following steps show how to patch up a **wtmp** file.

```
cd /var/adm/acct/nite
fwtmp < wtmp.MMDD > xwtmp
ed xwtmp
delete corrupted records or
delete all records from beginning
up to the date change
w
q
fwtmp -ic < xwtmp > wtmp.MMDD
```

Screen 6-2. Repairing a `wtmp` File

If the **`wtmp`** file is beyond repair, create a null **`wtmp`** file. The existence of this null file prevents 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. When this happens, the process is charged to the owner of the first login in the password file for the appropriate user ID.

Fixing tacct Errors

If your installation is using the accounting system to charge users for system resources, the integrity of **`sum/tacct`** will be important to you. Occasionally, mysterious **`tacct`** records may appear with negative numbers, duplicate user IDs, or a user ID of `65,535.` If this happens on your system, check **`sum/tacctprev`** (using **`prtacct`** to print it). If it looks all right, patch up the latest **`sum/tacct.`*`MMDD`*** file and then recreate **`sum/tacct.`** Screen 6-3 shows an example of a simple patchup procedure.

```
cd /var/adm/acct/sum
acctmerg -v < tacct.MMDD > xtacct
ed xtacct
     remove the bad records
     write duplicate uid records to another file
w
q
acctmerg -i < xtacct > tacct.MMDD
acctmerg tacctprev < tacct.MMDD > tacct
```

Screen 6-3. Repairing a `tacct` File

The current **`sum/tacct`** can be recreated by merging all existing **`tacct.`*`MMDD`*** files by using **`acctmerg`**, since the **`monacct`** procedure removes all the old **`tacct.`*`MMDD`*** files.

Restarting runacct

When called without arguments, **runacct** assumes it's being invoked for the first time of the day. If **runacct** is being restarted, the argument *MMDD* (which specifies the month and day for which **runacct** is to rerun the accounting) is necessary. 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**:

```
nohup runacct 2> /var/adm/acct/nite/fd2log &
```

To restart **runacct**:

```
nohup runacct 0601 2> /var/adm/acct/nite/fd2log &
```

To restart **runacct** in a specific state:

```
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 restores, in the file **fee**. This file is incorporated by **runacct** every day.

To register special fees, enter the following command:

```
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. With such an arrangement, an operator 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. (See Screen 6-7 for a sample report.) The **monacct** program also creates summary files (in the **/var/adm/acct/fiscal** directory) of the accounting information that has been collected. This information can be used to generate monthly billing. To generate a monthly billing, many system administrators customize the accounting process with their own shell scripts.

Setting Up Non-Prime Time Discounts

System accounting provides facilities to give users a discount for non-prime time system use. To make this arrangement work, you must inform the accounting system of the dates of holidays and the hours that are considered non-prime time, such as weekends. To do this, edit the **/etc/acct/holidays** file, which contains the prime/non-prime table for the accounting system. The format is composed of three types of entries:

Comment

Comment lines are marked by an asterisk in the first column of the line. Comment lines may appear anywhere in the file.

Year Designation

The year should be shown on the first data line (noncomment line) in the file and must appear only once. If you do not specify the current year, error messages will be sent to the designated user with appropriate privileges. The line consists of three fields of four digits each (leading white space is ignored). For example, to specify the year as 1992, prime time start at 9:00 A.M., and non-prime time start at 4:30 P.M., include the following entry:

```
1992  0900   1630
```

A special condition allowed for in the time field is that the time 2400 is automatically converted to 0000.

Company Holidays

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. See Table 6-2 for an example holiday list.

Table 6-2. 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

When **`runacct`** is invoked, it generates four basic reports:

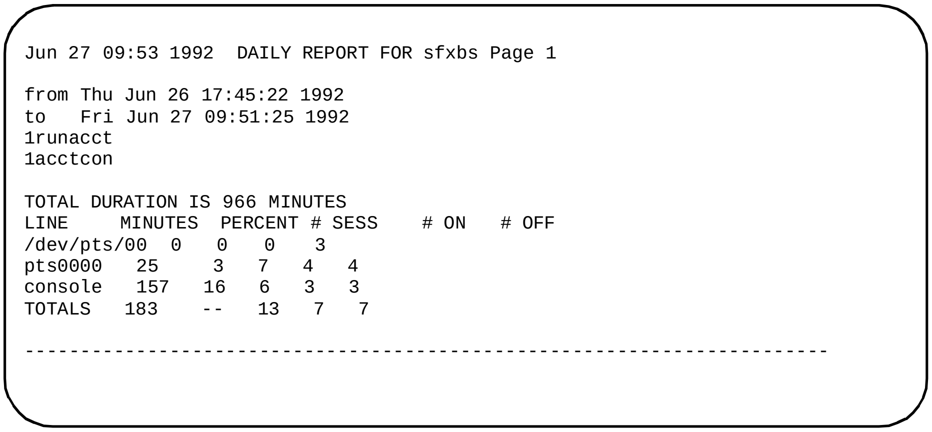
- The daily report shows line utilization by tty number.
- The daily usage report shows usage of system resources by users (listed in order of UID).

- The daily command summary shows 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.
- The last login shows the last time each user logged in (arranged in chronological order).

The rest of this section describes each report in detail.

Daily Report

This report gives information about each terminal line used. Screen 6-4 shows a sample daily report.



```
Jun 27 09:53 1992  DAILY REPORT FOR sfxbs Page 1

from Thu Jun 26 17:45:22 1992
to   Fri Jun 27 09:51:25 1992
1runacct
1acctcon

TOTAL DURATION IS 966 MINUTES
LINE     MINUTES  PERCENT # SESS    # ON   # OFF
/dev/pts/00  0   0   0   3
pts0000   25     3   7   4   4
console   157   16   6   3   3
TOTALS   183    --   13   7   7

------------------------------------------------------------------------
```

Screen 6-4. Sample Daily Report

The **`from`** and **`to`** lines tell you 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 records 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 multi-user state (accessible through the terminal lines). The columns are:

LINE: the terminal line or access port.

MINUTES: the total number of minutes that 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.

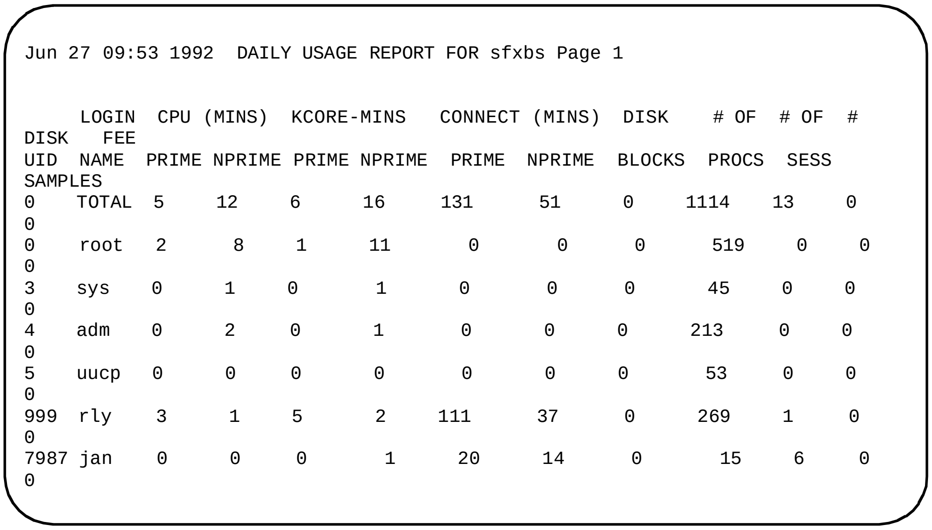
# ON: This column does not have much meaning anymore. It used to list the number of times a port was used to log a user on, but because **`login`** can no longer be executed explicitly to log in a new user, this column should be identical with SESS.

# OFF: This column reflects not just the number of times a user logs off but also any interrupts that occur on that line. Generally, interrupts occur on a port the first time **`ttymon`** is invoked after the system has been brought to multi-user state. This column is significant when the # `OFF` exceeds the # `ON` by a large factor. This usually means the multiplexer, modem, or cable is 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 on which the connect accounting is based. 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 at a furious rate, general system performance will be affected.

Daily Usage Report

The daily usage report gives a breakdown of system resource utilization by user. Screen 6-5 shows a sample of this type of report.



```
Jun 27 09:53 1992  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
```

Screen 6-5. Sample Daily Usage Report

The data provided includes the following:

UID: The user ID.

LOGIN NAME: The login name of the user. This information is useful because it identifies a user who has multiple login names.

CPU (MINS): The amount of time a user's process used the central processing unit. This category is divided into `PRIME` and `NPRIME` (non-prime) utilization. The accounting system's idea of this division is located in the **`/etc/acct/holidays`** file.

KCORE-MINS: A cumulative measure of the amount of memory a process uses while running. The amount shown reflects kilobyte segments of memory used per minute. This measurement is also divided into `PRIME` and `NPRIME` amounts.

CONNECT and (MINS): The amount of “real time” used. What this column really identifies is the amount of time 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 safely conclude the owner of the relevant login is a “line hog”: someone who logs in first thing in the morning and hardly touches the terminal the rest of the day. Watch out for this kind of user. This column is also divided into `PRIME` and `NPRIME` utilization.

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 a user may have a shell procedure that has run out of control.

# OF SESS: The number of times a user logged on to the system is shown in this column.

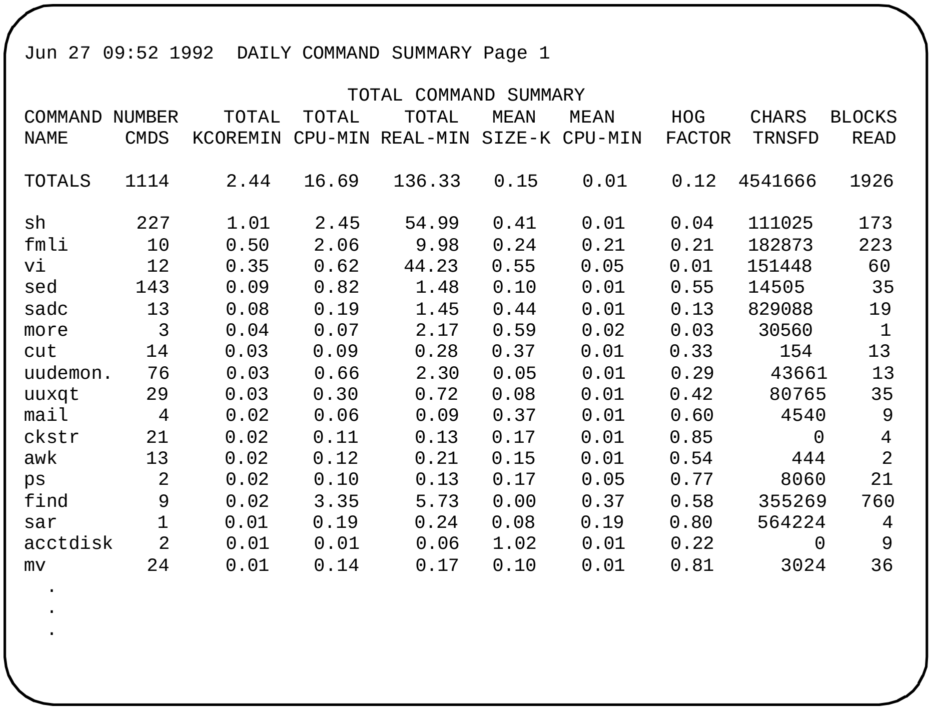
# DISK SAMPLES: This shows how many times disk accounting was run to obtain the average number of `DISK BLOCKS` listed earlier.

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, such as file restores.

Daily Command Summary

The daily command summary report shows the system resource utilization 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 command and monthly reports are virtually the same except that the daily command summary reports only on the current accounting period while the monthly total command summary tells the story for 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. Screen 6-6 shows a sample daily command summary.



```
Jun 27 09:52 1992  DAILY COMMAND SUMMARY Page 1

                                  TOTAL COMMAND SUMMARY
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
fmli       10      0.50    2.06     9.98   0.24    0.21    0.21   182873    223
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
  .
  .
  .
```

Screen 6-6. Sample Daily Command Summary

The data provided includes the following:

COMMAND NAME
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 with which you see program names such as **a.out**, **core**, or any other name that does not seem quite right. By doing so, you may be able to spot programs written for private use (such as someone's favorite version of online backgammon) hidden behind innocuous names. **acctcom** is also a good tool for determining who executed a suspiciously named command and whether it was executed with the UID or GID set to **root**.

PRIME NUMBER CMDS
The total number of invocations of this particular command during prime time.

NON-PRIME NUMBER CMDS
The total number of invocations of this particular command during non-prime time.

TOTAL KCOREMIN The total cumulative measurement of the amount of kilobyte segments of memory used by a process per minute of run time.

PRIME TOTAL CPU-MIN
The total processing time this program has accumulated during prime time.

NON-PRIME TOTAL CPU-MIN
The total processing time this program has accumulated during non-prime time.

PRIME TOTAL REAL-MIN
Total real-time (wall-clock) minutes this program has accumulated.

NON-PRIME TOTAL REAL-MIN
Total real-time (wall-clock) minutes this program has accumulated.

MEAN SIZE-K The mean of the `TOTAL  KCOREMIN` over the number of invocations reflected by `NUMBER CMDS`.

MEAN CPU-MIN The mean derived between the `NUMBER CMDS` and `TOTAL CPU-MIN`.

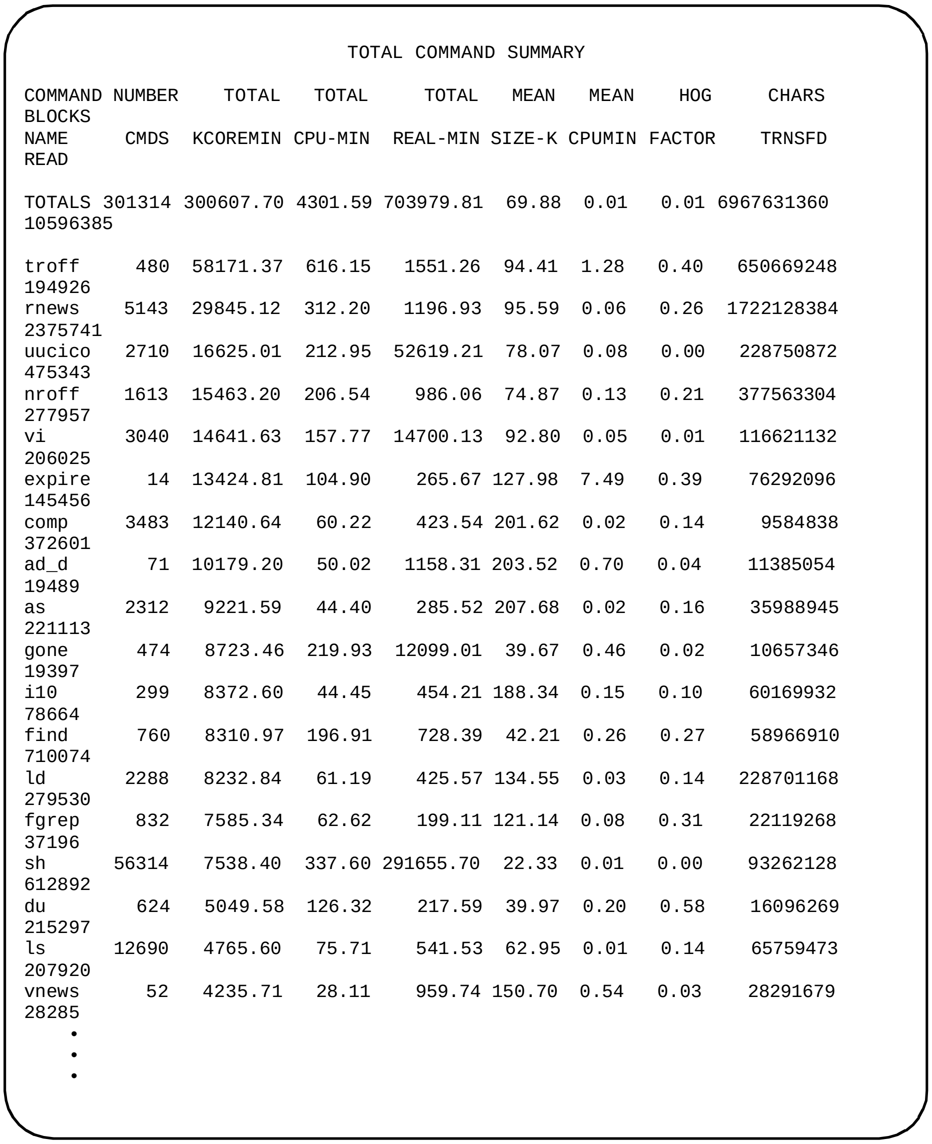
HOG FACTOR The ratio of system availability to system utilization, shown by the total CPU time divided by the elapsed time. This gives a relative measure of the total available CPU time consumed by the process during its execution.

CHARS TRNSFD The total count of characters transferred by the `read` and `write` system calls. (Because of overflow, this number may be negative.)

BLOCKS READ The total count of the physical block reads and writes performed by a process.

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`**. Screen 6-7 shows a sample report.

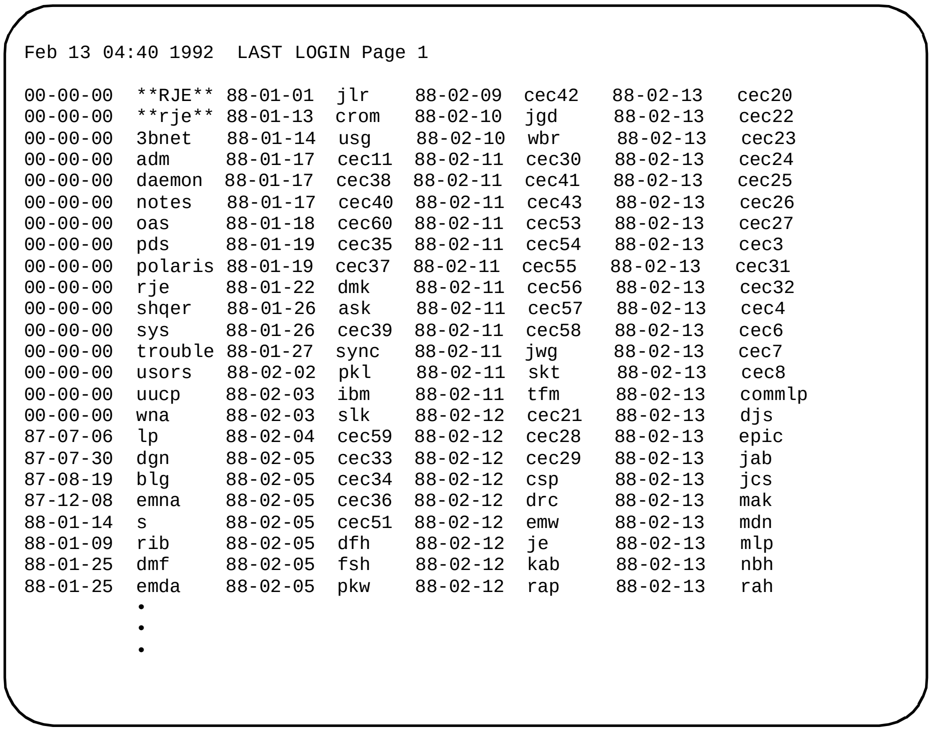


```
                              TOTAL COMMAND SUMMARY

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
    •
    •
    •
```

Screen 6-7. Sample Monthly Command Summary

Last Login Report

This report simply 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. Screen 6-8 shows a sample report.



```
Feb 13 04:40 1992  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   mdn
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
          •
          •
          •
```

Screen 6-8. Sample Last Login

Looking at the pacct File with acctcom

At any given time, the contents of the **/var/adm/pacct?** files or any file with records in the **acct.h** format may be examined 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: the name of the command (prepended with a **#** sign if the command was executed with the UID or GID set to **root**), the user, the tty name (listed as? if unknown), the starting time, ending time, and real time (in seconds), the CPU usage (in seconds), and the mean size (in K). The following information can be obtained by using options: F (the **fork/exec** flag, 1 for fork without exec), STAT (the system exit status), HOG FACTOR, KCORE MIN, CPU FACTOR, CHARS TRNSFD, and BLOCKS READ.

The following options are available with **acctcom**:

-a Show some average statistics about the processes selected (printed after the output records).

-b Read the file(s) backward, showing the latest commands first. (Has no effect if reading standard input.)

-f Print the **fork/exec** flag and system exit status columns.

-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 kcore-minutes instead of memory size.

-m Show mean core size (shown by default unless superseded by another option).

-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.

-l *line* Show only processes belonging to the terminal **/dev/***line*.

-u *user* Show only processes belonging to *user.*

-g *group* Show only processes belonging to *group*.

-s *time* Show processes existing at or after *time*, given in the format *hr[:min[:sec]].*

-e *time* Show processes existing at or before *time*, given in the format *hr[:min[:sec]].*

-S *time* Show processes starting at or after *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.

-n *pattern* Show only commands matching *pattern* (a regular expression as in **ed** except that "+"means one or more occurrences).

-q Don't print output records; just print averages (akin to **-a**).

-o *ofile* Instead of printing the records, copy them in **acct.h** format to *ofile.*

-H *factor* Show only processes that exceed *factor*, where *factor* is the hog factor explained above in the description of the **-h** option.

-O *sec* Show only processes with CPU system time exceeding *sec* seconds.

-C *sec* Show only processes with total CPU time (system plus user) exceeding *sec* seconds.

-I *chars* Show only processes transferring more characters than the cut-off number specified by *chars*.

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).

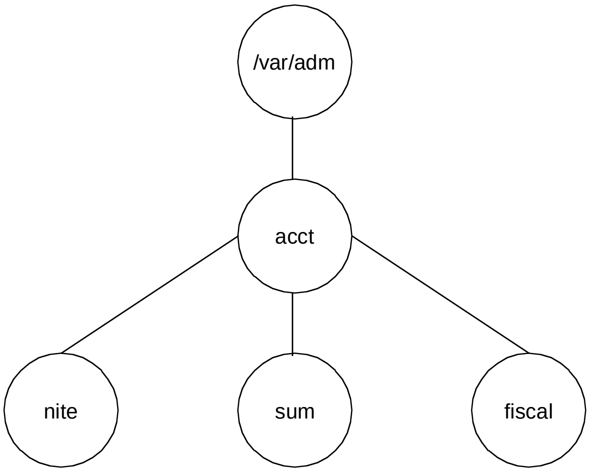


Figure 6-1. Directory Structure of the adm Login

The following files are found in the **/var/adm** directory:

dtmp	output from the **acctdusg** program
fee	output from the **chargefee** program in the format of ASCII **tacct** records
pacct	active process accounting file
pacct?	process accounting files switched via **turnacct**
Spacct?.*MMDD*	process accounting files for *MMDD* during execution of **runacct**

The **/var/adm/acct** directory contains the **nite**, **sum**, and **fiscal** directories, which contain actual data collection files. The **nite** directory contains files 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 the **`acctcon1`** program: connect session records in **`ctmp.h`** format
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 lock1	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	contains beginning and ending dates (taken 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? total command summary file for fiscal period? in internal summary format

fiscrpt? report similar to **rprt**? for fiscal period?

tacct? total accounting file for fiscal period?

Quick Reference Guide to Accounting

- Starting accounting:

  ```
  /usr/lib/acct/startup
  ```

- Turning off accounting:

  ```
  /usr/lib/acct/shutacct
  ```

- Switching the **pacct** file to the **pacct?** file:

  ```
  /usr/lib/acct/ckpacct
  ```

- Examining the contents of **pacct:**

  ```
  acctcom
  ```

- Charging a fee:

  ```
  /usr/lib/acct/chargefee login_name amount
  ```

- Processing accounting files into a daily summary:

  ```
  /usr/lib/acct/runacct 2 > /var/adm/acct/nite/fd2log
  ```

- Doing disk accounting:

  ```
  /usr/lib/acct/dodisk
  ```

- Creating a monthly accounting report:

  ```
  /usr/lib/acct/monacct
  ```

- Printing **tacct.h** files in ASCII format:

  ```
  /usr/lib/acct/prtacct
  ```

7
Installing Add-on Software

7
Installing Add-on Software

Introduction

NOTE

> The software configuration of your system should be altered when the system is in single-user mode only. Privileged use of the software management commands described in this chapter is restricted to single-user mode to ensure that Mandatory Access Control levels and privileges are assigned to installed files properly. See Chapter 3, “Booting and System States” in System Administration for a description of single-user mode. If your system is being run in compliance with the security criteria described in Chapter 11 “Introduction to Security”, adding privileged software to the system will invalidate the system rating, since this software essentially becomes part of the Trusted Computing Base (TCB). See Chapter 12 “Installing Software on an Enhanced Security System” for additional information.

Software management responsibilities include setting up your software installation environment, installing software on your computer, keeping track of it while it is on your system, and removing it as necessary. In particular, you will need to do the following tasks:

- Create and use admin files which define values for your installation default parameters.
- Store interactions with a package or set (using the **`pkgask`** command) in a file to be used when installing software in non-interactive mode.
- Install software packages or sets using the command **`pkgadd`**. It can be used to execute both interactive and non-interactive installation.
- Install software from a remote computer. An example shows how to use the Remote File Sharing Utilities, which make remote filenames appear to be local.
- Use the **`pkgchk`** command to check the integrity of packages or sets after they have been installed.
- Display information about installed packages or sets using the **`pkginfo`** command. Various types of information about packages or sets installed on your system can be displayed.

- Store packages without installing them. (You can spool a package onto your computer for future installation.)
- Remove packages or sets with the **pkgrm** command.

If the Operations, Administration and Maintenance (OA&M) package (a non-graphical menu interface) is installed on your system you can use it to complete many of these tasks. See *"Installing Add-on Software through OA&M Menus"* later in this chapter.

Basic Software Management Terminology and Concepts

To install software packages or sets properly you need to understand the basic terms defined below. This section also describes the installation software database and the differences between interactive and non-interactive installation, and explains how to name packages or sets on the command line.

A "software package" is a separately installable software component that provides one or more capabilities or services to users. The package may contain compiled programs, files, and installation scripts.

A "set" is a special purpose package, referred to as a Set Installation Package (SIP), and a collection of one or more packages that are members of the set. The SIP controls the installation of the set.

Both packages and sets are delivered on an "installation medium." An installation medium is any physical storage device, such as a diskette or tape, on which packages can be stored.

Packages are delivered on a medium in a "datastream format." (The datastream format consists of a header and a series of **cpio** archives and it can be read from any raw device.) **pkgadd** automatically determines the format of the medium when it reads the first volume. The format of a package can be translated (either from file system to datastream or vice-versa) by running the **pkgtrans** command. [See **pkgtrans(1)** online manual page and *Compilation Systems Manual* for details.]

Software packages or sets can be installed from a directory. This is quite useful in remote file sharing environments where a directory containing packages or sets is advertised from a server computer to a large network.

You can also "spool" a package for later installation. Spooling causes the contents of the package to be copied from the installation medium to a spool directory. No other installation action is taken. (For example, the installation scripts are not executed.)

You cannot spool a set directly. Rather, you should use **pkginfo** to determine which packages are members of the set, and then spool the package(s).

Setting Security Levels for Software

When installing software, you should assign program files a security level that is readable to all users who will be using the software. Unless otherwise specified by the package, **pkgadd** installs files at a level of **USER_PUBLIC** on **sfs**-type file systems.

The appropriate security level for software depends on which users need access to the files. For example, if the program needs to be accessible to all users, the executable file should be set to **`USER_PUBLIC.`** Executable files intended for more limited use should be protected by an appropriate security level. Any data files used by the software should be set to a level appropriate to the data they contain.

CAUTION

> You should not install software at the **`SYS_PUBLIC`** or **`SYS_PRIVATE`** levels unless that software has been certified as trusted. These levels are intended for trusted software only. Installing any other software at these levels will undermine the security of your system and invalidate your system's B2 rating.

See the *"Mandatory Access Control"* section of Chapter 17, "Administering Mandatory Access Control and Multilevel Directories" for details about security levels.

If the software you are installing will be used by users at multiple security levels, and if it makes use of public directories to which all users need write access, these directories should be Multilevel Directories (MLDs). For details, see the *"Multilevel Directories"* section of Chapter 15, "*Administering Mandatory Access Control and Multilevel Directories*" in this book.

Package Instances

Variations of a software package may reside on your system simultaneously. Each variation of a package is known as an "instance" and is treated as a separate entity. Three parameters, defined by the package developer, combine to uniquely identify each entity. You cannot install on the same system two instances of a package that have identical values for all three parameters. These parameters are:

`PKG` The package abbreviation (remains constant for every instance of a package)

`VERSION` the software package version

`ARCH` the software package architecture

For example, two versions of a package that run on the same system might be identified as:

```
PKG=“abbr”        PKG=“abbr”
VERSION=“2.1 p9”  VERSION=“2.1 p12”
ARCH=“(nh6000)”   ARCH=“(nh6000)”
```

Each package instance has a "package identifier." This ID maps the three identifying parameters to one name, the name of this package instance on your system. To look up the package identifier of a package instance, run

```
pkginfo
```

See *"Showing Information About Installed Packages or Sets"* for details.

The first instance of a package installed on a system is identified by the package abbreviation. If there are subsequent installations, **pkgadd** assigns a numerical suffix to the package abbreviation, beginning with .2. For example **mypkg.2**, is the package identifier for the second installed version of **mypkg**.

An instance is given the lowest extension available and so may not reflect the order of installation. For example, if **mypkg.2** was deleted after **mypkg.3** was installed, the next instance would be named **mypkg.2**.

When asked for *pkginst* in any procedure in this chapter, use the package identifier or SIP (for a set). Remember that when you have only one instance of a package on a system, the package identifier is the package abbreviation.

To execute a single command for all instances of a package, specify the abbreviation of the desired package, followed by an asterisk (*). If you use the asterisk in your command line, enclose the command line in single quotes or insert a backslash (\) before the asterisk. For example, if you want to display information about all instances of package **mypkg**, run the **pkginfo** command as follows:

```
pkginfo mypkg.\*
```

If you want to display information about only one instance of this package, specify the specific package identifier, as, for example:

```
pkginfo mypkg.3
```

Relocatable Packages

Some packages, or parts of a package (files), must be installed in a particular directory. Other packages have no requirement for where the package should be installed. The package developer can specify that files are relocatable (that is, that they may be placed in arbitrary locations).

The directory portion of a relocatable object pathname is filled in as the package is installed. There are four places from which the directory value can originate:

- The package itself may ask where you want to install relocatable objects.
- **pkgadd** asks you where you want to install a relocatable package if no default is set in the admin file and you are in interactive mode.
- The admin file can assign a value to an installation default parameter that defines the directory where relocatable packages should be installed. If set, the value of this parameter is used as the directory portion of the name.
- The package also delivers a directory name where relocatable objects should be installed if there is a question as to where they should be placed (for example, if you have no default defined in the admin file and are operating in non-interactive mode).

At installation, all relocatable objects are given full pathnames, and any variable value associated with them is resolved.

The Installation Software Database

The installation software database stores information about all packages and sets installed on the system. Entries for every component of a package or set contain a record of the package to which the component belongs, the component name, where it resides, its type, and a list of other packages that might reference the component. The database also keeps “attribute information” (such as the component's access permissions, owner ID, and group ID) and content information (such as the file size and the time of last modification).

The system uses this database to see if an object is shared by more than one package, to see if other packages depend on it, or to perform a number of other checks when adding or removing a package or set. When a package or set is installed or removed with **`pkgadd`** or **`pkgrm`**, information about it is automatically added to, or removed, from this database.

The installation software database keeps track of the status of packages. A package can be either fully installed, meaning it has successfully completed installation, or partially installed, meaning it has not successfully completed installation. In the latter case, you can remove the portions that were installed, based on the information stored in the installation software database.

You can use the **`pkginfo`** command to survey the contents of the installation software database. The commands **`installf`** and **`removef`** can be used to modify its contents. See the respective online manual pages for more details about these two commands.

Methods of Installation

You can install software packages or sets in either of two modes: “interactive” or “non-interactive.” In interactive mode, **`pkgadd`** queries you during installation and receives input on how to proceed. In non-interactive mode, you give **`pkgadd`** the information it needs by supplying two files: an admin file and a response file (discussed in later sections). You can also spool a package for later installation.

If **`pkgadd`** encounters a problem and cannot find instructions in the admin file or response file, it terminates the installation.

Preparing for Installation

It is not necessary to create a new admin file for every package installation. The **`default`** admin file is provided with the operating system. The **`default`** file prevents prompting for input during installation by setting appropriate defaults. The **`default`** admin file should always be used for set installation or removal. See *“Setting Installation Defaults”* for detailed information on default parameters.

A generic admin file named **`check`** is also delivered with your operating system. All the defaults defined in this file request that you be queried if a problem occurs.

You can create other admin files, if you want other default sets for different installation situations.

NOTE

> Do not make changes to the system-supplied admin files **`default`** or **`check.`** If you want to define different parameter values, create a new admin file. [See **`admin(4)`** for details.]

When you invoke the **`pkgadd`** command, it automatically uses the system supplied file, **`default`**. If you want to use the **`check`** admin file, or one of your own, specify the name of the file after the **`-a`** option.

For further details see the following sections, in this chapter:

- *"Setting Installation Defaults"* (instructions for creating an admin file)
- *"Storing Interactions with a Package or Set"* (instructions for creating a response file).

Interactive Installation

You can install a package in the interactive mode. This choice takes full advantage of the sophistication of the **`pkgadd`** command. You simply specify **`-a`** check with the **`pgadd`** command, and if problems occur during installation, you will be notified so you can instruct **`pkgadd`** how to proceed.

If **`pkgadd`** finds a potential problem, it needs instructions on how to proceed. You can create a list of default solutions in an admin file and specify that file on the command line. [See **`admin(4)`** for a description of admin files.]

Interactive Installation Checklist

When installing a package in interactive mode, you should:

1. Decide how your installation defaults should be defined for this installation. Choose an admin file that establishes these defaults or, create a new admin file by using an editor. (This is not applicable for sets since the **`default`** file is used.)
2. Install the package by running the **`pkgadd`** command.

Non-Interactive Installation

You can install a package or set in non-interactive mode. For some types of installation — such as those done in background or by batch execution — interactive mode cannot be used; non-interactive mode is mandatory.

When you do run **`pkgadd`** non-interactively (with the **`-n`** option) you must specify a response file. A response file contains a list of answers to problems not addressed by the instructions in the admin file. When **`pkgadd`** encounters problems during a non-interactive installation it checks both the admin file (for default solutions) and the response file (for system specific solutions). Before installing a package non-interactively, you need to determine whether either or both types of files are needed.

The advantage of non-interactive mode is that a package can be installed with no need for your intervention. As a result, the installation may be done more quickly than an interactive installation. The drawback is that you cannot supply instructions if unanticipated problems arise.

For sets, all interaction is accomplished immediately after the **`pkgadd`** command is executed, so you do not have to provide input later in the process.

Non-interactive Installation Checklist

When installing a package in non-interactive mode, you should:

1. Decide how installation defaults should be defined for this installation. Choose an admin file that establishes these defaults or, create a new admin file. This step is important in a non-interactive installation because the system will not be able to question you if problems arise.
2. Determine whether you need to create a response file. Some packages include scripts that ask for input during installation. If you are installing such a package or set, you will have to store answers to the script's questions in a response file. To create a response file run the **`pkgask`** command.

NOTE

> To find out if a package is interactive, and thus needs a response file for non**`-interactive`** installation, run the **`pkgask`** command. It will either begin creating a response file or inform you that a response file is not needed.

Install the package by running the **`pkgadd`** command. Include the **`-n`** option to request non-interactive mode, the **`-r`** option to specify a response file, and the **`-a`** option to specify an admin file. If you are using an admin file other than **`default`**, you may also want to use the **`-q`** option to suppress the display of non-essential information, and the **`-l`** option to create a log file which holds messages for later review.

Spooling a Package

Spooling copies a software package from the installation medium (without installing it) to a directory on your system, for later installation.

You cannot spool a set directly. Rather, you should use **`pkginfo`** to determine which packages are members of the set, and then spool the package(s).

Spooling Checklist

When spooling a package instead of installing it, you should:

1. Run the **pkgadd** command with the **-s** option.
2. When you are ready to install the spooled package, follow one of the checklists above and name the spool directory as the installation medium.

Setting Installation Defaults

The commands to install and remove packages and sets, **pkgadd** and **pkgrm** respectively, check for errors during their execution. When a problem occurs, they refer to an installation administration file for instructions on how to proceed. This “admin” file defines values for parameters, each of which supplies a resolution for a potential problem.

The default admin file, **default**, specifies that no checking will be done, except to see if there is enough room to install the package. (**default** causes the installation to proceed without prompting.) You can name an alternative admin file by using the **-a** *admin* option, where *admin* is the name of the alternative admin file. For package installation, for example, you may choose to install interactively by specifying the **check** admin file as the argument to **-a**. (**check** causes a prompt to appear whenever a problem or conflict occurs.) For set installation, the **default** admin file should always be used. Both **check** and **default** are located in **/var/sadm/install/admin.**

If you want to assign different values for your installation default parameters, you can create your own admin file(s).

NOTE

> Do not make changes to the system-supplied admin files **default** or **check.** If you want to define different parameter values, create a new admin file. [See **admin(4)** for details.]

Creating an Admin File

Create an admin file with the editor of your choice.You can use any name for a new admin file. Define each parameter on a single line in the following format:

param=value

A description of each parameter, along with a list of permissible values, is shown in the list below. Any of these parameters may be assigned the value ask, which means that if the situation occurs you will be asked to give instructions at that time.

You do not have to assign values to every parameter. If **pkgadd** needs a parameter value and one is not assigned in the admin file, the default value ask is used.

NOTE

The value `ask` cannot be defined in an admin file to be used for non-interactive installation. If it is, installation will fail when a problem occurs.

`basedir` Indicates the base directory where relocatable packages will be installed. The parameter may contain `$PKGINST` to indicate a base directory that is to be a function of the package instance. For example, if you make the assignment **`basedir=/opt/$PKGINST`**, all relocatable packages that use the **`basedir`** parameter will be placed under **`/opt`** in a directory having the same name as the package instance.

`mail` Defines a space-separated list of users to whom mail will be sent following installation of a package or set. If the list is empty, no mail is sent. If the parameter is not present in the admin file, the default value of **`root`** is used. (The value `ask` cannot be assigned to this parameter.)

`runlevel` Indicates resolution if the computer run level (that is, system state) is not correct for the installation or removal of a package or set. Options are:

- **`nocheck`**: do not check for run level
- **`quit`**: abort installation if run level is not met

`conflict` Indicates resolution if installation will cause a previously installed file to be overwritten, modified, or have its permissions changed, thereby creating a possible conflict between packages. Options are:

- **`nocheck`**: do not check for conflict; files in conflict will be overwritten.
- **`quit`**: abort installation if conflict is detected.
- **`nochange`**: override installation of conflicting files; they will not be installed.

`setuid` Checks for executables that will have set-UID or set-GID bits enabled after installation. Options are:

- **`nocheck`**: do not check for set-UID executables.
- **`quit`**: abort installation if set-UID processes are detected.
- **`nochange`**: override installation of set-UID processes; processes will be installed without set-UID bits enabled.

`action` Determines whether installation scripts provided by package developers may contain a possible security impact (for example, by enabling the set-UID or set-GID bits). Options are:

- **nocheck**: ignore security impact of scripts.
- **quit**: abort installation if scripts may have a negative security impact.

`partial`

Checks to see if a version of the package or set is already partially installed on the system. Options are:

- **nocheck**: do not check for a partially installed package.
- **quit**: abort installation if a partially installed package exists.

`instance`

Determines how to handle installation if a previous instance of the package (including a partially installed instance) is already installed on the system. Options are:

- **quit**: exit without installing if an instance of the package already exists (does not overwrite existing packages).
- **overwrite**: overwrite an existing package if only one instance exists. If there is more than one instance, but only one has the same architecture, it overwrites that instance. Otherwise, the administrator is prompted with existing instances and asked which to overwrite. (If installed in non-interactive mode, installation terminates.)
- **unique**: do not overwrite an existing instance of a package. Instead, a new instance of the package is created. The new instance will be assigned the next available package identifier.

`idepend`

Controls resolution during package or set installation if package dependencies are not met. Options are:

- **nocheck**: do not check package dependencies.
- **quit**: abort installation if package dependencies are not met.

`rdepend`

Controls resolution during package removal if other packages depend on the one to be removed. Options are:

- **nocheck**: do not check package dependencies.
- **quit**: abort removal if package dependencies are not met.

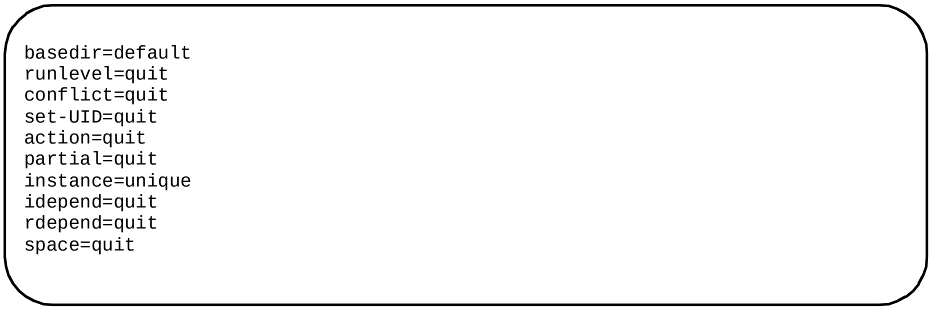
`space`

Controls resolution if disk space requirements for the package or set are not met. Options are:

- **nocheck**: do not check space requirements (installation fails if it runs out of space).

- **quit**: abort installation if space requirements are not met.

The **check** admin file (**/var/sadm/install/admin/check**) defines **mail** as root and all other parameters as ask. The sample admin file shown below defines parameters differently than check.



```
basedir=default
runlevel=quit
conflict=quit
set-UID=quit
action=quit
partial=quit
instance=unique
idepend=quit
rdepend=quit
space=quit
```

Screen 7-1. Sample admin File

Storing Interactions with a Package or Set

Before you install a package or set in non-interactive mode, you must prepare answers for the questions that a package installation script would ask you during the installation process. The **pkgask** command executes the appropriate installation script, thus showing you the questions and allowing you to respond to them. Your answers are stored in a file or a directory called a “response file.”

You supply a name for the response file on the command line when you execute **pkgadd** to install the package or set in non-interactive mode. The installation script will use the response file to access the information when it is needed.

Creating a Response File

To create a response file, execute

> pkgask **-r** *response pkginst*

where *response* is the full pathname of the file or directory in which your responses will be saved and *pkginst* is the package identifier or SIP for the package or set to be installed. When a SIP is specified as the **pkginst**, the response file must be a directory. If **pkginst** specifies a SIP, request scripts are run for all packages that are members of the set, and the resulting response files are placed in the directory specified in the **-r** option. (Use **pkginfo** to find out the package identifier or SIP if you do not know it. See *“Showing Information about Installed Packages and Sets”* for details on **pkginfo**.) If *response* is a directory, a file named **response/pkginst** is created.

NOTE

You must use a package identifier with a numerical suffix if multiple versions reside on the installation medium. When there is only one version of a package on a medium, the package identifier is the package abbreviation without a suffix.

The package identifier suffix defines the package instance only on that particular medium. A new suffix will be assigned when this package is installed on your system.To find out what instances are available on a medium, run **`pkginfo -d`** *device*.

Installing Software Packages or Sets

Software package or set installation is the process of copying a software package from an installation medium (such as a tape cartridge) onto your computer, performing any actions requested by installation scripts, and recording package objects in the installation software database. Installation can be performed in either interactive or non-interactive mode.

Installing a Package or Set in Interactive Mode

The default installation mode is interactive. To interactively install a software package or set, run the **`pkgadd`** command. For example, to install a package or SIP named `pkgA` from a tape device named `ctape1`, enter:

```
pkgadd -d ctape1 pkgA
```

You can install multiple packages at one time by separating package names with white space. The named packages must all be on one medium (one tape), or **`pkgadd`** will fail with an error. A sample command for installing multiple packages at one time follows:

```
pkgadd -d ctape1 pkgA pkgB pkgC
```

If you do not name the device on which the package resides, the command checks the default spool directory (**`/var/spool/pkg`**). If the package is not there, installation fails. The name given after the **`-d`** option must be a full pathname to a device or directory, or the device alias (as shown in the example).

NOTE

You must use a package identifier if multiple versions co-reside on the installation medium. In most cases, there will be only one instance of a package on a medium and the package identifier will be the package abbreviation without a suffix.

> Be aware that the suffix of a package identifier defines the package instance on that particular medium. A new package identifier will be assigned this package when it has been installed on your system. (Run **pkginfo -d** *device* to find out which instances are available on a medium.)

pkgadd is a sophisticated command that will interact with you if it has a question about what it should do. For example, if you supply a device name, but not a package name or SIP, it will show you a list of packages or sets on the device (or in the directory) specified and ask you to choose one to install. In addition, it checks for a number of potential problems.

Interacting with pkgadd

When **pkgadd** encounters a problem, it first checks the admin file for instructions. If you have specified that the **check** admin file should be used, or if the instructions in any admin file say to consult the administrator (meaning a parameter is defined as `ask`), **pkgadd** gives you a message describing the problem and prompts you for a reply. (if the **default** admin file is used, no prompts will be displayed). The prompt is usually `Do you want to continue with this installation`. You should respond with `yes`, `no`, or `quit`. If you have given more than one package (such as `pkgA`, `pkgB`, and `pkgC` in the example above) and you respond with `no`, the current installation is stopped but **pkgadd** is instructed to continue with installation of the other packages. If you enter `quit`, **pkgadd** stops installation of all packages.

See "*Setting Installation Defaults*," earlier in this chapter, for a description of the potential problems for which **pkgadd** checks, and the default instructions you can define for each problem.

Installing a Package or Set with an Alternative Admin File

Unless you specify differently, **pkgadd** uses the admin file **default**. To inform **pkgadd** that you want to use an alternative admin file, invoke it with the **-a** option. For example, to install a package named **mypkg** using an admin file named **myadmin,** execute the following:

```
pkgadd -d ctape1 -a myadmin mypkg
```

Or, you may specify the **check** admin file, if you want more checking to be done:

```
pkgadd -a check pkginst
```

However, if you are installing a set, use the **default** admin file. **default** will be used automatically, in the absence of the **-a** *admin* option. For example:

```
pkgadd -d ctape1 sipname
```

is equivalent to:

```
pkgadd -d ctape1 -a default sipname
```

If you put your admin file in the **/var/sadm/install/admin** directory. you need only specify the filename as the argument to the **-a** option when executing **pkgadd**. You can create admin files in other directories but if you do, **-a** *admin* must specify the full pathname of the file.

Installing a Package or Set in Non-interactive Mode

The installation procedure described thus far has been interactive. You can also install packages or sets in non-interactive mode. In non-interactive mode you do not supply input to the **pkgadd** command or monitor its output.To install a package or set in non-interactive mode, use the **-n** option. For example, to install **mypkg** without interaction, enter:

```
pkgadd -d ctape1 -n mypkg
```

You may also consider using the **-q** option to suppress the list of files being installed and other non-essential information, and the **-l** option to log errors for later review.

Using a Response File When Installing a Package or Set

In some instances, non-interactive installation of a package or set requires a response file. This file holds answers to questions for which the installation script would prompt you to provide answers if you were installing the same package or set interactively. To create a response file, run the **pkgask** command as described in *"Storing Interactions with a Package or Set"*, earlier in this chapter. After storing your response to the package or set installation script, use your response file as input to the **pkgadd** command by executing a command line such as the following:

```
pkgadd -d device -n -r response pkginst
```

where *device* is the name of the device (or directory) on which the package or set currently resides, **-n** requests non-interactive installation, *response* is the file or directory you created with the **pkgask** command, and *pkginst* is the package identifier or SIP of the package to be installed.

For example, to install **mypkg** in non-interactive mode, supplying a response file named **myresponse**, enter:

```
pkgadd -d ctape1 -n -r /pkgA/myresponse mypkg
```

As shown here, the full pathname of the response file must be provided.

If **pkginst** is a package identifier (not a SIP), and **-r** names a specific response file, you can install only one package at a time. But, multiple packages can be installed with one command. To accomplish this, use the **-r** option to specify the name of a directory that holds several response files corresponding to the packages you want to install. Each response file must be given the same name as the package to which it corresponds. For example, if you enter:

```
pkgadd -d device -n -r response_dir pkgA pkgB pkgC
```

pkgadd will use *response_dir*/pkgA, *response_dir*/pkgB, and *response_dir*/pkgC to install pkgA, pkgB, and pkgC, respectively.

Troubleshooting Software Installation

The **pkgadd** command performs numerous checks as it installs a software package or set and, in a sense, performs troubleshooting for you. You will know a problem has occurred when:

- **pkgadd** displays a descriptive message. You should decide what actions to take based on the message. In most cases **pkgadd** will prompt you for instructions and wait. If the **-l** (log mode) option is specified no message will be displayed, rather, a log file containing error information is created.
- Installation results in a partially installed package. When this happens, you should attempt to reinstall the package.

Table 7-1 describes some common installation errors and illustrates that, in most cases, you will need to either answer a prompt, or attempt to reinstall the package or set to correct an error.

Determining the Status of a Package or Set

The status of a package or set will be either fully or partially installed. A package or set is considered fully installed, and is noted as such in the installation software database, when the system is notified that all actions needed to install the package or set have been executed successfully. If necessary actions have not taken place, or have been incomplete or unsuccessful, then the package is considered partially installed and is noted as such in the database.

The status field in the display created by executing **pkginfo -l** *pkginst* shows whether the package or set was fully or partially installed.

Reinstalling a Package

Any time a package or set installation results in partially installed software, you should attempt to reinstall the package or set. If you know the reason it was only partially installed, correct the problem first, and then reinstall.

NOTE

In some cases, you will have to remove a partially installed package before reinstalling it. If so, you will be notified when you execute **pkgadd**.

Table 7-1. Common Installation Errors

Problem:	Accidental or purposeful termination of installation
Cause:	Break key used, program terminated, system difficulties
Fix:	Results in a partially installed package or set. Reinstall.
Problem:	Ran out of space on system during installation
Cause:	Inadequate space
Fix:	Results in a partially installed package or set. Reinstall when space is available.
Problem:	Read error occurs during installation
Cause:	Corrupt media, hardware problems
Fix:	Attend to media or hardware problems. Attempt reinstallation.
Problem:	Content verification error
Cause:	Temporarily out of space, bad media, file changed by editing before check made
Fix:	Attempt reinstallation.
Problem:	Incorrect tape
Cause:	Installing tapes out of sequence or using a tape not part of the package or set being installed
Fix:	Will receive message requesting that the correct tape be inserted. Follow the instructions supplied by **pkgadd**.

Checking Installation Accuracy

You may want to check the integrity of a package after it has been installed on your system. To do this, run **pkgchk** with the **-n** option. This command determines whether an object has been modified by software or other actions since its installation, but the **-n** option indicates that volatile files should not be checked. The **-n** option should be used for most post-installation checking. The command line for checking a package named `pkgA` would look like this:

```
pkgchk -n pkgA
```

NOTE

If pkgA is a SIP, then the Set Installation Package itself will be checked, but the packages which are members of the set will not be checked. **pkgchk** checks packages, not sets of packages. If you want to check the packages that comprise a set, run **pkginfo** to determine the name of each package in the set, and then run **pkgchk** for the individual packages.

You can name more than one package on the command line by separating the package names with spaces.

You can check specific pathnames, instead of all components of a package, by using the **-p** option to name the paths you want to check. If **pkgchk** is not given any name at all, meaning no package names or pathnames, then it checks the entire contents of a computer. You can name more than one pathname as long as the names are separated by commas (with no white space).

NOTE

Packages must be identified by their package identifier. All instances of a package can be requested by adding .* to the package abbreviation. If you use the asterisk (*) in your command line, enclose the command line in single quotes or insert a backslash (\) before the asterisk to prevent the shell from interpreting the * character. The **PKGINST** field on the **pkginfo** display shows the package identifier.

Defining the Type of Accuracy Check

pkgchk performs two kinds of checks. It checks file attributes (the permissions, ownership, and security attributes of a file, and major/minor numbers for block or character special devices) and the file contents (the size, checksum, and modification date of a file). By default, the command checks both the file attributes and the file contents.You can check only the file attributes by using the **-a** option or only the file contents by using the **-c** option.

Checking Against the pkgmap File

The **pkgchk** command compares the file attributes and contents of the installed package against the installation software database. The entries for a package may have been changed since the time of installation. For example, another package may have changed a package component. The database will reflect that change.

If you want to compare the current integrity of a package to its integrity at the time it was originally installed, use the **-m** and **-e** options to specify the original description files. For example, if pkgA is mounted or spooled in the **/install** directory, the following command would be used.

```
pkgchk -m /install/pkgA/pkgmap \
    -e /install/pkgA/pkginfo pkgA
```

(This command should be entered on one line. It is shown here on two lines for the sake of legibility.)

Correcting Differences While Checking Accuracy

To correct file attributes when discrepancies are found, invoke **pkgchk** with the **-f** option. For example

```
pkgchk -n -f mypkg
```

attempts to correct any differences between the package components and the installation database or, if the **-m** and **-e** options have been used, between the components and the original **pkgmap** and **pkginfo** files. Again, the **-n** option should be specified so that no attempt is made to "correct" volatile files, which may have undergone legitimate changes as a result of system activity, and therefore should not be set back to their original state.

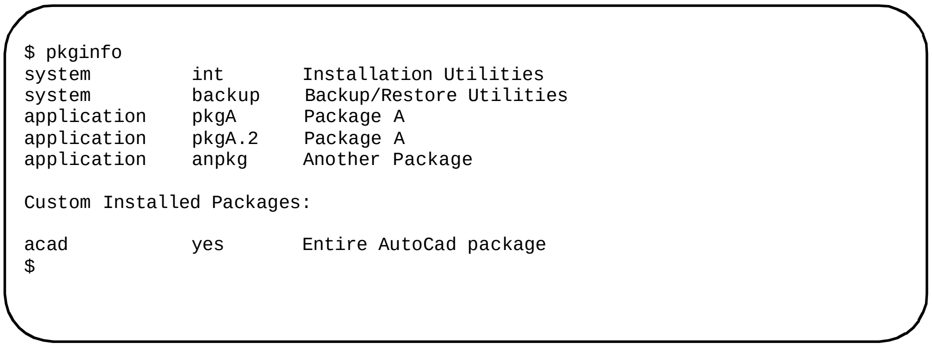
Showing Information about Installed Packages or Sets

You can display information about installed packages or sets by using the **pkginfo** command. It has a number of options that allow you to customize both the format and the contents of the display.

You can request any number of package instances if each name is separated by white space. If **pkginfo** is invoked with no packages named, it displays information for all completely installed packages on your system except for those whose category is "set." If the **pkginfo** command is used with the **-c** set option, then information will be displayed about Set Installation Packages.

The Default pkginfo Display

When **pkginfo** is executed without options, it displays the category, package instance, and package name of all packages (but not SIPs) that have been completely installed on your system. The display is organized by categories, as shown in the following example. (Categories are defined by the package developer.)



```
$ pkginfo
system         int       Installation Utilities
system         backup    Backup/Restore Utilities
application    pkgA      Package A
application    pkgA.2    Package A
application    anpkg     Another Package

Custom Installed Packages:

acad           yes       Entire AutoCad package
$
```

As shown in the last line of this example, the display for non-System V packages (for example, XENIX® packages) is different. For these custom installed packages, the abbreviated package name is shown, followed by the word `yes` (which indicates that the package is installed), and the full package name.

Displaying Set Installation Packages

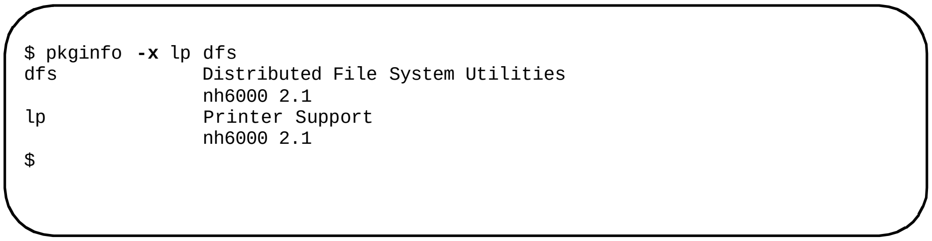
You can use the **`pkginfo`** command with the **`-c`** option, and specify the category as `set` to display information about Set Installation Packages (SIPs).

Customizing the Format of the pkginfo Display

You can get a **`pkginfo`** display in any of three formats: short, extracted, or long.

The short format is the default format shown previously. It shows only the category, package abbreviation, and full package name. It presents one line of information per package.

The extracted format shows the package abbreviation, package name, package architecture (if available), and package version (if available). Use the **`-x`** option to request the extracted format, as shown in the next example.



```
$ pkginfo -x lp dfs
dfs             Distributed File System Utilities
                nh6000 2.1
lp              Printer Support
                nh6000 2.1
$
```

If **`pkginfo`** is invoked to obtain information on set member packages located on tape media, all options are allowed since the information is readily available on the tape.

Using the **`-l`** option produces a display in the long format showing all of the available information about a package, as in the following example.

```
$ pkginfo -l lp
   PKGINST:  lp
      NAME:  Printer Support
  CATEGORY:  system
      ARCH:  nh6000
   VERSION:  2.1
   BASEDIR:
    VENDOR:  HCSC
    PSTAMP:  1.0 4/30/94
  INSTDATE:  Feb 18 1992 07:52
    STATUS:  completely installed
     FILES:  nh6000 installed pathnames
                 28 shared pathnames
                  6 linked files
                 41 directories
                 31 executables
                 11 setuid/setgid executables
               2661 blocks used (approx)
```

See the **pkginfo(4)** online manual page for complete details on these fields.

Customizing the Contents of the pkginfo Display

You can use the following **pkginfo** options to specify packages to be included in the display. Full details on these options are given on the **pkginfo(1)** online manual page.

-c Select packages based on their category membership.

-i Request that only completely installed packages be included in the display.

-p Request that only partially installed packages be included in the display.

-a Request that all packages with a particular architecture be included in the list.

-v Request that all packages with a particular version be included in the list.

-d Request that all spooled packages on a particular device, or in a particular directory, be included in the list.

NOTE

The **-p** and **-i** options cannot be used in conjunction with **-d**, because **-d** implies “all packages spooled on this device” (or in this directory). The **-a**, **-l** and **-v** options will not work for sets whose member packages span over several tapes.

Showing the Value of a Parameter

When you want to know the value of only one parameter of a package, run

```
pkgparam -v pkginst param
```

where *pkginst* is the package identifier and *param* is the name of the parameter you want displayed. *param* must match the parameter definition; in most cases this means it should be entered in all capital letters. You can name multiple parameters when executing **pkgparam** as long as they are separated by white space. The **-v** option requests the verbose format, which shows the parameter name and its value. The following example displays the value of the NAME parameter.

```
$ pkgparam -v lp NAME
NAME='Printer Support'
$
```

See **pkginfo(4)** online manual page for complete details on the parameters.

Storing Packages without Installing Them

When you store a package, you copy its components directly from an installation medium to a spool directory, without invoking any installation actions, such as running installation scripts or updating the installation software database.

Spooling a Package

To store packages without installing them, run the **pkgadd** command with the **-s** option. For example, to copy a package named **mypkg** from tape drive ctape1 to a spool directory named **/var/temp/spooldir**, run

```
pkgadd -d ctape1 -s /var/temp/spooldir mypkg
```

When you follow the **-s** option with the word **spool**, **pkgadd** copies the package into the default spool directory (**/var/spool/pkg**).

NOTE

You must use a package identifier with a numerical suffix if multiple versions reside on the installation medium. When there is only one version of a package on a medium, the package identifier is the package abbreviation without a suffix.

The package identifier suffix defines the package instance only on that particular medium. A new suffix will be assigned when this package is installed on your system.To find out what instances are available on a medium, run

pkginfo **`-d`** *device*

You cannot spool a set directly. Rather, you should use **`pkginfo`** to determine which packages are members of the set, verify that you have sufficient disk space to spool to, and then spool the package(s).

Checking the Accuracy of a Spooled Package

You can use this command to check the accuracy of a spooled package instead of an installed package by using the **`-d`** option and naming the directory into which the package was spooled (or the device onto which it was spooled). The **`pkgchk`** command will look in this directory (or on this device) and perform its check. For example,

```
pkgchk -d spooldir pkgA
```

looks in the spool directory **`spooldir`** and checks the accuracy of a package named `pkgA`.

The checks made for a spooled package are limited because not all information can be audited until a package is installed.

If `pkgA` is a SIP, then the Set Installation Package itself will be checked, but the packages which are members of the set will not be checked. **`pkgchk`** checks packages, not sets of packages. If you want to check the packages that comprise a set, run **`pkginfo`** to determine the name of each package in the set, and then run **`pkgchk`** for the individual packages.

Showing Information about Spooled Packages

You can request that all spooled packages on a particular device, or in a particular directory, be included in the **`pkginfo`** list by using the **`-d`** option. For example, to show information in the extracted format for all the packages in the spool directory **`/opt/spooldir`**, run:

```
pkginfo -d /opt/spooldir -x
```

Removing a Spooled Package

The **`-s`** option of the **`pkgrm`** command removes a package from the spool directory. For example, to remove `pkgA` from the spool directory **`/opt/spooldir`**, run

```
pkgrm -s /opt/spooldir pkgA
```

If you name only the spool directory (but no packages) all spooled packages will be removed from the named directory.

Removing Packages or Sets

The **`pkgrm`** command removes both fully and partially installed packages or sets from the system. It attempts to return the system to the same state in which it was running before the package or set was installed.

To remove a software package or SIP named **`mypkg`** from your system, enter

```
pkgrm mypkg
```

If a SIP is specified, all packages which are members of the set, and the SIP itself, are removed in reverse order from which they were installed.

You can remove multiple packages (which are not members of a set) with one command by separating package names with white space, as follows:

```
pkgrm pkgA.3 mypkgB mypkgC.2
```

NOTE

> Packages must be identified by their package instance, while sets must be identified by their SIP. All instances of a package can be requested by adding an asterisk (*) to the end of the package abbreviation. If you use the asterisk (*) in your command line, enclose the command line in single quotes or insert a backslash (\) before the asterisk to prevent the shell from interpreting the * character. The **`PKGINST`** field on the **`pkginfo`** display shows the package identifier.

To remove a package in non-interactive mode, use the **`-n`** option. If the package cannot be removed in this mode (because some interaction is required) **`pkgrm`** exits and removal fails.

Removing a Package or Set with an Alternative Admin File

Two of the parameters set in the admin file affect package or set removal. They are **`runlevel`** (which gives instructions for what to do when the run level is not met) and **`rdepend`** (which defines whether or not to check for other packages with dependencies on this package). You can have alternative admin files to assign different values to these two parameters.

`pkgrm` uses the admin file **`default`** unless you use the **`-a`** option to inform **`pkgrm`** that you want to use an alternative admin file. For example, to remove a package or set named **`mypkg`** using an admin file named **`myadmin`**, execute the following:

```
pkgrm -a myadmin mypkg
```

If you are removing an individual package you may specify the **`check`** admin file with a **`-a`** option.

```
pkgrm -a check pkginst
```

If you are removing a set you must use the **`default`** admin file. This happens automatically unless you specify an alternate admin file using the **`-a`** option. Thus, the proper way to remove a set is:

```
pkgrm sipname
```

which is equivalent to:

```
pkgrm -a default sipname
```

NOTE

If a SIP is specified, all packages which are members of the set, and the SIP itself, are removed in reverse order from which they were installed.

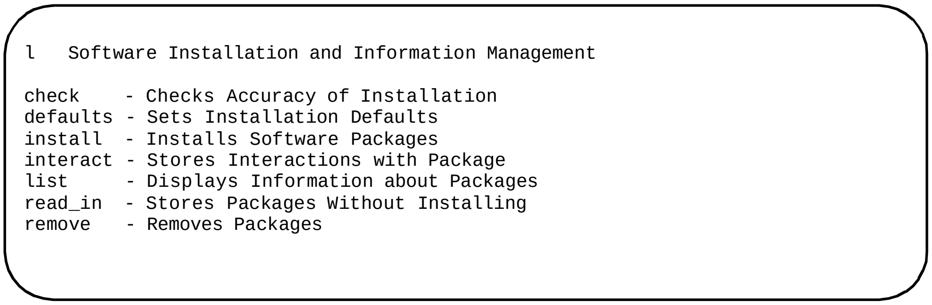
Unless you specify a full pathname after the **`-a`**, **`pkgrm`** looks for the admin file in the **`/var/sadm/install/admin`** directory.

Installing a Patch

Patches are distributed as packages. Each patch package provides object replacements for only one previously distributed package. The patch package is named according to the original package (the one being patched) followed by a patch revision number. For example, a patch package that replaces objects originally contained in the `base` package may be named `base-001`.

Installing Add-on Software through OA&M Menus

Tare he system administration menus are only available if the Operations, Administration and Maintenance (OA&M) package is installed on your system. To access the menu for installing and removing software, type **sysadm** software. The following menu will appear on your screen:



```
l   Software Installation and Information Management

check    - Checks Accuracy of Installation
defaults - Sets Installation Defaults
install  - Installs Software Packages
interact - Stores Interactions with Package
list     - Displays Information about Packages
read_in  - Stores Packages Without Installing
remove   - Removes Packages
```

Screen 7-2. Menu for Software Installation and Information Management

The following table shows how the tasks listed on the software menu correspond to the shell commands discussed throughout this chapter.

Task to Be Performed	**sysadm** Task	Shell Command
Set installation defaults	defaults	**vi(1) admin(4)**
Store interaction with packages or sets	interact	**pkgask(1M)**
Install software package or set	install	**pkgadd(1M)**
Check accuracy of installation	check	**pkgchk(1M)**
Show installed packages or sets	list	**pkginfo(1)**
Store packages or sets (not install them)	read_in	**pkgadd(1M)**
Remove packages or sets	remove	**pkgrm(1M)**

Details about the format of admin files are available in the online manual pages. For details about **vi**, and other commands refer to the online manual pages.

Quick Reference to Software Management

- Setting installation defaults:

  Installation defaults are defined in an admin file. This file can be created with any editor and should contain a list of parameter definitions in the format of *param=value*.

You can create this file in any directory; however, **pkgadd** normally looks in the **/var/sadm/install/admin** directory for admin files. If you put your admin file in this directory, you only need to supply the filename after the **-a** option when executing **pkgadd**. If you create your file in a different directory, you must supply the full pathname after the **-a**. The parameters that can be set in this file, along with their description and possible values, are shown below.

- basedir (package relocation information) directory name or **$PKG** or **$PKGINST** or **ask**
- **mail** (who should receive mail after installation) space-separated list of user IDs, defined as null (no mail sent), or not defined (mail is sent to root)
- runlevel (run level dependencies not met) options: **nocheck**, **quit**, **ask**
- conflict (name conflict because a file or directory with the same name already exists) options: **nocheck**, **quit**, **nochange**, **ask**
- setuid (check for set-UID or set-GID execution) options: **nocheck**, **quit**, **nochange**, **ask**
- action (check security impact of package scripts) options: **nocheck**, **quit**, **ask**
- partial (partially installed version exists) options: **nocheck**, **quit**, **ask**
- instance (instance already exists) options: **quit**, **overwrite**, **unique**, **ask**
- idepend (package dependencies not met at installation time) options: **nocheck**, **quit**, **ask**
- rdepend (package dependencies not met at removal time) options: **nocheck**, **quit**, **ask**
- space (disk space requirements not met) options: **nocheck**, **quit**, **ask**

- Storing interactions with a package or set:

  pkgask **-d** *device* **-r** *response pkginst*

  where *device* is the device on which the package is stored, *response* is the name of the file or directory in which your answers to questions from a package installation script will be written, and *pkginst* is the package identifier of the package, or the SIP of the set, to be installed. The file or directory created with this procedure should be used in the procedure to install a package or set in non-interactive mode.

- Installing a software package:

  pkgadd **-a** check **-d** *device pkginst* where *device* is the full pathname of the device or directory on which the package is stored and *pkginst* is the package identifier of the package to be installed. *device* can also be the device alias. You may also specify **`-a`** check if you want to install the package interactively.

- Installing a software set:

  ```
  pkgadd -d device pkginst
  ```

  where *device* is the full pathname of the device or directory on which the package is stored and *pkginst* is the SIP of the set to be installed. *device* can also be the device alias. The **`default`** admin file is automatically used.

- Installing a software package or set in non-interactive mode:

  ```
  pkgadd -n -d device pkginst
  ```

  where *device* is the name of the device on which the package is stored and *pkginst* is the package identifier of the package or the SIP of the set to be installed. The **`-n`** option requests the non-interactive mode.

- Installing a software package in non-interactive mode with a response file:

  ```
  pkgadd -n -d device -r response pkginst
  ```

  where *device* is the name of the device on which the package is stored, *pkginst* is the package identifier of the package, or the SIP of the set, to be installed; and *response* is the full pathname of the response file to be used during the installation process.

- Installing a software package or set and using an alternative admin file:

  ```
  pkgadd -d device -a adminfile pkginst
  ```

  where *device* is the name of the device on which the package is stored, *pkginst* is the package identifier of the package, or the SIP of the set, to be installed; and *adminfile* is the name of the alternative admin file to be used. **`pkgadd`** looks in the **`/var/sadm/install/admin`** directory for *adminfile* unless you supply a full pathname.

- Checking the installation accuracy of an installed package:

  ```
  pkgchk -n pkginst
  ```

  where *pkginst* is the package identifier of the package you want to check. You can name more than one package ID if the names are separated by white space. (If no package name is supplied, the entire contents of the computer will be checked.)

- Checking the installation accuracy of a specific pathname:

  ```
  pkgchk -n -p pathname
  ```

  where only *pathname* will be checked.You can name more than one pathname if the names are separated by commas.

- Checking the installation accuracy of only the file contents or only the file attributes of a package:

```
pkgchk [-a|-c] pkginst
```

where **-a** will check only the file attributes of a package and **-c** will check only the file contents. *pkginst* is the package identifier of the package to be checked.

- Checking the installation accuracy of a file against the original description files instead of the installation software database:

```
pkgchk -m /install/pkginst/pkgmap
-e /install/pkginst/pkginfo
```

where **-m** and **-e** check the package instance, *pkginst*, against the original **pkgmap** and **pkginfo** files from the package's distribution medium that are mounted or spooled at **/install**.

- Correcting file attributes as they are checked:

```
pkgchk -n -f pkginst
```

where *pkginst* is the package identifier of the package to be checked. When the **-f** option is used, the command attempts to correct any differences in the file attributes between the package and the installation software database.

- Spooling a package:

```
pkgadd -d device -s spooldir pkginst
```

where *device* is the name of the device on which the package is stored, *spooldir* is the name of the directory into which the package is to be spooled, and *pkginst* is the package identifier of the package to be installed.

- Checking the installation accuracy of a spooled package:

```
pkgchk -d spoolarea pkginst
```

where *spoolarea* is the name of the directory into which the package was spooled (or the device onto which it was spooled) and *pkginst* is the package identifier of the package. You can name more than one package if the names are separated by white space.

- Showing information about installed packages or sets:

```
pkginfo pkginst
```

where *pkginst* is the package identifier of the package for which you are requesting information. You can name any number of instances separated by spaces. The display shows the package category, package instance, and package name for the each instance requested. If *pkginst* is a SIP, information about its member packages is displayed, if available. Executing **pkginfo** without naming a package instance displays information about all fully installed packages (but not SIPs) on your system.

- Showing information about installed SIPs:

```
pkginfo -c set pkginst
```

where **-c** specifies the category **set**, and where *pkginst* is the SIP of the set for which you are requesting information.

- Showing information about spooled packages or packages on a particular device:

  ```
  pkginfo -d device
  ```

  where *device* is the name of a device, directory, or spool directory. Using the **-d** option with **pkginfo** displays information about all packages on that device or in that directory.

- Showing information about a spooled SIP or SIPs on a particular device:

  ```
  pkginfo -c set -d device
  ```

  where **-c** specifies the category **set**, and where *device* is the name of a device, directory, or spool directory. Using the **-d** option with **pkginfo -c** set displays information about all sets on that device or in that directory.

- Removing a spooled package:

  ```
  pkgrm -s spooldir [pkginst]
  ```

  where *spooldir* is the name of the spool directory and *pkginst* is the name of the package to be removed. If no package identifier is supplied, all packages spooled in the named directory will be removed.

- Customizing the format of a **pkginfo** display (example 1):

  ```
  pkginfo [-x|-l] pkginst
  ```

  where *pkginst* is the package identifier of the package, or packages, for which you are requesting information. The **-x** option requests the extracted display consisting of the package instance, package name, package architecture (if available), and package version (if available). The **-l** option requests the long display that consists of all available information about a package.

  Executing **pkginfo** without the **-x** or **-l** option causes the information to be displayed in the default format that shows the category, package instance, and full package name.

- Customizing the contents of a **pkginfo** display (example 2):

  ```
  pkginfo [-c category] [-i|-p] [-n] [-a arch]
      [-v version] pkginst
  ```

  where the options define the contents of the **pkginfo** display as follows:

  - **-c** *category* includes all packages in the category defined. specify **-c** set for information about sets.
  - **-i** includes only fully installed packages.
  - **-p** includes only partially installed packages.
  - **-a** *arch* includes all packages with the specified architecture.

  - **-v** *version* includes all packages with the specified version.

pkginst is the package identifier of the package(s) for which you are requesting information. The **-a**, **-l** and **-v** options will not work for sets whose member packages span over several tapes.

- Showing the value of only one parameter:

  `pkgparam` **`-v`** *pkginst param*

  where *pkginst* is the package identifier and *param* is the name of the parameter, or parameters, you want displayed. Separate multiple parameters by white space. **-v** shows the parameter value with a label (for example, **BASEDIR='/opt/pkgAdir'**). Without the **-v** option, only the parameter value is shown.

- Storing a package without installing it:

  `pkgadd` **`-d`** *device* **`-s`** *spooldir pkginst*

  where *device* is the name of the device on which the package is stored, *spooldir* is the name of the directory into which the package should be read, and *pkginst* is the package identifier of the package to be read.

- Removing a package or set:

  `pkgrm` *pkginst*

  where *pkginst* is the package identifier of the package or the SIP of the set to be removed. When *pkginst* is a SIP, all packages which are members of the set, and the SIP itself, are removed in reverse order from which they were installed.

- Removing a software package or set in non-interactive mode:

  `pkgrm` **`-n`** *pkginst*

  where *pkginst* is the package identifier of the package to be removed. The **-n** option requests the non-interactive mode.

- Removing a software package using an alternative admin file:

  `pkgrm` **`-a`** *admin pkginst*

  where *pkginst* is the package identifier of the package to be removed and *admin* is the name of the alternative admin file. Only the **default** admin file should be used for set removal. **pkgrm** looks in the **/var/sadm/install/admin** directory for the admin file unless you supply a full pathname.

8
Directories and Files

8
Directories and Files

Introduction

This chapter describes:

- directories and files that are important for administering a system
- directories that are new for this software release
- the reorganization of the directory structure introduced in this release
- the new organization of the root file system, and significant directories mounted on root

NOTE

To maintain a secure environment, do not change the file or directory permissions from those assigned at the time of installation.

Directories in root

The **/** (root) file system contains executables and other files necessary to boot and run the system. The directories of the root file system are explained next.

/bck

The **/bck** directory is used to mount a backup file system for restoring files.

/dev

The **/dev** directory contains block and character special files that are usually associated with hardware devices or STREAMS drivers.

/etc

The **/etc** directory contains machine-specific configuration files and system administration databases.

/export

The **/export** directory contains the default root of the exported file system tree.

/home

The **/home** directory contains user directories.

/home2

The **/home2** directory contains user directories.

/install

The **/install** directory is used by the **sysadm** command to mount utilities packages for installation and removal (**/install** file system).

/lost+found

The **/lost+found** directory is used by **fsck** to save disconnected files and directories.

/mnt

The **/mnt** directory is used to mount file systems for temporary use.

/opt

The **/opt** directory is the mount point from which add-on application packages are installed.

/proc

The **/proc** directory is the mount point of the **proc** file system which provides information on the system's processes.

/save

The **/save** directory is used by the **sysadm** command.

/sbin

The **/sbin** directory contains executables used in the booting process and in manual recovery from a system failure.

/stand

/stand directory contains the standalone (bootable) programs and data files necessary for the system boot procedure.

/tmp

The **/tmp** directory contains temporary files. If the Enhanced Security Utilities are installed, **/tmp** is a multilevel directory.

/usr

The **/usr** directory is the mount point of the **usr** file system.

/var

The **/var** directory is the mount point of the **var** file system. It contains those files and directories that vary from machine to machine, such as **tmp**, **spool**, and **mail.** The **/var** file system also contains administrative directories such as **/var/adm** and **/var/opt,** the latter of which is installed by application packages.

Directories in /etc

This section describes the directories under the **`/etc`** directory, which contain machine-specific configuration files and system administration databases.

/etc/bkup/method

This directory contains files that describe all backup and restore methods currently used on your computer.

/etc/conf

This directory contains files and programs used to reconfigure the system.

/etc/conf/mod.d

This directory contains the modules used when booting a system with dynamically loadable kernel modules.

/etc/cron.d

This directory contains administrative files for controlling and monitoring **`cron`** activities.

/etc/default

This directory contains files that assign default values to certain system parameters.

/etc/init.d

This directory contains executable files used in upward and downward transitions to all system states. These files are linked to files beginning with `S` (start) or `K` (stop) in **`/etc/rc`***n*.d, where *n* is the appropriate system state. Files are executed from the **`/etc/rc`***n*.d directories.

/etc/lp

This directory contains the configuration files and interface programs for the LP Print Service.

/etc/mail

This directory contains files used in administering the electronic mail system.

/etc/mail/lists

This directory contains files, each of which contains a mail alias. The name of each file is the name of the mail alias that it contains. [See the **`mailx(1)`** command for a description of the mail alias format.]

/etc/rc.d

This directory contains executable files that perform the various functions needed to initialize the system to system-state 2. The files are executed when **`/usr/sbin/rc2`** is run.

/etc/rc0.d

This directory contains files executed by **`/usr/sbin/rc0`** for transitions to system states 0 and 6. Files in this directory are linked from the **`/etc/init.d`** directory, and begin with either a `K` or an `S.` `K` shows processes that are stopped; `S`, processes that are started when entering system states 0 or 6.

/etc/rc1.d

This directory contains files executed by **`/usr/sbin/rc1`** for transitions to system state 1. Files in this directory are linked from the **`/etc/init.d`** directory, and begin with either a `K` or an `S`. `K` shows processes that should be stopped; `S`, processes that should be started when entering system state 1.

/etc/rc2.d

This directory contains files executed by **`/usr/sbin/rc2`** for transitions to system state 2 (multi-user state). Files in this directory are linked from the **`/etc/init.d`** directory, and begin with either a `K` or an `S`. `K` shows processes that should be stopped; `S`, processes that should be started when entering system state 2.

/etc/rc3.d

This directory contains files executed by **`/usr/sbin/rc3`** for transitions to system state 3 (networking state). Files in this directory are linked from the **`/etc/init.d`** directory, and begin with either a `K` or an `S`. `K` shows processes that should be stopped; `S`, processes that should be started when entering system state 3.

/etc/saf

This directory contains files and subdirectories used by the Service Access Facility. The following commands in **`/usr/sbin`** use **`/etc/saf`** subdirectories for data storage and retrieval: **`nlsadmin`**, **`pmadm`**, and **`sacadm`**. The following files are included:

`_sactab` A list of port monitors to be started by the Service Access Controller (SAC). Each port monitor listed in this table has a **`_pmtab`** file in the **`/etc/saf/`***pmtag* directory, where *pmtag* is the tag of this port monitor (such as **`/etc/saf/tcp`** for the tcp port monitor).

`_sysconfig` The configuration script used to modify the environment for the Service Access Facility.

/etc/save.d

This directory contains files used by the **`sysadm`** command for backing up data. The following files are included:

`except` A list of the directories and files that should not be copied as part of a backup is maintained in this file.

`timestamp/...` The date and time of the last backup (volume or incremental) is maintained for each file system in the **`/etc/save.d/timestamp`** directory.

/etc/security

This directory contains subdirectories for audit, executable file privileges, Trusted Facility Management (TFM) privileges and a subdirectory for each of the enhanced security features. The subdirectories contain databases which the system uses to enforce security policy. The subdirectory **`/etc/security/mac`** also contains the history files in which the system records changes to security level definitions.

/etc/shutdown.d

This directory is maintained only for compatibility reasons.

/etc/skel

This directory contains the files and directories built when using the **`useradd`** command with the **`-m`** argument. All directories and files under this location are built under the `$HOME` location for the new user.

Files in /etc

The following files are used in machine-specific configuration and system administration databases.

/etc/bkup/bkexcept.tab

This file contains a list of files to be excluded from an incremental backup.

/etc/bkup/bkhist.tab

This file contains information about the success of all backup attempts.

/etc/bkup/bkreg.tab

This file contains instructions to the system for performing backup operations on your computer.

/etc/bkup/bkstatus.tab

This file contains the status of backup operations currently taking place.

/etc/bkup/rsmethod.tab

This file contains descriptions of the types of objects that may be restored using the full or partial restore method.

/etc/bkup/rsnotify.tab

This file contains the electronic mail address of the operator to be notified whenever restore requests require operator intervention.

/etc/bkup/rsstatus.tab

This file contains a list of all restore requests made by users of your computer.

/etc/bkup/rsstrat.tab

This file specifies a strategy for selecting archives when handling restore requests. In completing restore operations for these requests, the backup history log is used to navigate through the backup tape to find the desired files and or directories.

/etc/d_passwd

This optional file contains a list of programs that will require dial-up passwords when run from **`login`**. This file will be present if the system administrator has created it as a means of requiring an additional password that must be entered when users attempt to login from the devices found in **`/etc/dialups.`**

Each line in the file is formatted as

program`:`*encrypted_password*`:`

where *program* is the full path to any programs into which a user can log in and run. The password referred to in the *encrypted_password* is the one that will be used by the dial-up password program. This password must be entered before the user is given the login prompt. It is used in conjunction with the file **`/etc/dialups`**. (See Chapter 14, “User Account and Group Management” in *System Administration* for details.)

/etc/default/audit

This file may contain the following parameters that control auditing:

`AUDIT_PGM` Program that will be executed if there is a log switch.

AUDIT_NODE Node name to be appended to the audit event file name.

AUDIT_DEFPATH Pathname of the directory for audit event log files. The value for this parameter in the distributed system is **/var/audit.**

AUDIT_LOGERR Action taken if there is an write error to the audit event log file. The allowable values are DISABLE, which disables the auditing subsystem, and SHUTDOWN, which shuts the computer system down to firmware mode. The value for this parameter in the distributed system is SHUTDOWN.

AUDIT_LOGFULL Action taken when the audit event log file becomes full. The allowable values are DISABLE, which disables the auditing subsystem, SHUTDOWN, which shuts the computer system down to firmware mode, and SWITCH, which switches to an alternate event log file. The value for this parameter in the distributed system is SHUTDOWN.

/etc/default/cron

This file contains the following parameters for the **cron** command.

CRONLOG Enables (YES) or disables (NO) **cron** logging. If CRONLOG is set to YES, the **cron** log file is **/var/cron/log.**

BACKUP Names a file to which the contents of the **cron** log file will be copied when it reaches SIZE. BACKUP defaults to **/var/cron/olog** if not defined here.

LINES Number of lines to keep in the **cron** log file after a copy has been made. LINES defaults to 100 if not defined here.

SIZE Maximum size of the **cron** log file before it gets copied. SIZE defaults to 50,000 bytes if not defined here.

/etc/default/login

This file may contain the following parameters that define a user's login environment:

ALTSHELL Alternate shell status available to users (yes or no).

CONSOLE Root login allowed only at the console terminal.

DISABLETIME Number of seconds to sleep after a failed login.

HZ Number of clock ticks per second.

IDLEWEEKS Number of weeks a password may remain unchanged before the user is denied access to the system.

LOGFAILURES Number of failed login attempts allowed.

`MANDPASS` Mandatory password required for logins. The original value is no. If the value is `yes`, any attempt to log in to an account that does not have a password will fail. This supersedes the effect of `PASSREQ`. Lowest level allowed to log in to the system.

`MAXTRYS` Maximum number of login attempts permitted.

`PASSREQ` Password requirement on logins (`yes` or `no)`.

`PATH` User's default `PATH`.

`SLEEPTIME` Number of seconds to sleep before printing an error message.

`SUPATH` Root's default `PATH`.

`SYS_LOGIN_HIGH` Highest level allowed to log in to the system.

`SYS_LOGIN_LOW` Lowest level allowed to log in to the system.

`TIMEOUT` Number of seconds allowed for logging in before a timeout occurs.

`TIMEZONE` Time zone used within the user's environment.

`ULIMIT` File size limit (`ulimit`).

`UMASK` User's value for `umask`.

/etc/default/passwd

This file contains the following information about the length and aging of user passwords, and the availability of the **`passwd`** command for changing user passwords:

`MINWEEKS` Minimum number of weeks before a password can be changed.

`MAXWEEKS` Maximum number of weeks a password can be unchanged.

`PASSLENGTH` Minimum number of characters in a password.

`WARNWEEKS` Number of weeks before a password expires that the user is to be warned.

`LOGIN_ONLY` If this parameter exists and its value is equal to `Yes,` an unprivileged user can change passwords only by invoking the **`login`** command with the **`-p`** option. If it does not exist, or its value is anything other than the string `Yes` (including NULL), an unprivileged user can change passwords using the **`passwd`** command. `LOGIN_ONLY` is set to `Yes` by default if the Enhanced Security Utilities are installed; otherwise, it is not set.

/etc/default/privcmds

This file contains parameters whose values affect the generation and validation of check sums by the **filepriv** and **initprivs** commands, respectively:

GEN_CKSUM

If the value of this parameter is **No**, then the **filepriv** command will not generate a check sum value for the Privilege Data File (PDF) located in **/etc/security/tcb/privs;** this results in faster performance compared to generating the check sum value each time the command is run. If the value of this parameter is anything other than **No** (including **NULL**, the default), then the **filepriv** command generates a check sum each time it is run.

VAL_CKSUM

If the value of this parameter is **No,** then the **initprivs** command will not validate the checksum value stored in the Privilege Data File (PDF) located in **/etc/security/tcb/privs.** This results in faster performance compared to validating the check sum value each time the command is run. If the value of this parameter is anything other than **No** (including **NULl**), then the **initprivs** command validates the checksum for each file each time it is run. The default value for this parameter is **No**.

VAL_SIZE

If the value of this parameter is **No**, then the **initprivs** command will not validate the file size value stored in the Privilege Data File (PDF) located in **/etc/security/tcb/privs**. If the value of this parameter is anything other than **No** (including **NULL**), then the **initprivs** command validates the file size for each file each time it is run. The default value for this parameter is **Yes**.

VAL_VALIDITY

If the value of this parameter is **No**, then the **initprivs** command will not validate the ctime value stored in the Privilege Data File (PDF) located in **/etc/security/fcb/privs**. The ctime for a file is the time of the last file status change. A file’s ctime is changed by the following system calls: **chmod(2)**, **chown(2)**, **creat(2)**, **link(2)**, **mknod(2)**, **pipe(2)**, **unlink(2)**, **utime(2)**, and **write(2)**. If the value of this parameter is anything other than **No** (including **NULL**), then the **initprivs** command validates the ctime for each file each time it is run. The default value for this parameter is **No**.

On systems where the Super User (SUM) privilege policy module is enabled, disabling the ctime validity checking in **initprivs** also disables the ctime validity checking performed by the kernel when a file is executed with **exex(2)**.

/etc/default/sh

This file (which does not exist by default) contains a value for the following parameter affecting the use of the shell **`/sbin/sh:`**

`TIMEOUT` An integer value that specifies the number of seconds that can elapse without user activity before a shell will exit. If this value is `0`, undefined, or the file **`/etc/default/sh`** does not exist (the default), the shell will wait for user input until it is explicitly terminated.

/etc/default/su

This file contains values for the following parameters affecting the use of the **`su`** command:

`SULOG` The pathname of a file in which you can log all attempts to execute **`su`**.

`CONSOLE` If a user executes **`su`** to become a privileged user on a device other than *device*, a printed message appears on *device* to inform the administrator of that fact.

`PATH` When a user executes **`su`** to become a privileged user, the user's path is set to *path_list*. The default is **`/usr/bin:/usr/ccs/bin`**.

`SUPATH` When a user executes **`su`** to become a privileged user, the user's path is set to *path_list*. The default is **`/sbin:/usr/sbin:/usr/bin:/etc:/usr/ccs/bin.`**

`PROMPT` If this parameter exists and is set to `No`, the **`su`** command does not prompt for a password (even if one is defined for the *login_name*). The invoking user, however, must still have appropriate privilege to execute **`su`** successfully. If this parameter does not exist, or is set to anything other than `No` (including `NULL`), **`su`** prompts for a password when invoked and validates the password (if one is defined for *login_name*). If the Enhanced Security Utilities are installed, `PROMPT` is set to `No` by default; otherwise, it is not set.

/etc/default/tar

This file contains device names with corresponding block and size values to be used by the **`tar(1)`** command. Each line in the file has the form:

`archive`*n*=*device*   *block*   *size*

where *n* is a single digit integer that can be used as the value of # in the **`tar`** command's #`s` modifier, *device* is the device name, modifier, *device* is the device name, and *block* and *size* are integers used as the blocking factor and tape size, respectively.

For example, the first line in the file might look like this:

```
archive0=/dev/rdsk/1s018         360
```

/etc/default/ttymonxp

This file contains defaults used by the **`ttymon`** command for defining the Secure Attention Key (SAK) for a device. (SAK processing is in effect only if the Enhanced Security Utilities are installed.) The file contains information in this form

```
DEVICE_NAME=SAKTYPE:SAKDEF:SAKSEC
```

where `SAKTYPE` indicates the type of SAK representative, `SAKDEF` is the SAK representative, and `SAKSEC` is the secondary SAK.

/etc/default/useradd

This file may contain the following parameters that provide default values for the **`useradd`** command:

`SHELL`	full pathname of user's login shell
`HOMEDIR`	base directory in which to create user's home directory
`SKELDIR`	directory that contains skeleton information (such as a **`.profile`** file) to be copied to the user's home directory
`GROUPID`	default group ID
`INACT`	the maximum number of days allowed between uses of a login name
`EXPIRE`	the date on which a login name can no longer be used
`DEFLVL`	user's default MAC level; valid only if the Enhanced Security Utilities are installed and running on the system
`FORCED_PASS`	a password generator, defined for the user
`AUDIT_MASK`	default user audit mask; valid only if the Auditing Utilities are installed and running on the system

/etc/default/userdel

This file may contain the following parameter:

`UIDAGE`	amount of time (in months) before a deleted user ID can be reassigned

/etc/device.tab

This file is the device table. It lists the device alias, path to the vnode, and special attributes of every device connected to the computer.

/etc/devlock.tab

This file is created at run time and lists the reserved (locked) devices. Device reservations do not remain intact across system reboots.

/etc/saf/pmtag/_config

This file contains a configuration script used to customize the environment for the port monitor tagged as *pmtag* (such as **/etc/saf/tcp/_config** for the tcp port monitor). Port monitor configuration scripts are optional.

/etc/dgroup.tab

This file lists the group or groups to which a device belongs.

/etc/dialups

This file contains a list of terminal devices that cannot be accessed without a dial-up password. This file will be present if the system administrator has created it as a means of requiring an additional password that must be entered when users attempt to login from the devices listed in it. It is used in conjunction with the file **/etc/d_passwd.** (See Chapter 14, “User Accounts and Group Management” in the “Security Administration” part of this book for more information on creating dial-up passwords.)

/etc/group

This file describes each user group to the system. An entry is added for each new group with the **groupadd** command.

/etc/inittab

This file contains instructions for the **/sbin/init** command. The instructions define the processes created or stopped for each initialization state. Initialization states are called system states or run states. By convention, system state 1 (or S or s) is single-user state;

system states 2 and 3 are multi-user states. Chapter 3, “Booting and System States” of *System Administration* summarizes the various system states and describes their uses. [See **`inittab(4)`**.]

/etc/mail/mailcnfg

This file permits site-specific customization of the mail subsystem. See **`mailcnfg(4)`** and the “The Mail Service” chapter of *Network Administration* for details.

/etc/mail/mailsurr

This file lists actions to be taken (such as routing translations and logging) when mail containing particular patterns is processed by **`mail`**. See **`mailsurr(4)`** and the “The Mail Service” chapter in *Network Administration.*

/etc/mail/mailx.rc

This file contains defaults for the **`mailx`** program. It may be added by the system administrator. See **`mailx(1)`**.

/etc/mail/notify and /etc/mail/notify.sys

These files are used by the **`notify`** program to determine the location of users in a networked environment and to establish systems to use in case of file error.

/etc/motd

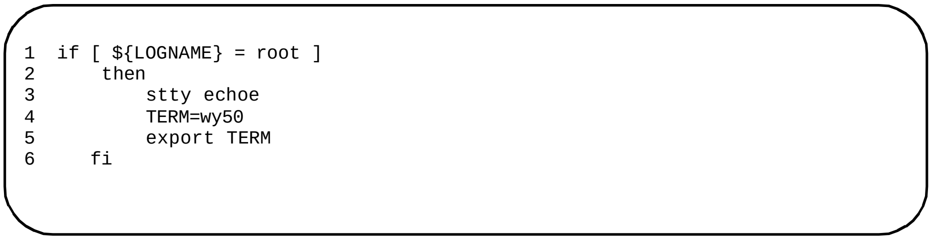
This file contains the message of the day. The message of the day is displayed on a user's screen after that user has successfully logged in. (The commands that produce this output on the screen are in the **`/etc/profile`** file.) This message should be kept short and to the point. The **`/var/news`** files should be used for lengthy messages.

/etc/passwd

This file identifies each user to the system. An entry is automatically added for each new user with the **`useradd`** command, removed with the **`userdel`** command, and modified with the **`usermod`** command.

/etc/profile

This file contains the default profile for all users. The standard (default) environment for all users is established by the instructions in the **/etc/profile** file. The system administrator can change this file to set options for the **root** login. For example, the six lines of code shown in Screen 8-1 can be added to the **/etc/profile.** This code defines the erase character, automatically identifies the terminal type, and sets the TERM variable when the login name is **root**.



```
if [ ${LOGNAME} = root ]
    then
        stty echoe
        TERM=wy50
        export TERM
   fi
```

Screen 8-1. Excerpt from /etc/profile

/etc/rfs/rmnttab

This file is created by the **rmount(1M)** command. This file contains a listing of unsuccessfully mounted resources or disconnected resources. These resources are polled by the **rmnttry(1M) cron** entry.

/etc/dfs/dfstab

This file specifies the Remote File Sharing resources or network file system resources from your machine that are automatically shared to remote machines when entering networking mode (system state 3). Each entry in this file should be a **share(1M)** command line.

/etc/saf/pmtag/_pmtab

This is the administrative file for the port monitor tagged as *pmtag*. It contains an entry for each service available through the *pmtag* port monitor.

/etc/saf/_sactab

This file contains information about all port monitors for which the Service Access Controller (SAC) is responsible.

/etc/saf/_sysconfig

This file contains a configuration script to customize the environments for all port monitors on the system. This per-system configuration file is optional.

/etc/TIMEZONE

This file sets the time zone shell variable `TZ`. The `TZ` variable is initially established for the system via the **`sysadm setup`** command. To change the `TZ` variable in the **`TIMEZONE`** file, execute the **`sysadm datetime`** command or edit the file **`/etc/TIMEZONE`** (with **`vi`** or another screen editor). The `TZ` variable can be redefined on a user (login) basis by setting the variable in the associated **`.profile`**. The **`TIMEZONE`** file is executed by **`/usr/sbin/rc2.`** [See **`timezone(4)`**.]

/etc/ttydefs

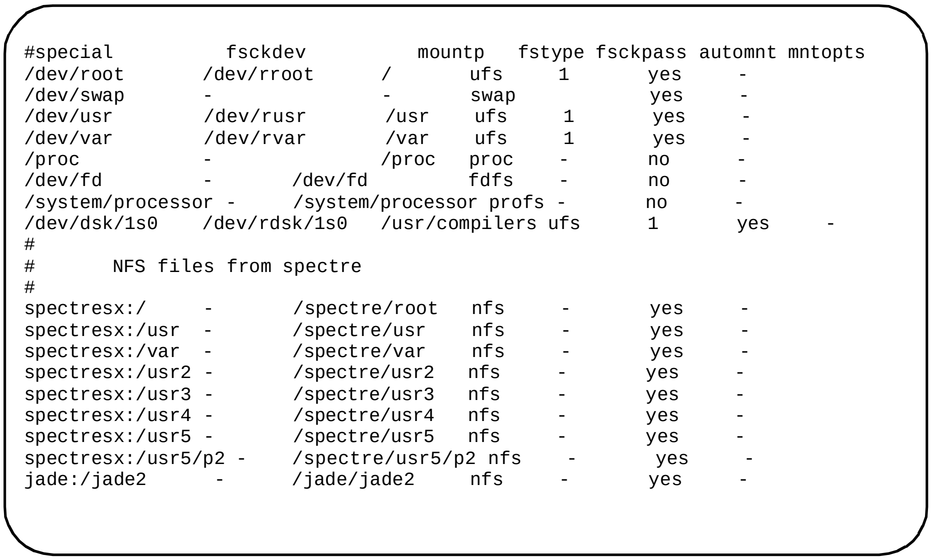
This file contains information used by ttymon port monitor to set the terminal modes and baud rate for a TTY port. (See Chapter 5, “Managing Ports” in *System Administration* for more information.)

/etc/vfstab

This file provides default values for file systems and remote resources. The following information can be stored in this file:

- the block and character devices on which file systems reside
- the resource name
- the location where a file system is usually mounted
- the file system type
- information on special mounting procedures

These defaults do not override command line arguments that have been entered manually. [See **`mountall(1M)`**.] Screen 8-2 shows a sample of this file.



```
#special          fsckdev          mountp   fstype fsckpass automnt mntopts
/dev/root       /dev/rroot      /       ufs     1       yes     -
/dev/swap       -               -       swap            yes     -
/dev/usr        /dev/rusr       /usr    ufs     1       yes     -
/dev/var        /dev/rvar       /var    ufs     1       yes     -
/proc           -               /proc   proc    -       no      -
/dev/fd         -       /dev/fd         fdfs    -       no      -
/system/processor -     /system/processor profs -       no      -
/dev/dsk/1s0    /dev/rdsk/1s0   /usr/compilers ufs      1       yes     -
#
#       NFS files from spectre
#
spectresx:/     -       /spectre/root   nfs     -       yes     -
spectresx:/usr  -       /spectre/usr    nfs     -       yes     -
spectresx:/var  -       /spectre/var    nfs     -       yes     -
spectresx:/usr2 -       /spectre/usr2   nfs     -       yes     -
spectresx:/usr3 -       /spectre/usr3   nfs     -       yes     -
spectresx:/usr4 -       /spectre/usr4   nfs     -       yes     -
spectresx:/usr5 -       /spectre/usr5   nfs     -       yes     -
spectresx:/usr5/p2 -    /spectre/usr5/p2 nfs    -       yes     -
jade:/jade2      -      /jade/jade2     nfs     -       yes     -
```

Screen 8-2. Sample `/etc/vfstab` File

Directories in /usr

This section describes the directories in the **`/usr`** file system. The **`/usr`** file system contains architecture-dependent and architecture-independent files and system administration databases that can be shared.

/usr/bin

This directory contains public commands and system utilities.

/usr/include

This directory contains public header files for C programs.

/usr/lib

This directory contains public libraries, daemons, and architecture dependent databases.

/usr/lib/lp

This directory contains the directories and LP specific utilities used in processing requests to the LP Print Service.

/usr/lib/mail

This directory contains directories and files used in processing mail.

/usr/lib/mail/surrcmd

This directory contains programs necessary for mail surrogate processing.

/usr/sadm/bkup/bin

This directory contains executables for the backup and restore services.

/usr/sbin

This directory contains executables used for system administration.

/usr/share

This directory contains architecture independent files that can be shared.

/usr/share/lib

This directory contains architecture independent databases.

Files in /usr

This section describes the files in the **`/usr`** directories, which contain architecture-dependent and architecture-independent files and system administrative databases that can be shared.

/usr/sbin/rc0

This file contains a shell script executed by **`/usr/sbin/shutdown`** for transitions to single-user state, and by **`/sbin/init`** for transitions to system states 0, and 6. Files in the **`/etc/rc0.d`** directory are executed when **`/usr/sbin/rc0`** is run. The file `K00ANNOUNCE` in **`/etc/rc0.d`** prints the message `System services are now being stopped`. Any task that you want executed when the system is taken to system states 0, s, or 6 is done by adding a file to the **`/etc/rc0.d`** directory.

/usr/sbin/rc1

This file contains a shell script executed by **`/sbin/init`** for transitions to system state 1. Executable files beginning with `S` or `K` in the **`/etc/rc1.d`** directories are executed when **`/usr/sbin/rc1`** is run. All files in **`rc1.d`** are linked from files in the **`/etc/init.d`** directory. Other files may be added to the **`/etc/rc1.d`** directory as a function of adding hardware or software to the system.

/usr/sbin/rc2

This file contains a shell script executed by **`/sbin/init`** for transitions to system state 2 (multi-user state). Executable files in the **`/etc/rc.d`** directory and any executable files beginning with `S` or `K` in the **`/etc/rc2.d`** directories are executed when **`/usr/sbin/rc2`** is run. All files in **`rc2.d`** are linked from files in the **`/etc/init.d`** directory. Other files may be added to the **`/etc/rc2.d`** directory as a function of adding hardware or software to the system.

/usr/sbin/rc3

This file is executed by **`/sbin/init.`** It executes the shell scripts in **`/etc/rc3.d`** for transitions to networking state (system state 3).

/usr/sbin/rc6

This shell script is run for transitions to system state 6 (for example, using **`shutdown -i6`**). If the operating system needs to be reconfigured, the **`/etc/conf/bin/idcpunix`** script is run, and, if the reconfiguration is successful, **`/usr/sbin/rc6`** reboots the operating system without running diagnostics. If the reconfiguration is unsuccessful, a shell is spawned.

/usr/sbin/shutdown

This file contains a shell script to shut down the system gracefully in preparation for a system backup or scheduled downtime. After stopping all nonessential processes, the **`shutdown`** script executes files in the **`/etc/shutdown.d`** directory by calling **`/usr/sbin/rc0`** for transitions to system state 1. For transitions to other system states, the **`shutdown`** script calls **`/sbin/init.`**

/usr/share/lib/mailx/C/mailx.help and /usr/share/lib/mailx/C/mailx.help.~

Help files for **`mailx`**. The file **`mailx.help.~`** (note that the last character in this filename is a tilde) contains help messages for the tilde commands available with **`mailx`**. See **`mailx(1)`**.

Directories in /var

This section describes the directories of the **`/var`** directory, which contain files and directories that vary from machine to machine.

/var/adm

This directory contains system logging and accounting files. It also contains the **`dumpdates`** file used by **`fsdump(1M)`**

/var/audit

This directory contains files of audit records.

/var/audit/auditmap

This directory contains data files used by audit reporting functions.

/var/crashfiles

This directory contains system dumps.

/var/cron

This directory contains the **`cron`** log file.

/var/lp

This directory contains log files for the LP Print Service.

/var/mail

This directory contains subdirectories and mail files that users access with the **`mail(1)`** and **`mailx(1)`** commands. If the Enhanced Security Utilities are installed, **`/var/mail`** is a multilevel directory.

/var/mail/:saved

This directory contains temporary storage for mail messages while **`mail`** is running. Files are named with the user's login name while they are in **`/var/mail.`**

/var/news

This directory contains news files. The file names are descriptive of the contents of the files; they are analogous to headlines. When a user reads the news, using the **`news`** command, an empty file named **`.news_time`** is created in his or her login directory. The date (time) of this file is used by the **`news`** command to determine if a user has read the latest news file(s).

/var/opt

This directory is created and used by application packages.

/var/options

This directory contains a file (or symbolic link to a file) that identifies each utility installed on the system. This directory also contains information created and used by application packages (such as temporary files and logs).

/var/preserve

This directory contains backup files for **vi** and **ex**.

/var/sadm

This directory contains logging and accounting files for the backup and restore services, software installation utilities, and package management facilities.

/var/sadm/install/logs

This directory contains the package installation log files.

/var/sadm/pkg

This directory contains data directories for installed software packages.

/var/saf

This directory contains log files for the Service Access Facility.

/var/spool

This directory contains temporary spool files.

/var/spool/cron/crontabs

This directory contains **crontab** files for the **adm**, **root**, and **sys** logins. If the Enhanced Security Utilities are installed, **/var/spool/cron/crontabs** is a multilevel directory. Users whose login names are in the **/etc/cron.d/cron.allow** file can establish their own **crontab** file using the **crontab** command. If the **cron.allow** file does not exist, the **/etc/cron.d/cron.deny** file is checked to determine if the user should be denied the use of the **crontab** command.

As **root**, you can use the **crontab** command to make the desired entries. Revisions to the file take effect at the next reboot. The file entries support the calendar reminder service and the Basic Networking Utilities. Remember, you can use the **cron** function to decrease the number of tasks you perform with the **sysadm** command; include recurring and habitual tasks in your **crontab** file. [See **crontab(1)**.] [See **crontab(1)**.]

/var/spool/lp

This directory contains the work area for the LP Print Service.

/var/spool/smtpq

This directory contains Simple Mail Transfer Protocol (SMTP) directories and log files. If the Enhanced Security Utilities are installed, **/var/spool/smtpq** is a multilevel directory. Directories named ***host*** contain messages spooled to be sent to that host. Files named **LOG.*n*** contain the logs from the past seven days (Sunday's log is called **log.0**). The current day's log is simply LOG.

/var/spool/uucp

This directory contains files to be sent by **uucp**. If the Enhanced Security Utilities are installed, **/var/spool/uucp** is a multilevel directory.

/var/spool/uucppublic

This directory contains files received by **uucp**.

/var/spool/uucppublic/receive

This directory contains files received by **uucp**. If the Enhanced Security Utilities are installed, **/var/spool/uucppublic/receive** is a multilevel directory.

/var/tmp

This directory contains temporary files. If the Enhanced Security Utilities are installed, **/var/tmp** is a multilevel directory.

/var/uucp

This directory contains logging and accounting files for **uucp**.

Files in /var

This section describes the files in the **/var** directories, which contain information that varies from machine to machine.

/var/adm/spellhist

If the SPELL Utilities are installed, this file contains a history of all words that the **spell** command fails to match. Periodically, this file should be reviewed for words that you can add to the dictionary. Clear the **spellhist** file after reviewing it. [See **spell(1)** for information on adding words to the dictionary, cleaning up the **spellhist** file, and other commands that can be used with the SPELL Utilities.]

/var/adm/utmp

This file contains information on the current system state. This information is accessed with the **who** command.

/var/adm/utmpx

This file contains information similar to that in the **/var/adm/utmp** file, along with a record of the remote host.

/var/adm/wtmp

This file contains a history of system logins. The owner and group of this file must be **adm**, and the access permissions must be 664. Each time **login** is run this file is updated. As the system is accessed, this file increases in size. Periodically, this file should be cleared or truncated. The command line **>/var/adm/wtmp** when executed by **root** creates the file with nothing in it. The following command lines limit the size of the **/var/adm/wtmp** file to the last 3600 characters in the file:

```
# tail -3600c /var/adm/wtmp > /var/tmp/wtmp
# mv /var/tmp/wtmp /var/adm/wtmp
#
```

The **/usr/sbin/cron, /usr/sbin/rc0,** or **/usr/sbin/rc2** command can be used to clean up the **wtmp** file. You can add the appropriate command lines to the **/var/spool/cron/crontabs/root** file or add shell command lines to directories such as **/etc/rc2.d, /etc/rc3.d,** and so on.

/var/adm/wtmpx

This file contains information similar to that in the **`/var/adm/wtmp`** file, along with a record of the remote host.

/var/adm/loginlog

After five unsuccessful login attempts, all the attempts are logged in this file. This file contains one record for each failed attempt.

/var/adm/sulog

This file contains a history of substitute user (**`su`**) command usage. As a security measure, this file should not be readable by `others`. The **`/var/adm/sulog`** file should be truncated periodically to keep the size of the file within a reasonable limit. The **`/usr/sbin/cron,`** the **`/usr/sbin/rc0,`** or the command can be used to clean up the **`sulog`** file. You can add the appropriate command lines to the **`/var/spool/cron/crontabs/root`** file or add shell command lines to directories such as **`/etc/rc2.d,`** **`/etc/rc3.d,`** and so on. The following command lines limit the size of the log file to the last 100 lines in the file:

```
# tail -100 /var/adm/sulog > /var/tmp/sulog
# mv /var/tmp/sulog /var/adm/sulog
#
```

/var/cron/log

This file contains a history of all actions taken by **`/usr/sbin/cron.`** The **`/var/cron/log`** file should be truncated periodically to keep the size of the file within a reasonable limit. The **`/usr/sbin/cron,`** **`/usr/sbin/rc0,`** or **`/usr/sbin/rc2`** command can be used to clean up the **`/var/cron/log`** file. You can add the appropriate command lines to the **`/var/spool/cron/crontabs/root`** file or add shell command lines in the following directories (as applicable): **`/etc/rc2.d,`** **`/etc/rc3.d,`** and so on.

The following command lines limit the size of the log file to the last 100 lines in the file:

```
# tail -100 /var/cron/log > /var/tmp/log
# mv /var/tmp/log /var/cron/log
#
```

/var/sadm/bkup/logs/bklog

This file contains a process log used when troubleshooting a backup operation.

/var/sadm/bkup/logs/bkrs

This file contains a process log used when troubleshooting a backup or restore operation for which a method was not specified.

/var/sadm/bkup/logs/rslog

This file contains a process log used when troubleshooting a restore operation.

/var/sadm/bkup/toc

This file contains table of contents entries created by a backup method.

9
Administering Privilege

9
Administering Privilege

Introduction

Privilege, in the simplest terms, is the ability to override system restrictions on the actions of users. All operating systems provide for the exercise of special privilege under certain conditions, to perform certain sensitive system operations. Sensitive system operations affect the configuration of the system or its availability to the user community it serves.

Most users cannot, for example, execute commands affecting the hardware or software configuration of the system. Activities such as mounting and checking file systems, adding users, modifying user profiles, adding and removing peripherals, installing application software, password administration, and administration of the user terminal lines, are restricted to certain users.

The restriction of privilege has traditionally been implemented by designating a special user identifier (UID) of `0`; the login name historically associated with this UID is **`root`**.

When an individual logs in as **`root`**, that individual has unrestricted access to every file on the system, and the ability to perform operations that alter system operation. On such systems, commands performing sensitive system operations check to see whether the effective UID of the process requesting the operation is `0`. If it is, the user process can perform any operation.

The **`root`** login possesses, in effect, the one privilege necessary to override all system restrictions on command execution and access: the superuser privilege.

The privilege mechanism in these systems is the check for a UID of `0`. Any child process spawned by a process with a UID of `0` has unlimited system access; that is, the child processes inherit the privileges of the parent. When a user process attempts a sensitive system operation, the system checks the effective UID of the process. If it is `0`, the child process spawned is given unlimited access to the system.

This type of privilege mechanism is a UID-based privilege mechanism and has one privilege: the superuser privilege. The inheritance policy employed is a simple one: each child of a process with an effective UID of `0` is given unlimited system access.

PowerMAX OS replaces this privilege mechanism with a more flexible mechanism that allows the configuration of the system with a privilege policy module that suits the needs of the user community. This mechanism checks the invoking process for the presence of one or more of a discrete set of privileges corresponding to each sensitive system operation.

The privileges inherited by a new process are derived from the calling process's privileges and the privileges set on the file being executed. This type of privilege mechanism is called a file-based privilege mechanism.

The most important advantage of this privilege mechanism over the UID-based privilege mechanism is the ability to apportion system privileges to executing processes with fine granularity. The inheritance mechanism used provides the ability to control the assertion of privilege throughout the execution of a process, so no privileges are inherited by a new process unless explicitly desired.

The superuser privilege is replaced by a list of discrete privileges based on the categorization of sensitive system operations into groups of operations exercising the same kind of privilege. For example, many different commands might need to override discretionary read access restrictions on files to perform their functions. Defining a privilege such as `P_DACREAD`, and designating it as one of the possible privileges allows for a more controlled possession of privileges by processes than the superuser privilege. Since the privilege only affects discretionary read access, a process that possesses only this privilege cannot perform other privileged operations such as writing to a file without discretionary write access. For a complete list of the privileges supported, refer to the **`intro(2)`** man page.

While the privilege mechanism provides the means by which a system can apportion and control process privileges, the privilege policy module provides the rules by which the system grants privileges to processes.

It is important to recognize that the list of system privileges and privileges on files are all part of the basic privilege mechanism provided with the standard package. The system retains the traditional privilege model (see the *"SUM"* section below); this provides recognition of users with a user ID of 0 (i.e. **`root)`** as superuser. Additionally, you may attach specific privileges to executable files. This allows programs to perform privileged operations without having to become root. With the Enhanced Security Utilities installed, you can use the Least Privilege Policy module, which provides greater security through its inheritance mechanism (see the *"LPM"* section below).

Privileges Associated with a File

For every executable file there may be two sets of privileges that pass on privileges when that program is executed via an `exec` system call:

- Fixed privileges are always given to the new program, independent of the calling or parent process's privileges.
- Inheritable privileges are given to the new program only if they exist in the calling process's privilege set. Inheritable privileges are only used by the LPM privilege module, not by the SUM privilege module. The process's privileges are always considered to be inheritable when using the SUM module. (See *"Privilege Policy Modules"* below.)

These sets are disjoint, that is, a privilege can not be defined as both fixed and inheritable for the same file. If an executable file does not require any privileges then both sets are empty.

In order for a shell script to propagate privileges whether they are acquired by way of **`tfadmin(1M)`** or **`filepriv(1M)`**, the script file must begin with a line of the form:

```
#! pathname [arg]
```

where pathname is the path of the interpreter (usually a shell), and arg is an optional argument.

CAUTION

> Privileges associated with a file are removed when the validity information for the file changes (for example, when the file is opened for writing or when the modes of the file change). This removes the privileges; the privileges must be set again in order for the command to run with privilege. This validity checking can be optionally disabled. See the **`initprivs(1M)`** man page for more information.

Privileges Associated with a Process

After a `fork`, the privileges of the parent and child processes are identical. However, when an `exec` system call is performed, the privileges of the new program are determined from those of the program performing the `exec` and from the privileges associated with the executable file.

Each process has two sets of privileges:

- The maximum set contains all the privileges granted to the process either as fixed or inherited privileges.
- The working set contains all the privileges currently being used by the process.

How the privileges for a new process are determined is specific to the privilege policy module installed. Two different policy modules are provided — one in the Standard Package and one in the Enhanced Security Utilities Package — that allow you to specify the way privileges are inherited by processes.

The maximum and working sets of privileges associated with the calling process can be displayed using the **`priv(1)`** command or the **`sh(1)`** builtin **`priv`** command. Privileges can be added or deleted in the maximum and working sets of the calling process using the **`sh(1)`** builtin **`priv`** command.

The **`procpriv(2`**) system call and the **`procprivl(3C)`** library routine can be used to add, retrieve, remove or count privileges associated with the calling process. See the *"Security Considerations"* section in the "Directory and File Management" chapter of the PowerMAX Programming Guide for further information.

Privileges Associated with Users

Privileged users need to perform sensitive tasks, but because privileges are associated with processes and executable files, not user IDs (except for the special case of UID=0 when using the SUM privilege policy module), it is not possible to grant privileges to users directly. The Trusted Facility Management (TFM) tools provide an interface between users and privileges. The TFM tools maintain a database of users and the commands they may execute with privilege. The **`tfadmin(1M)`** command invokes the requested command, regulating the privileges based on the TFM database information.

The **`tfadmin`** command is invoked with the desired command line as its arguments. The fixed privilege set of the **`tfadmin`** command file contains all privileges, so the **exec** system call of **`tfadmin`** turns on all privileges in the resulting process. When **`tfadmin`** is invoked, it finds out the real identity (real UID) of the invoking user. It then uses that identity to find the user's entry in the TFM database.

The TFM database contains three types of information:

- the list of privileged commands that belong to each user
- the list of roles to which each user is assigned
- the list of privileged commands that belong to each role

When **`tfadmin`** finds the user's entry, it looks for the requested command in the list of specific commands, and if it does not find it, in the list of roles. Once the appropriate entry is found, **`tfadmin`** changes his own process's maximum and working sets of privileges to match the entry's privileges (using **`procpriv(2)`**), and then executes the command. If the executable file has any fixed file privileges associated with it (via the **`filepriv(1M)`** command), they will be added to the privileges obtained from the TFM database by the **exec** system call. All privileges obtained from the TFM database and the executable file's fixed privileges are propagated across the chain of execution of any child processes.

Unlike file privileges, the TFM database privileges are not invalidated if the executable file is modified. The TFM tools eliminate the need to place fixed file privileges with **`filepriv(1M)`** on most commands (other than the **`tfadmin`** command itself).

An administrator has much greater control over which users execute with privilege using the TFM tools than what is available using file privileges. Anyone that has execute permissions on the file will be granted the fixed set of file privileges for that file. The system administrator retains the most control with the TFM tools by specifying exactly which commands can be executed with privilege by each user. However, this can become a burden for the administrator to have to add new commands to the TFM database every time users need to execute additional commands with privilege.

Privileges can be assigned to the user's shell instead of assigning privileges to individual commands. Because privileges are inherited by child processes, any privileges given to the user's shell by **`tfadmin`** will be inherited by every command the user executes under that shell.

In order for a shell script to propagate privileges, whether they are acquired by way of **`tfadmin`** or **`filepriv`**, the script file must begin with a line of the form:

```
#! pathname [arg]
```

where pathname is the path of the interpreter (usually a shell), and arg is an optional argument.

Assigning privileges to users' shells is not recommended on systems running with the Enhanced Security Utilities installed because it does not follow the policy of least privilege. Every command the user invokes under the privileged shell will execute with privileges.

For further information about the TFM tools and privileged shells, see Chapter 10 “Trusted Facility Management”.

Privilege Policy Modules

The privilege mechanism allows you to configure the operating system with a privilege policy module that meets the requirements of your user community.

The system is delivered with a Super User Module (SUM) providing the same functionality provided by the superuser privilege in previous releases. The Enhanced Security Utilities are delivered with a Least Privilege Module (LPM), providing a more restrictive privilege policy.

The SUM module is used unless the Enhanced Security Utilities are installed and you specifically include the LPM module.

SUM

This privilege mechanism functions exactly the same as the superuser, a UID-based mechanism common to earlier releases of the OS. It also provides additional flexibility by allowing the association of fixed privileges with executable files.

This mechanism works by using:

- a list of system privileges
- a working and maximum set of privileges for each process on the system
- a fixed set of privileges for executable files

If the bootable operating system is configured with the SUM module [specified in the **`/etc/conf/sdevice.d/sum`** file; see *“Configuring the Kernel”* in the “Managing System Performance” in volume 2 of this book], when a user executes a command, the privilege mechanism does the following:

- the union of the maximum privileges of the calling process and the fixed privileges of the executable are placed in the maximum and working sets of the new process
- if the effective UID of a calling process is equal to the tunable parameter **`privid`**, or a file being executed has its set-user-ID-on-execution bit set and is owned by **`privid`**, all privileges are placed in the maximum and working sets of the new process

If the effective UID of a process changes during execution, then the following occurs:

- if none of the real, new effective, or saved UIDs of a process are equal to **`privid`**, then set all privileges in the current maximum and working sets of that process to the saved set of fixed file privileges
- if the new effective UID of a process is equal to **`privid`**, then set the current working privileges to the current maximum set
- if the new effective UID of a process is not equal to **`privid`** (but either the real and/or saved UIDs are equal to **`privid`**), then set all privileges in the current working set to the saved ser of fixed file privileges

With **`privid`** set equal to `0`, this behavior preserves the omnipotence of a process with effective `uid 0`. The Standard Package is delivered with **`privid`** equal to `0`. Except in rare cases, this value should not be changed.

In the SUM module, fixed privileges on a file are passed to a new process. This behavior provides a way for non-`uid 0` processes to execute commands with privilege. The union of the maximum privileges of the calling process and fixed privileges of the executable are propagated to a new process regardless of UID. The ability to specify fixed privileges when using the SUM module is provided:

- to allow certain system processes to run with fixed privileges needed to function properly.
- to allow logins other than **`root`** to execute privileged commands if desired.
- to allow applications programs to be written independent of any particular privilege policy module; writing privileged applications is covered in *"Programming with UNIX System Calls"*.

A good example of the first two cases is the **`ps`** command. Various system and other processes (owned by unprivileged UIDs) use **`ps`** in a privileged manner to get the status of processes that are not owned by the same UID. These processes do not run set-UID to `uid 0` and the set-UID-on-execution bit of **`ps`** is not set. Therefore, to be executed by non-`uid 0` processes in a way that exercises privilege, the SUM module propagates the union of the maximum privileges of the parent process and the fixed privileges of **`ps`** to the new process.

Figure 9-1 portrays a conceptual explanation of fixed privileges when the SUM module is configured.

LPM

The LPM policy module (included in the Enhanced Security Utilities) is designed to meet the requirements for a system running in compliance with government B2-level security requirements. These guidelines specify that processes execute with only the amount of privilege necessary to perform their given function, and no more.

To accomplish this, the LPM module recognizes inheritable privileges on executable files as well as fixed privileges. Whenever a command is executed under LPM, the working and maximum sets of the new process are set to contain:

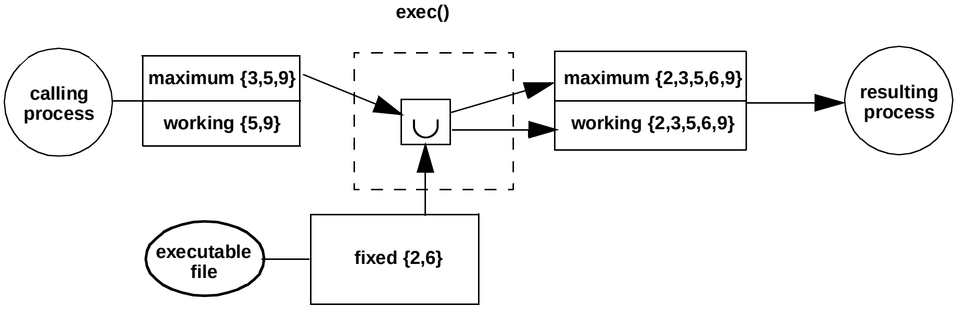


Figure 9-1. Fixed Privileges (SUM Module)

- those privileges that are common to both the maximum set of the calling process and the inheritable set of the executable file
- and any fixed privileges on the file

LPM severs the connection between UID and privilege present in prior releases. To pass a privilege to a new process (execute a command with privilege), either the current process must possess the privilege(s) necessary in its maximum set, or the executable must possess the necessary privilege(s) in its fixed set. No process can execute a command with a privilege that the command was not designed to enforce.

Figure 9-2 portrays a conceptual explanation of fixed and inheritable privileges when the LPM Module is configured.

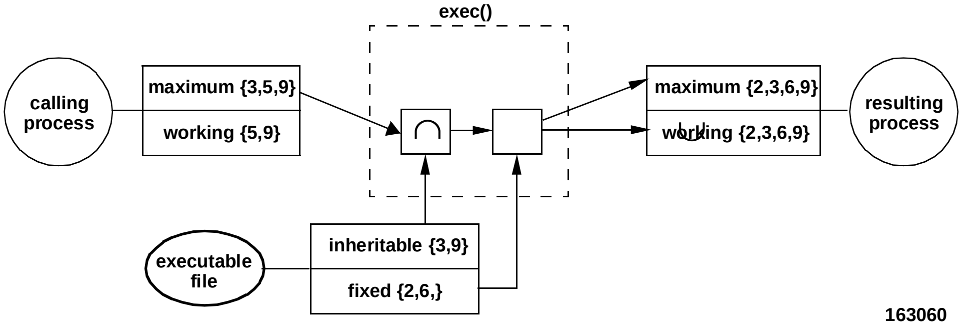


Figure 9-2. Fixed and Inheritable Privileges (LPM Module)

Another characteristic of the LPM module is, unlike the SUM module wherein **`root`** processes always run with privilege, all user processes start with empty privilege sets. To allow a set of authorized users to perform administrative functions on the system, authorized users spawn privileged child processes from unprivileged parents through the Trusted Facility Management mechanism.

Configuring To Use SUM or LPM

See *"Configuring the Kernel"* in the "Managing System Performance" chapter in volume 2 for information on how to include or exclude SUM and LPM for multi-user operation. You do this by changing settings in the **/etc/conf.d/sdevice.d/lpm** and **/etc/conf.d/sdevice.d/sum** files, and then rebuilding the kernel. You must choose either SUM or LPM; you cannot include or exclude both.

Remember, LPM is available only with the Enhanced Security Utilities installed.

System Start-Up and the Kernel Privilege Table

The kernel maintains a table of file privileges in memory. It is initialized at system startup by the **initprivs(1M)** command using the file privilege entries in the Privilege Data File (PDF), **/etc/security/tcb/privs**.

NOTE

> If the PDF is missing, an error results and the system is halted. The system should then be rebooted from tape.

Entries in the PDF are added, deleted, or modified using the **filepriv(1M)** command. When the **filepriv** command adds a file to the PDF, it records checksum, size and last updated time information about the file in addition to the file privileges. **initprivs** compares this validity information to the current values for the file. By default, if these do not match, the file will not be granted privileges by **initprivs** and the entry will not be passed to the kernel to add to the kernel privilege table. This validity checking can be disabled by resetting flags contained in the file **/etc/default/privcmds**. On systems where the SUM module is configured (the Enhanced Security Utilities are not installed), disabling validity checking in **initprivs** also disables the validity checking performed by the kernel when a file is executed with **exec(2)**. See **initprivs(1M)** for further information.

File privilege entries in the kernel privilege table can be set, retrieved or counted using the **filepriv(2)** system call. The **filepriv(2)** system call does not modify the PDF entry for the file. Privileges that are changed with **filepriv(2)** are valid only until the next reboot, at which time the changes are lost and the privileges are as defined in the PDF.

The **filepriv(1M)** command updates the kernel privilege table (using the **filepriv(2)** system call) each time additions, deletions or changes are made to entries in the PDF.

You may want to change some of these privileges in accordance with your local security policy and needs. If you add trusted software to your system, you will want to assign privileges to that software. (See Chapter 12, *"Installing Software on an Enhanced Security System"* in this book.)

The following sections discuss how to display, set, and change file privilege information. Privilege information should always be changed with great care, since the privileges assigned to executables directly affect the security of the system.

File Privileges and the filepriv Command

The **`filepriv`** command is used to:

- set up the PDF used at system start-up
- display privilege information about a file
- install new programs requiring privileges
- change privileges on existing files
- remove privileges on existing files

NOTE

> If you modify in any way a file with privileges associated with it, those privileges are removed and you must re-set the privileges with the **`filepriv`** command. This precludes a malicious user replacing a privileged program with another program that can then execute with those same privileges. Otherwise, a malicious user might somehow replace a command possessing read privileges with their own program. Executing that program would allow reading any file on the system. Since the privileges disappear when a file is modified, even if the user could put their program in place, it would not run with privilege.
>
> This validity checking can be disabled by modifying option flags contained in **`/etc/default/privcmds`**. See **`initprivs(1M)`** for details.
>
> You must have the `P_SETSPRIV` privilege when setting or deleting file privileges.

The following sub-sections explain how to perform these tasks through examples.

Displaying Privilege Information

To display the privilege information for a file, use the **`filepriv`** command with the desired file names, but without any options.

Before You Begin

You must have read access to the file for which you wish to display. The **`filepriv`** command does not have any privileges.

Procedure

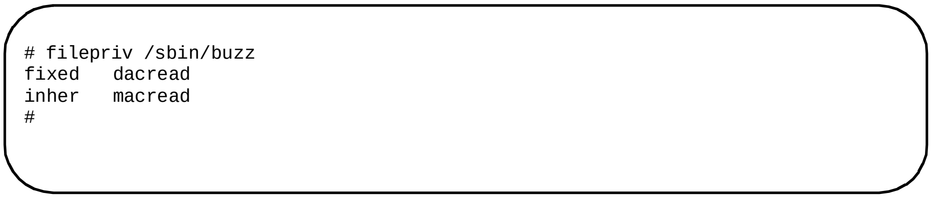
To display privilege information for a file, perform the following step:

1. Enter:

   `filepriv` *file_name*

Example: Displaying File Privileges

Screen 9-1 shows how to display the privileges associated with the executable **`/sbin /buzz.`**



```
# filepriv /sbin/buzz
fixed   dacread
inher   macread
#
```

Screen 9-1. Displaying Privileges of an Executable File

This shows that the **`/sbin/buzz`** executable has the fixed privilege `P_DACREAD` and the inheritable privilege `P_MACREAD.`

NOTE

Since inheritable privileges have no effect on a SUM system, they will not be displayed.

Installing a New Program Requiring Privileges

You can also use **`filepriv`** to set privileges on files. Because **`filepriv`** removes all privileges currently associated with the file before the privileges specified on the command line are applied, you must specify all privileges for a file during one invocation of **`filepriv`**. Finally, you must specify the absolute path of a file when setting privileges.

Procedure

To set the fixed and inheritable privileges for a file, perform the following step:

1. **filepriv** with the **-f** and **-i** options; the **-f** option is used to define the fixed privileges. The **-i** option is used to define the inheritable privileges. You can not specify the same privilege in both the **-f** and **-i** options, because a privilege must be either fixed or inherited. If you are specifying only one set of privileges, fixed or inherited, you do not need to specify both options. Enter:

   filepriv **-f** *priv*,*priv*... **-i** *priv*,*priv*... *file_name*

   The *file_name* must be the full path name of the file.

Example: Setting Privileges on a File

Suppose you want to install a new program, **/sbin/muzz,** that always requires privilege to override Discretionary Access Control (DAC) read restrictions and also inherit the privilege to override DAC write restrictions. Therefore, you want to make dacread a fixed privilege and dacwrite an inheritable privilege.

Using the example of the **muzz** program, to set the required privileges for the program enter:

```
filepriv -f dacread -i dacwrite /sbin/muzz
```

NOTE

Although SUM does not support inheritable privileges, you can use **filepriv** to set inheritable privileges on a file while running with SUM. However, in these cases, filepriv will print the following warning: UX:filepriv:WARNING: “inher” set not supported by this privilege mechanism. Since **filepriv** cannot be used to check inheritable privileges on a system using SUM, you will have to check the PDF directly if you wish to verify that the inheritable privileges were set correctly.

Changing Privileges on an Existing File

Changing the privileges on an existing file is much like installing a new program requiring privileges. As in the example above, use **filepriv** with the **-f** option to define the fixed privileges and the **-i** option to define the inheritable privileges. Because **filepriv** removes all privileges previously associated with a file when it is invoked with either the **-f** or the **-i** option, the previous privileges are removed and the specified privileges are installed.

The correct sequence of steps to change the privileges on an existing file is to display the existing privileges, change the privileges by installing the new privileges, and then display the existing privileges again to make sure that the correct privileges are in place.

Removing Privileges on Existing Files

To remove all privileges associated with an executable, use **`filepriv`** with the **`-d`** option. As with setting privileges, you must specify an absolute path when deleting privileges.

Procedure

To remove all privilege from a file, enter the following step:

1. Enter:

   `filepriv` **`-d`** *file_name*

   The *file_name* specified must be the full path name.

Example: Removing Privileges from a File

The operating system shows how to remove all privileges for **`/sbin/nuzz.`** Of course, once the privileges are removed **`/sbin/nuzz`** will no longer work as expected because it no longer has the privileges required to complete its task.

```
# filepriv /sbin/nuzz
fixed   dacread
inher   dacwrite
# filepriv -d /sbin/nuzz
# filepriv /sbin/nuzz
#
```

See **`filepriv(1M)`** in the online manual page for more details.

Backup and Restore of Files With Privileges

The Trusted Import/Export backup and restore tool **`tcpio(1)`** is used to create an archive from a specified list of files. For executable files with file privileges, the file privileges will be saved and may be displayed using “`tcpio -t`” or “`tcpio -v`”. However, tcpio does not restore file privileges. A privileged user must assign the appropriate privileges to the restored file using the **`filepriv(1M)`** command.

The **`fsdump(1M)`**, **`fsrestore(1M),`** and **`cpio(1)`** commands do not save file privileges.

10
Trusted Facility Management

10
Trusted Facility Management

Executing Processes with Privilege: TFM

Administrators and privileged users need to perform sensitive tasks, but because privileges are associated with processes, not user IDs (except for the special case of `UID=0` if you are using the SUM privilege policy module), it is not possible to grant privileges to users to perform these sensitive tasks. The Trusted Facility Management tools (TFM) provide the means to maintain a database of users and the commands they may execute with privilege. The **`tfadmin(1)`** command invokes the requested command, regulating the privileges based on the TFM database information.

Each command entry in the database includes an alias for the command, the path to the executable file, and the privileges to be granted. An administrator or user who is in the TFM database must execute the **`tfadmin`** command to run these commands with the granted privileges. Users not in the TFM database cannot execute commands with privilege, and will get an error if they attempt to use **`tfadmin.`**

TFM also gives you the ability to associate a set of privileged commands with a role. The privileged commands associated with each role are then associated with people assigned to that role.

When first setting up a new system, an administrator should carefully consider who should be assigned which roles and what new roles, if any, should be defined. See the *"TFM and Administrative Roles"* section for a discussion of roles.

Also, see Chapter 12, "Installing Software on an Enhanced Security System" of this book for information on setting up roles.

You can administer and access the TFM database with the following three commands:

- The **`adminuser`** command associates a user with individual privileged commands or associates a user with a role.
- The **`adminrole`** command can display, add, change, or delete roles in the TFM, and it can associate commands and privileges with each of the roles.
- Administrators and authorized users must use **`tfadmin`** to execute commands with privilege.

The **`tfadmin`** command (shown in Figure 10-1) is invoked with the desired command line as its arguments. The fixed privilege set of the **`tfadmin`** command file contains all privileges, so the **exec** system call of **`tfadmin`** turns on all privileges in the resulting process. When **`tfadmin`** is invoked, it finds out the real identity (real UID) of the invoking user. It then uses that identity to find the user's entry in the TFM database.

The TFM database contains three types of information:

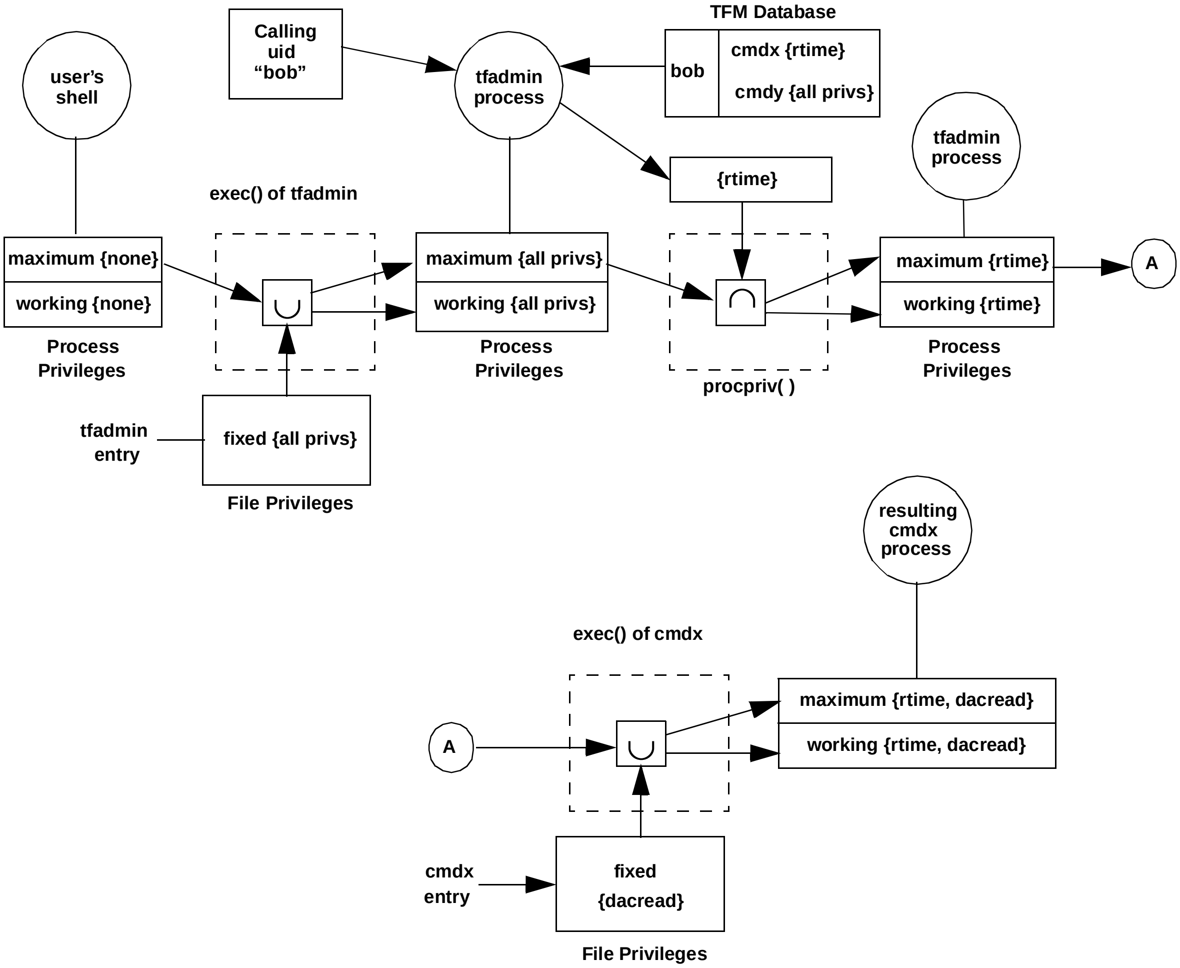


Figure 10-1. tfadmin Execution of cmdx with Privilege (SUM Module Installed)

- the list of privileged commands that belong to each user
- the list of roles to which each user is assigned
- the list of privileged commands that belong to each role

When **`tfadmin`** finds the user's entry, it looks for the requested command in the list of specific commands, and if it does not find it, it then looks in the list of roles. Once the appropriate entry is found, **`tfadmin`** changes his own process's maximum and working sets of privileges to match the entry's privileges (using **`procpriv(2)`**), and then executes the command. If the executable file has any fixed file privileges associated with it (via the **`filepriv(1M)`** command), they will be added to the privileges obtained from the TFM database by the **exec** system call. All privileges obtained from the TFM database and the executable file's fixed privileges are propagated across the chain of execution of any child processes.

In order for a shell script to propagate privileges whether they are acquired by way of **`tfadmin`** or **`filepriv`**, the script file must begin with a line of the form:

```
#! pathname [arg]
```

where pathname is the path of the interpreter (usually a shell), and arg is an optional argument

Unlike file privileges, the TFM database privileges are not invalidated if the executable file is modified. The TFM tools eliminate the need to place fixed file privileges with **`filepriv(1M)`** on most commands (other than the **`tfadmin`** command itself).

An administrator has much greater control over which users execute with privilege using the TFM tools than what is available using file privileges. Anyone that has execute permissions on the file will be granted the fixed set of file privileges for that file. The system administrator retains the most control with the TFM tools by specifying exactly which commands can be executed with privilege by each user.

Adding Commands for a User

Use **`adminuser`** with the **`-a`** option to assign commands to users in the TFM database.

Procedure

To assign a command to a user, perform the following steps:

1. Enter

   `adminuser` **`-n -a`** *entry1*,*entry2*... *user_name*

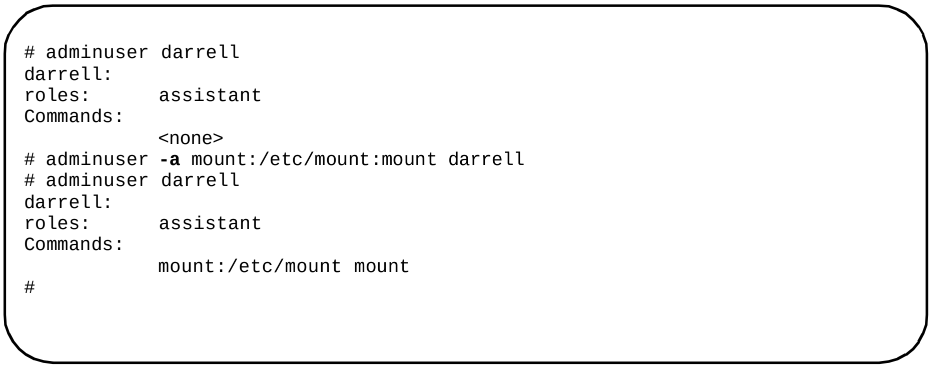
   Each *entry* is a command and privilege entry as specified on the **`adminuser(1M)`** online manual page. When you assign a command to a user, make sure the user belongs to a group from which the relevant command is accessible. The **`-n`** option is only necessary if the user is not already defined in the TFM database.

Example

Screen 10-1 shows how to allow `darrell` to execute the **`mount`** command with privileges.

The initial **`adminuser`** displays the current entries for `darrell`, in this case the role `assistant`. The next **`adminuser`** adds the command **`mount`** for `darrell`. The first **`mount`** is the command alias `darrell` would use with the **`tfadmin`** command.

The second portion, **`/etc/mount,`** specifies the full path of the command. Any privileges following the path, in this case **`mount`**, are granted to the user when executing the command. If `darrell` did not already exist in the TFM database, you would also need to specify the **`-n`** option.



```
# adminuser darrell
darrell:
roles:      assistant
Commands:
            <none>
# adminuser -a mount:/etc/mount:mount darrell
# adminuser darrell
darrell:
roles:      assistant
Commands:
            mount:/etc/mount mount
#
```

Screen 10-1. Adding a Command for a User

The final **`adminuser`** displays all the TFM entries for `darrell`, at this point the role `assistant` and the command **`mount.`**

Removing Commands from the TFM Database

Use **`adminuser`** with the **`-r`** option to remove commands and privileges from a user.

Procedure

To remove commands and privileges from a user, perform the following steps:

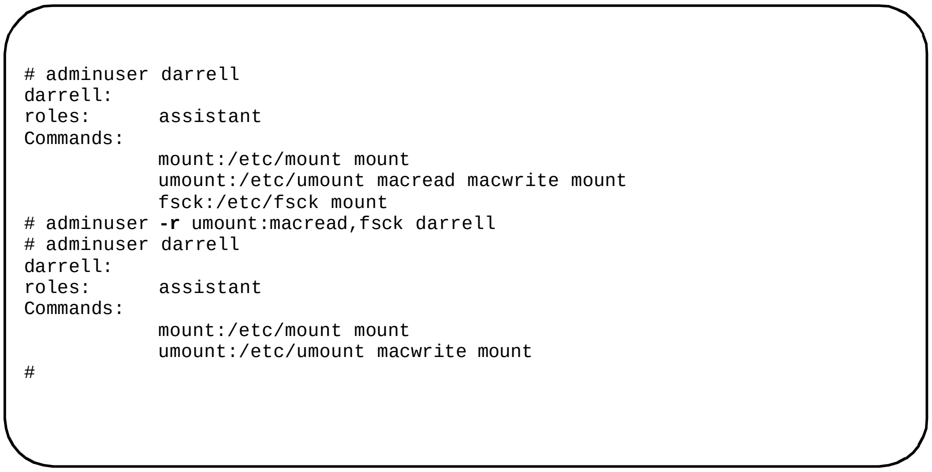
1. Enter

   `adminuser` **`-r`** *command_name1*`:`*priv*`,`*priv role_name*

   The *command_name1* is the name of a command defined for that role.

Example

Screen 10-2 shows how to remove a privilege from a command and remove an entire command entry for a user.



```
# adminuser darrell
darrell:
roles:      assistant
Commands:
            mount:/etc/mount mount
            umount:/etc/umount macread macwrite mount
            fsck:/etc/fsck mount
# adminuser -r umount:macread,fsck darrell
# adminuser darrell
darrell:
roles:      assistant
Commands:
            mount:/etc/mount mount
            umount:/etc/umount macwrite mount
#
```

Screen 10-2. Removing Privileges and Commands for a User

Assuming that entries exist in the TFM database for **`umount`** and **`fsck`** for `darrell`, the **`macread`** privilege is removed from the **`umount`** command, while the remaining privileges are left intact, and the entire entry for **`fsck`** is removed. Note that the privileges used here are examples only; they may not be privileges you want to use with these commands.

Removing a User from the TFM Database

Use **`adminuser`** with the **`-d`** option to remove a user from the TFM database.

Procedure

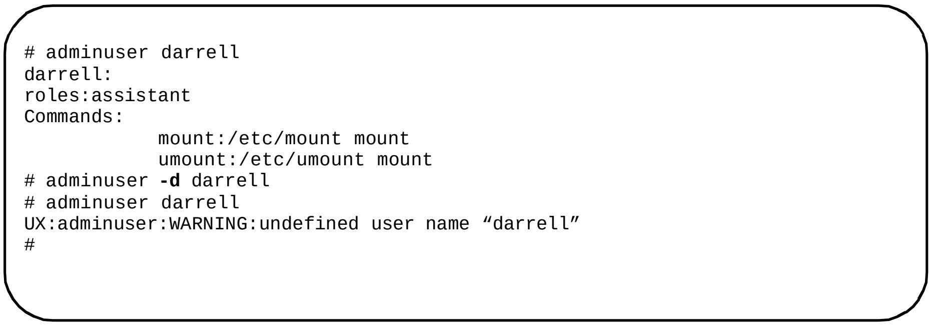
To remove a user from the TFM database, perform the following steps:

1. Enter

   `adminuser` **`-d`** *user_name*

Example

Screen 10-3 shows how to remove the user `darrell` from the TFM database.



```
# adminuser darrell
darrell:
roles:assistant
Commands:
            mount:/etc/mount mount
            umount:/etc/umount mount
# adminuser -d darrell
# adminuser darrell
UX:adminuser:WARNING:undefined user name “darrell”
#
```

Screen 10-3. Removing a User from the TFM Database

This command removes the user `darrell` from the TFM database.

Privileged Shells

The system administrator retains the most control with the TFM tools by specifying exactly which commands can be executed with privilege by each user. However, this can become a burden for the administrator to have to add new commands to the TFM database every time users need to execute additional commands with privilege.

Privileges can be assigned to the user's shell instead of assigning privileges to individual commands. Because privileges are inherited by child processes, any privileges given to the user's shell by **`tfadmin`** will be inherited by every command the user executes under that shell.

To add user bob to the TFM database and setup a privileged **`sh(1)`** shell, the administrator enters:

```
adminuser -n -a sh:/usr/bin/sh:dacread:rtime bob
```

The “-n” option is required if the user is not already in the TFM database. When user bob wants to execute his shell with privileges, he enters:

```
tfadmin sh
```

When user bob logs onto the system, he would run without any privileges until he enters the **`tfadmin`** command to startup the privileged shell. When he exits the privileged shell, he would return to running his unprivileged shell.

If the user needs to always run with the privileged shell, he can add the `“tfadmin sh”` statement to the end of his **`.profile`**. He would not be required to enter the **`tfadmin`** command every time he logs onto the system.

Another approach to assigning privileges is the use of roles. Roles are very useful when you have several users that you want to assign the same set of privileged commands. The following example sets up a role named OS_USERS. Privileges will be set on each of the different shells the users in the OS_USERS role might want to use as their shell. The following commands add the role OS_USERS and assign privileges to three shells:

```
adminrole -n -a sh:/usr/bin/sh:dacread:rtime OS_USERS
adminrole -a ksh:/usr/bin/ksh:dacread:rtime OS_USERS
adminrole -a csh:/usr/bin/csh:dacread:rtime OS_USERS
```

The “-n” option is required if the role is not already in the TFM database. For each user to be added to the OS_USERS role, the administrator enters:

```
adminuser -n -o OS_USERS joe
adminuser -n -o OS_USERS sue
```

When user joe wants to execute his shell with privileges, he enters:

```
tfadmin OS_USERS: ksh
```

The “OS_USERS:” does not need be specified on the **tfadmin** command unless joe is a member of more than one role. When user joe logs onto the system, he would run without any privileges until he enters the **tfadmin** command to startup the privileged shell. When he exits the privileged shell, he would return to running his unprivileged shell.

If the user needs to always run with the privileged shell, he can add the `“tfadmin OS_USERS: ksh”` statement to the end of his **.profile**. He would not be required to enter the **tfadmin** command every time he logs onto the system.

Assigning privileges to users' shells is not recommended on systems running with the Enhanced Security Utilities installed because it does not follow the policy of least privilege. Every command the user invokes under the privileged shell will execute with privileges.

TFM and Administrative Roles

To enhance security, the power of superuser to override system restrictions has been divided into separate process privileges, also, the concept of an all-purpose administrator can be replaced with the concept of administrative roles. Each role is responsible for different areas of system or security administration.

By separating the responsibilities and assigning them to different people, you increase the security of your site. The separation of responsibilities and abilities to perform sensitive system tasks into separate roles decreases the risk that an administrator will knowingly or unknowingly perform a task that violates the security policy.

Initially, the Standard Package has four administrative roles defined. You may change or add to these if desired, keeping in mind the effect of the changes on the security of the system. To maintain the security of the system, no person should be assigned more than one of these roles.

No one is associated with these roles when installed. You must carefully decide who to assign to each role. You must then follow the directions in Chapter 17, “Administering Mandatory Access Control and Multilevel Directories” in this book to assign initial roles and groups during system installation if you are running with the Enhanced Security Package installed.

You can use the **adminuser** command (see below) to assign administrators to roles. You may also define new roles using the **adminrole** command. (See below.)

The four initial roles are:

- Site Security Officer (SSO)

  Any person assigned this role makes virtually all decisions for the allocation of resources among security levels, and the security level configuration of the system. The duties of the SSO include the following:

  - define the security level constraints on system resources and use
  - control the mechanisms that limit covert channel activity
  - define the users and discretionary access control groups
  - assign clearances for all users

  The SSO does not have control of the security audit functions; these activities belong to the AUD role. (See below.)

  The SSO should be trained in local security policies and practices, and should have a knowledge of generally accepted security practices in order to make security-related configuration decisions.

- Auditor (AUD)

  The auditor is responsible for control of the security audit system (this excludes UNIX System accounting). The duties include the following:

  - set the parameters that control the generation of raw audit information
  - modify or delete primary (as opposed to copies of) information generated by the security audit system
  - control the audit archive

  The auditor and SSO form a “check and balance” system, because the SSO sets and enforces the security policy and the AUD controls the auditing information to show that the policy is enforced and has not been circumvented. To maintain this accountability, we recommend you assign the roles of the SSO and the AUD to different people.

- Operator (OP)

  The operator performs regular, non-security critical activities, such as the following:

  - start and stop the system, including duties such as check disk packs for consistency
  - format new media
  - maintain network tables and interfaces
  - generate raw UNIX System accounting data
  - set terminal parameters

  - enable or disable a login, but not change the password, clearance, or any other security related login parameters

  Although these and similar functions affect system security in the broadest sense, they do not affect the TCB, because the OP does not make decisions affecting security levels. For example, the operator may be able to configure a new terminal port but not its allowed security levels.

- Security Operator (SOP)

  The Security Operator role performs routine, daily activities similar to those performed by the operator; however, some of these activities (such as those involving security level definition) are security-sensitive and are restricted to the SOP. The SOP can be thought of as an operator with special capabilities.

  The SOP role includes the following capabilities:

  - perform all duties of the OP
  - specify the level for imported data from single-level media
  - make routine backups and restores
  - mount and dismount mountable media

Table 10-1 shows a portion of the commands associated with the initial four roles and the roles which are allowed to use them.

Table 10-1. Sample Commands for Initial Roles

Command	OP	SOP	SSO	AUD
auditlog	n	n	n	*
chroot	n	n	y	n
cp	n	n	y	n
date	n	n	y	n
fsck	y	y	y	n
ln	n	n	y	n
umount	n	y	y	n
mv	n	n	y	n
* Only if the Audit Package is installed.				

Note that these four initial roles separate the administration of the system into logical groups of tasks, and that these groups of tasks prevent any one person from having sufficient power to circumvent the system restrictions and the security policy.

This separation of responsibilities is intentional; it enhances security and should not be changed. That is, it is possible for one person to have access to more than one role, for example, it is acceptable for the SSO to have access to OP or SOP duties, but not vice versa. This practice, however, is not recommended. The separation of administrative roles and the assignment of these roles to separate people provides a system of checks and balances. This also provides for greater accountability of system activities.

Note that each administrative role is automatically assigned the privileged commands needed to perform the functions of that role. For example, the Security Operator has access to the commands needed to run a trusted backup and restore, the auditor has access to the commands needed to administer the Auditing Utilities, and so on. The administrator who installs the Enhanced Security Utilities does not need to assign new commands to these roles in order for the administrators to perform their administrative tasks. For a listing of the commands in the TFM database assigned to each role, use the **`adminrole`** command as described below.

Displaying Commands and Privileges for a Role

Use the **`adminrole`** command to see a complete list of commands and privileges for each role.

Procedure

To display commands and privileges for a role, perform the following step:

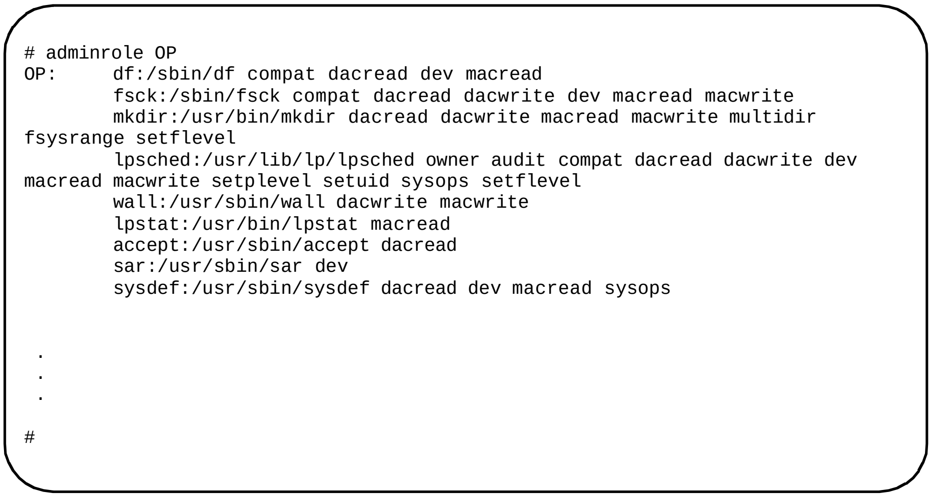
1. Enter:

   `adminrole` *role_name*

Example: Displaying Commands and Privileges for a Role

For example, to see all the commands for the role OP enter

```
adminrole OP
```



```
# adminrole OP
OP:     df:/sbin/df compat dacread dev macread
        fsck:/sbin/fsck compat dacread dacwrite dev macread macwrite
        mkdir:/usr/bin/mkdir dacread dacwrite macread macwrite multidir
fsysrange setflevel
        lpsched:/usr/lib/lp/lpsched owner audit compat dacread dacwrite dev
macread macwrite setplevel setuid sysops setflevel
        wall:/usr/sbin/wall dacwrite macwrite
        lpstat:/usr/bin/lpstat macread
        accept:/usr/sbin/accept dacread
        sar:/usr/sbin/sar dev
        sysdef:/usr/sbin/sysdef dacread dev macread sysops


 .
 .
 .

#
```

Screen 10-4. Displaying Commands and Privileges for a Role

Adding Commands to a New Role

Use **adminrole** with the **-n** and **-a** options to add commands and privileges to a new role.

Procedure

To add commands and privileges to a new role, perform the following steps:

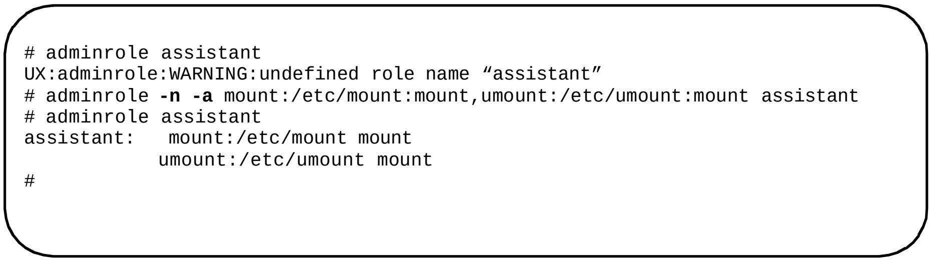
1. Enter

   adminrole **-n -a** *entry1*,*entry2 role_name*

   Each *entry* is a command and privilege entry as specified on the **adminrole(1M)** online manual page.

Example

Screen 10-5 shows how to create a new role, assistant, and associate the **mount** and **umount** commands with the appropriate privilege with the role assistant.



```
# adminrole assistant
UX:adminrole:WARNING:undefined role name “assistant”
# adminrole -n -a mount:/etc/mount:mount,umount:/etc/umount:mount assistant
# adminrole assistant
assistant:   mount:/etc/mount mount
            umount:/etc/umount mount
#
```

Screen 10-5. Adding Commands and Privileges for a New Role

The first **adminrole** displays any TFM database entries for the role assistant. In this case the role does not exist. The next **adminrole** creates the new role assistant; the **-n** option shows this is a definition of a new role. If role assistant already exists in the TFM database omit the **-n** option. As with the **adminuser** command, the initial **mount** and **umount** are the command aliases used with the **tfadmin** command, **/etc/mount** and **/etc/umount** are the paths for the executables, and the final **mount** for each definition specifies the **mount** privilege should be granted when running these commands.

Adding Commands to an Existing Role

Use **adminrole** with the **-a** option to add commands and privileges to a new or existing role.

Procedure

To add commands and privileges to an existing role, perform the following steps:

1. Enter

   `adminrole` **`-a`** *entry1*`,`*entry2 role_name*

   Each *entry* is a command and privilege entry as specified on the **`adminrole(1M)`** online manual page.

Example

Screen 10-6 shows how to associate the **`date`** command with the appropriate privilege with the role `assistant,` given an initial definition for **`mount`** and **`umount.`**

```
# adminrole assistant
assistant:   mount:/etc/mount mount
            umount:/etc/umount mount
#
# adminrole -a date:/usr/bin/date:dacwrite:macwrite:sysops assistant
# adminrole assistant
assistant:   mount:/etc/mount mount
            umount:/etc/umount mount
            date:/usr/bin/date dacwrite macwrite sysops
```

Screen 10-6. Adding Commands and Privileges for an Existing Role

The first **`adminrole`** displays any TFM database entries for the role `assistant`. In this case entries exist for the **`mount`** and **`umount`** commands. The next **`adminrole`** adds the **`date`** command. As with the **`adminuser`** command, the initial **`mount`**, **`date`**, and **`umount`** are the command aliases used with the **`tfadmin`** command, **`/etc/mount,`** **`/usr/bin/date`**, and **`/etc/umount`** are the paths for the executables, followed by the list of privileges that should be granted when running these commands: **`mount`** for the **`mount`** and **`umount`** commands, and **`dacwrite`**, **`macwrite,`** and **`sysops`** for the **`date`** command.

Command/Role Assignments: A Sample Scenario

The following table shows some example administrative roles and the commands that you might want to define on your system. The examples are, in fact, what is defined on your system with the standard package, and the additional commands setup by the Audit and Enhanced Security packages. The command/role assignments must be set up such that users assigned to a particular role can execute commands associated with the role with necessary privileges. These privileged administrative actions are indicated on the online manual pages for the commands. If you are using TFM tools to maintain system security, then whenever one of these commands is referenced in this book, and is being used in a way that requires privilege (as indicated on its online manual page), you must be logged in as a login that has been assigned to a role that allows the execution of the command with appropriate privileges.

To execute a command with privilege, you must precede the command line you want to execute with **`tfadmin`**. See *"Executing Commands with Privilege"* later in this chapter, and **`tfadmin(1M)`** online manual page for more information.

You can view the privileges apportioned to each command for each role using the **`adminrole`** command.

Removing Commands and Privileges for a Role

Use **`adminrole`** with the **`-r`** option to remove commands and privileges from a role.

Table 10-2. Command/Role Assignments

AUD (Auditor)					
Standard:					
none					
Audit:					
auditlog	auditoff	auditrpt	auditset	cat	find
auditmap	auditon				
OP (Operator)					
Standard:					
accept	df	fsck	lpsched	lpstat	mkdir
sar	sysdef	wall			
Enhanced Security:					
admalloc	devstat	rtcpio			
SOP (Security Operator)					
Standard:					
accept	at	cancel	cpio	crontab	df
disable	du	enable	find	fsck	fuser
init	kill	lpadmin	lpfilter	lpforms	lpmove
lpsched	lpshut	lpstat	lpusers	ls	mkdir
mknod	mount	pmadm	ps	reject	sacadm
sar	shutdown	sttydefs	sysdef	umount	volcopy
wall					
Enhanced Security:					
admalloc	devstat	rtcpio	tcpio		

Table 10-2. Command/Role Assignments (Cont.)

SSO (Site Security Officer)					
Standard:					
accept	addgrpmem	at	cancel	cat	chgrp
chmod	chown	chroot	cp	cpio	cron
crontab	date	defadm	df	disable	dispadmin
du	enable	find	finduser	fsck	fuser
groupadd	groupdel	groupmod	init	ipcrm	ipcs
kill	ln	logins	lp	lpadmin	lpfilter
lpforms	lpmove	lpsched	lpshut	lpstat	lpusers
ls	mkdir	mknod	mount	mv	nice
passwd	pmadm	priocntl	ps	putdev	reject
rm	sac	sacadm	sar	shutdown	strchg
sttydefs	sysdef	ttyadm	ttymon	umount	useradd
userdel	usermod	volcopy	wall		
Enhanced Security:					
admalloc	chlvl	defsak	devstat	getacl	lvlname
rtcpio	setacl	tcpio			

Procedure

To remove commands and privileges from a role, perform the following steps:

1. Enter

   `adminrole` **`-r`** *command_name1*`:`*priv*`,`*priv role_name*

   The *command_name* is the name of a command defined for that role.

Example

Screen 10-7 shows how to remove the **`mount`** privilege from the **`mount`** command for role `assistant`.

```
# adminrole assistant
assistant:   mount:/etc/mount mount
            umount:/etc/umount mount
# adminrole -r mount:mount assistant
# adminrole assistant
assistant:   mount:/etc/mount
            umount:/etc/umount mount
#
```

Screen 10-7. Removing a Privilege from a Command for a Role

Of course, once this privilege has been removed from the database, executing **`mount`** for role `assistant` via **`tfadmin`** will not complete successfully.

Removing a Role

Use **`adminrole`** with the **`-d`** option to remove a role.

Procedure

To remove a role, perform the following steps:

1. Enter

   `adminrole` **`-d`** *role_name*

Example

Screen 10-8 shows how to remove the role `assistant` from the TFM database.

```
# adminrole assistant
assistant:   mount:/etc/mount
            umount:/etc/umount mount
# adminrole -d assistant
# adminrole assistant
UX:adminrole:WARNING:undefined role name “assistant”
#
```

Screen 10-8. Removing a Role

This command removes the role `assistant` from the TFM database along with any commands associated with the role.

Functioning in More Than One Role

In some cases, an administrator may be associated with more than one role. For example, if the **mount** command is present for `darrell` in roles `assistant` and `backup`, then `darrell` would enter

```
tfadmin assistant: mount /dev/dsk/0s0 /x
```

to execute the command with the privileges granted for the `assistant` role, and would enter

```
tfadmin backup: mount /dev/dsk/0s0 /x
```

to execute the command with the privileges granted for the `backup` role.

If the role is omitted from the **tfadmin** command line, **tfadmin** searches the definitions for `darrell` in the TFM database in the order they are specified in the database (that is, the order in which they are listed when printed using the **adminuser** command).

See **adminrole(1M)** and **adminuser(1M)** for complete details on their use.

Assigning Roles to Users

Use **adminuser** with the **-o** option to assign roles to users in the TFM database.

Procedure

To assign a role to a user, perform the following steps:

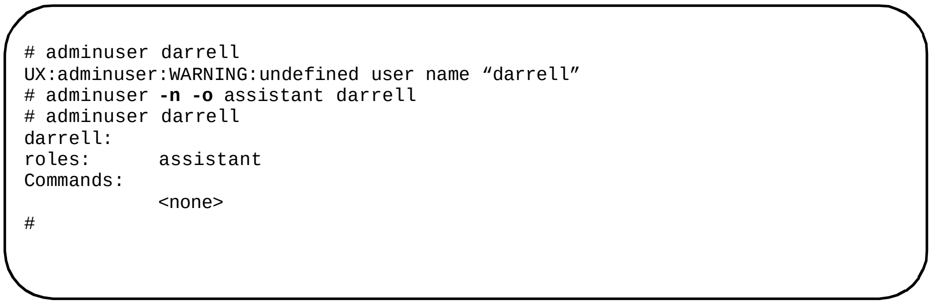
1. Enter

   `adminuser` **`-n -o`** *role_name user_name*

   When you assign a user to a role, make sure the user belongs to a group from which the relevant privileged commands are accessible. The **-n** option is only necessary if the user is not already defined in the TFM database.

Example

Screen 10-9 shows how to assign the role `assistant` to a user, `darrell`, who does not currently exist in the TFM database.



```
# adminuser darrell
UX:adminuser:WARNING:undefined user name “darrell”
# adminuser -n -o assistant darrell
# adminuser darrell
darrell:
roles:      assistant
Commands:
            <none>
#
```

Screen 10-9. Assigning a Role to a User

The first **adminuser** without any options but with the name darrell displays any entries currently in the TFM database for darrell. In this case, darrell does not exist in the database, so a warning message is displayed. Note that this is not the same as saying that darrell is not a user on the system. Because this user is not already defined in the database, we must use the **-n** option to specify that the user is new to the TFM database.

The second **adminuser** assigns the role assistant to the new administrator darrell. The **-n** option shows that this is the first entry in the database for this user. (If darrell had already existed in the TFM database, the **-n** option would have been omitted.) The last **adminuser** displays the entries in the TFM database for darrell, in this case just the role assistant.

NOTE

> Note that an administrator, although properly assigned to an administrative role in the TFM database, must also be able to log in at the appropriate level if the Enhanced Security Utilities are being used. That is, the Operator must be able to log in at SYS_OPERATOR, and all other administrators must be able to log in at SYS_PRIVATE. Note also that users who log on at these levels are by definition administrators even if they are not associated with any roles in the TFM database.

The tfadmin Command

The **tfadmin** command allows administrators and privileged users to execute those commands with which they have been associated via the **adminuser** or **adminrole** commands.

Administrators must include the following system directories in the shell PATH environment variable; the recommended PATH assignment (for use in a **.profile** or execution at the shell prompt) is:

```
PATH=${PATH}:/sbin:/usr/sbin
export PATH
```

Notice that the current directory is not in the PATH. Because the current directory may contain untrusted programs or malicious programs with the same name as a system program, we recommend against including the current directory in your PATH. If you do include it, list it last, as follows:

```
PATH=${PATH}:/sbin:/usr/sbin::
export PATH
```

Including the current directory last reduces the risk of executing an untrusted copy of a system program.

We assume you have logged in using a login that is assigned to the role or user appropriate for the given task; that is, using a login that's assigned to a roleor user that can execute the commands necessary for the task at hand.

Entering Commands into the TFM Database

The system comes with a default TFM database that lists aliases for commands that require privileges, the privileges associated with them, and the roles that are capable of executing them. You will need to associate administrators with roles so those administrators will be able to execute those commands.

Executing Commands with Privilege

Once entries for you exist in the TFM database, use **tfadmin** to execute those commands. The TFM database entries will include any privileges you are granted when executing the commands. The privileges acquired via the TFM database are propagated as fixed privileges if child processes are exec'd by the commands in the TFM database.

The command name used with **tfadmin** in the database does not need to be the same as the actual command, but it must correspond with a command entry in the TFM database.

Procedure

To execute a command with **tfadmin**, perform the following steps:

1. Execute the command as you normally would, but prepend **tfadmin** to the command. Enter:

   tfadmin *command command_arguments*

Example

For example, assume an entry for **mount** exists in the TFM database for darrell. In this case, the **mount** entry in the TFM database has a full path of **/etc/mount** and darrell is allowed to execute it with the **mount** privilege.

darrell would run

```
tfadmin mount /dev/dsk/0s0 /x
```

to mount the device **/dev/dsk/0s0** on the mount point **/x.**

If the full path for the **mount** command were associated with mnt in the TFM database entry for darrell, then darrell would need to execute

```
tfadmin mnt /dev/dsk/0s0 /x
```

to use the **mount** command. Otherwise, **tfadmin** would print an error and exit.

By allowing entries such as mnt to alias for the full pathname, the user is assured that they are executing the desired executable, and not being spoofed by a malicious intruder, who could otherwise introduce their own **mount** program somewhere in the user's PATH.

tfadmin and Other Commands

A number of commands support conditional execution of other commands by using the **tfadmin** command. The commands **sysadm**, **cron**, and **sac** execute some, or in the case of **sac**, all commands using **tfadmin**.

This allows the propagation of privilege to other commands without the need of setting allprivs on **sysadm**, **cron**, and **sac**. The **sac** command executes all commands using **tfadmin** because port monitors require privileges in all cases.

When to Use tfadmin or filepriv

The decision on whether to use **filepriv** or **tfadmin** should be based on considerations of security above all else.

For example, the **filepriv** command assigns fixed privileges to command files. Thereafter, the fixed privileges become a file attribute, and the file will always run with privilege for those users who can access the command file. You may want to assign fixed privileges to commands when you are more concerned that users are accessing a version of a command that you can verify is secure, than with the identities of the people using the command.

On the other hand, **tfadmin** associates privileged commands with a known set of users that you define in the TFM database. Users not in the TFM database cannot execute these commands with privilege. Thus, commands that can have a wide-ranging or destructive effect on the system can be restricted to a group of people you know and trust.

TFM, sh and cron

The **root** and **sys crontab** files execute commands requiring privilege. **cron** sets the $TFADMIN variable to the full path of the **tfadmin** command and exports it before executing the **crontab** entry command line. If the command requires privileges, then those privileges must be listed for the administrator or role using them via **crontab** in the TFM database, and the **crontab** entry must be edited so `$TFADMIN` precedes the command to be executed.

The shell, **sh(1)**, also sets the `$TFADMIN` environment variable.

2
Security Administration

Replace with Part 2 tab

Part 2 - Security Administration

11
Introduction to Security

11
Introduction to Security

How to Use This Chapter

This chapter will help you understand the security needs of your system, and the role that you play in assuring system security as an administrator. It is intended to describe the rationale for security, how the various mechanisms are implemented on the system, and the procedures you should follow to keep your system secure.

This chapter also provides you with instructions on how to install an Enhanced Security system . You should read and understand the instructions for this task thoroughly before proceeding.

Once you are familiar with the mechanisms and procedures described, you'll probably use this chapter as a reference to procedures. If you find you need more detail about certain commands, refer to the specific online manual page for more details.

Security and Your System

The Foundation Set contains a complete set of basic security mechanisms. Several add-on packages provide additional, optional mechanisms that you may wish to use to further upgrade the security of your system. These are:

- Auditing Set, which allows you to add auditing to your system,
- Enhanced Security Set, containing the Access Control List Utilities and the Enhanced Security Utilities.

The Auditing Set is discussed in more detail in the *Audit Trail Administration* guide.

The Enhanced Security Set available with the operating system is designed to provide a B2 level of security functionality, as defined by the Department of Defense and the National Computer Security Center. The Enhanced Security Set also provides some B3 features. The operating system with the Enhanced Security Utilities installed includes special software that provides mechanisms to protect the information on the system. Administrative personnel are responsible for using the mechanisms in the system in a manner that does not violate security.

Who Should Read This Section?

As an administrator you need to understand the meaning of these security-related terms, especially the meaning of the term system and the meaning of the word security as applied to the OS, including security with the Enhanced Security Set installed.

With the Enhanced Security Set installed, administrators need to know how to use those utilities in a way that does not violate security.

This chapter discusses and explains the security policy for the operating system, both with and without the Enhanced Security Set installed and how the system enforces security. Each concept or element is discussed in an introductory-level explanation.

These separate explanations are followed by a discussion of how these separate elements work together to provide security in the section in this chapter entitled *"How the Components of the System Work Together."* This section provides a more detailed explanation of how the security mechanisms work together to implement security.

Some people with advanced knowledge of and experience with the operating system may find the introductory discussions unnecessary, but they provide a background that helps to understand the detailed explanations that follow.

The following section begins the explanation of these concepts with an overview of computer security.

What Is Security?

The need for computer security comes mainly from the multi-user nature of computer systems. If every computer user had a locked office containing a private computer, then there would be no need for a secure operating system. But most computer systems have many users who share resources. The security mechanisms in the Enhanced Security Set are designed to provide for controlled sharing of computer resources, and thus provide security. Even single-user systems require security if they are eventually used to export information.

Security for a computing system means that the information on the system is protected from unauthorized disclosure. For the purposes of this discussion, security also encompasses the concept of integrity, that is, the assurance that information is protected from unauthorized modification or corruption.

There are many ways in which the security of a computer system can be violated. Unauthorized access to read or write files can be the result of:

- the abuse of privileges by users or administrators
- malicious programs that surreptitiously gain privileges or access to files
- idle browsing of files that are inadequately protected by existing security mechanisms

To help administrators protect information from unauthorized access, the operating system offers both basic and enhanced security. The mechanisms supplied in the Enhanced Security Set provide enhanced security. This chapter uses the terms secure and security when discussing the general concept of computer security and when discussing the Enhanced Security Set. This usage is distinguished from the term basic security, which is used in this document when discussing the basic protection mechanisms provided in the operating system without the Enhanced Security Set installed.

A review of basic security will provide a background for understanding the security offered by the Enhanced Security Set.

A computer operating system stores and processes information in the form of electronic data. In doing so, a computer operating system (also known as an operating system, a computer system, or simply a system) provides an interface between you, the user of the computer, and the computer. An operating system provides you with commands, library routines, functions, and programs that allow you to tell the computer how to store and process the information that belongs to you.

To perform its storage and processing functions correctly, a computer system must keep data separate from other data and must also restrict access to data. Computer systems typically have mechanisms that: identify users to the system, keep data separate, and limit access to data. By making access decisions, these mechanisms enforce rules about who can access what and thus supply basic security.

Most computer systems make access decisions based on a unique identity assigned to each user on the system at login. While you are logged in, all data you enter, create, and process belongs to you. Data is stored in named files on the computer system. Each file you own is kept separate from the rest of your files and from the files belonging to other users.

The operating system supplies basic security through the use of the **`login`** and **`passwd`** mechanisms. These mechanisms identify you to the system and put you in control of your data via access permission bits. These bits allow you to control the other users' access to your files; this is security by access control. Additionally you may attach specific privileges to executable files. This allows programs to perform privileged operations without having to become root.

Although the operating system and most other computer systems supply basic security, basic security differs from the concept of security as defined by the Department of Defense and the National Computer Security Center at the B2 level. The mechanisms ensuring basic security are not sufficient for a secure operating system to meet the requirements for receiving a B2 security rating. The operating system with the Enhanced Security Set installed enforces more stringent security.

Elements of the Operating System Security Policy

The security policy of the operating system with the Enhanced Security Set installed prescribes a relationship between access rules and access attributes. The access attributes allow the system to define several distinct levels of authorization, and the access rules provide the mechanism for the system to prevent unauthorized access to sensitive information.

More specifically, the security policy for a computing system running the Enhanced Security Set describes the relationships among five elements. The first two elements are the subjects and objects on a computer system that interact with each other.

- Subjects cause information to flow among objects or they change the system status. A process is represented on the system as a subject when it requests an action.
- Objects are those parts of a computing system that contain or receive information. Examples of objects are data files, program files, directories, named pipes (also referred to as FIFO s), unnamed pipes, symbolic links, memory, terminals, line printers, disks, tapes, and, when they receive information, processes.

Typical interactions are for subjects to create, read, or write objects. Note that a process may be a subject or an object, depending on whether it's requesting an action or receiving information, respectively.

The remaining three elements of the security policy define the ways in which subjects and objects interact. These elements are access attributes, access rules, and privileges.

- The access attributes of a subject or an object define its position within the classification scheme that the system uses to segregate computer users and information on the computer system.
- The access rules embody the policy that segregates information for the system. The system determines whether a subject can access a given object by comparing the access attributes of the subject with the access attributes that are required to access the object. Only if a subject passes all relevant access checks can it access an object.
- Privileges determine a subject's ability to perform certain restricted system calls, commands, and functions. Privileges also allow some processes to override access checks while performing some system calls. The security policy requires that a process has only the privileges it needs to perform its task and that it relinquish a privilege when the privilege is no longer needed. Only system administrators have access to commands with privilege.

As a very simplified approximation, you can think of the security policy as segregating the levels of authorization in a hierarchy, that is, some levels are conceptually "higher" than other levels. A subject can read an object if and only if the subject's level is higher than or equal to the level of the object. A subject can write to an object if and only if the subject's level is equal to the level of the object.

NOTE

Please note that this is a more restricted approach to the generally accepted focus of security policies that consider the levels of objects for write access. In the more general approach, a subject can write to an object if and only if the subject's level is equal to or less than the level of the object.

Actually, the relationship between security levels is much more complex than this simple approximation. Security levels can be disjoint. For more detailed explanations of security levels, refer to the *"Mandatory Access Control"* section of this chapter, Chapte r17, “Administering Mandatory Access Control and Multilevel Directories” in this book, and the “Managing Files Systems Securely” chapter of the *User's Guide.*

In enforcing the security policy, the system assigns access attributes to subjects and objects according to the security level as instituted by the system administrators, and then uses the access rules to ensure that subjects do not access objects for which the subjects do not have the proper access attributes.

The system further restricts the use of certain commands and system calls to subjects (processes) which have the proper privileges.

The Kernel and System Architecture

This section contains information to familiarize you with the system architecture of the operating system, including definitions of terms, an explanation of why well-designed and well-implemented system architecture is a requirement for a secure operating system, and brief explanations of the seven service subsystems within the system architecture.

This section is intended for system administrators and for other persons who need an understanding of the system architecture of the operating system.

System Architecture (SA) Definition

In order for an operating system to be a useful and convenient interface between the user and the hardware, it must provide certain basic services. These services are provided by a number of routines that collectively make up the operating system. Because these routines exist for the purposes of supplying specific services, the operating system has an underlying structure defined by these services. This underlying structure and its design are called the system architecture. The terms system architecture and system structure are used somewhat synonymously.

Software engineers design and implement the system architecture of an operating system so that its parts work well together, just as an architect designs the architecture of a building so that the building does its job, it is structurally sound, and it is aesthetically pleasing. System administrators, system programmers, applications programmers, and users refer to the system architecture to provide a conceptual understanding of the parts of the operating system and the relationships among them.

But sound system architecture is more than a design concept or an aesthetic goal; it is also an important aspect of ensuring that the operating system is secure.

The heart of the system architecture is the philosophy of the engineers who designed it. The the operating system is based on a philosophy of consistency and simplicity, making it easy to understand and use. The operating system appears to the user to consist of files, which are places to store information, and processes, which manipulate information and cause it to flow from one location to another. (In security terminology, a file is an object and a process is a subject when the process is writing to or reading from a file.)

The operating system embodies the following characteristics:

- It treats all files as byte streams.
- It has a hierarchical file system.

- It treats all devices as files to make interfacing to them simple and consistent.
- It provides user services that are simple and easy to use.
- It provides primitives that allow complex programs to be built up from simpler existing programs.
- It hides the hardware architecture from the user as much as possible.

Thus, the operating system must provide services to create, maintain, store, and manipulate files, services to manage the processes that use those files, and services to provide the services. Accordingly, the routines within the operating system architecture are structured into the following seven service subsystems:

- File Management
- I/O Management
- Kernel Utilities
- Memory Management
- Process Management
- System Services
- Access Control

Figure 11-1 graphically depicts the seven service areas and their interactions.

The solid lines delineate the edges of the seven service subsystems. The dashed lines within the seven subsystems delineate the parts of the subsystems. Note that the relationships between the subsystems are greatly simplified.

This figure gives a graphic representation of the structure of the seven service subsystems within the system architecture, and it indicates that there exist only well defined and intended ways for them to interact. The remainder of this section describes the structure of these seven service areas and their interactions.

How System Architecture Relates to Security

In order to be secure, a computing system must enforce a security policy. (See *“Elements of The Operating System Security Policy”* earlier in this chapter.) The security policy for the operating system with the Enhanced Security Set installed demands that the operating system must restrict access to and flow of the information that it processes to prevent unauthorized disclosure or alteration of that information. A secure system relies upon the enforcement of the security policy. Some of the components of the system actively enforce the security policy by, for example, restricting access to files or devices. Other components of the system, such as the seven service subsystems of the operating system, embody the security policy in their architecture.

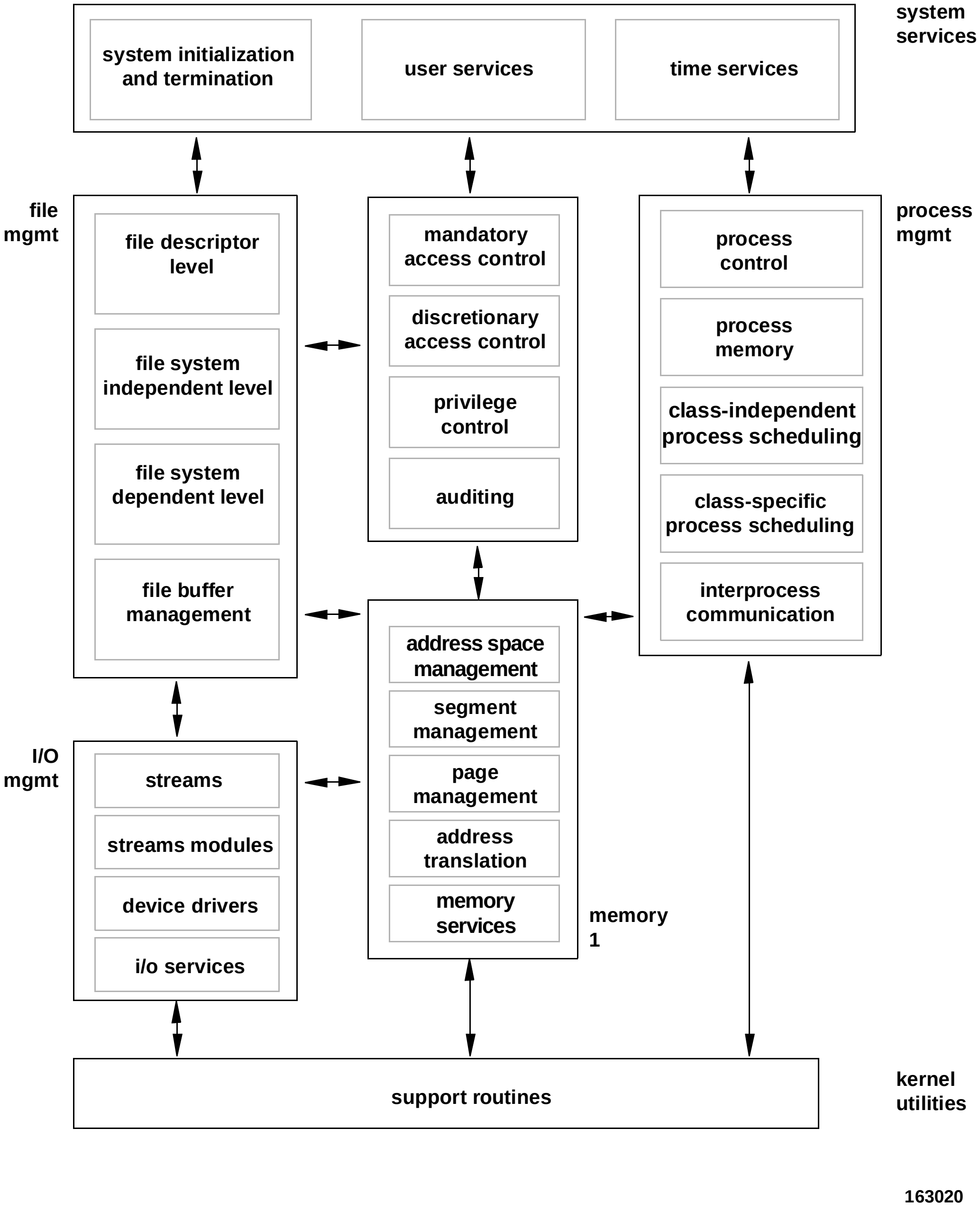


Figure 11-1. The Operating System Architecture

A secure system relies upon the routines in the operating system to perform the tasks for which they were intended and only those tasks. The integrity of a secure system would be violated if routines affected each other in unknown or unwanted ways, because information could be distributed or altered in violation of the security policy. Well-designed system architecture ensures integrity by keeping the seven service subsystems separate from each other and by allowing the areas to interact only in known and intended ways.

Relationship of Modularity to Security

Moreover, the routines within each service subsystem must exhibit software modularity. Modularity means that, within each of the seven service areas defined by the system architecture, the routines are grouped into independent, distinct modules, each of which:

- has a single, well defined function
- has well defined interfaces
- has well defined parameters
- is called whenever its function is required

Modularity within the system architecture enforces security by keeping operating system functions separate and unique. Thus, modular system architecture is a requirement of the security policy for the operating system.

Relationship of Object Reuse to Security

Object reuse refers to the allocation or reallocation of system resources (storage objects) to a subject. Security requires that no system resource can be used to pass data from one process to another in violation of the security policy. This includes internal system resources not normally visible to users such as buffers and caches. In general, the operating system clears these resources of residual data before assigning them to a process, thus assuring that no process intentionally or unintentionally inherits or reads the data of another process.

More specifically, the requirements for security imply the controlled sharing of these resources. For example, the line printer needs to be controlled so that it prints only one user's output at a time. It would be a security violation if one user's output were mixed in with that of another and printed on the same physical sheet of paper. Fortunately, keeping print jobs separate is a rather straightforward task for the line printer system to accomplish. In a similar way, it is easy to prevent more than one user from using a terminal at any given time. While there is nothing to prevent two people from sharing the same login session, internal mechanisms for terminal handling ensure that a terminal is flushed of residual data after a login session before another user can log on to it.

However, the controlled sharing of memory is more difficult to manage. The operating system allows several processes to execute in memory almost simultaneously. Sections of memory may be allocated to one process for a while, then deallocated, then reallocated to another process. The constant reallocation of memory is a potential security problem, because residual information may remain when a section of memory is reassigned to a new process after a previous process is finished with it.

The operating system ensures that such unintended sharing of information can not happen. When memory is reallocated, it is zeroed out completely or reinitialized before it can be accessed by a new process. Thus, there is no residual information in memory carrying over from one process to another.

Finally, the kernel for the operating system is always resident in physical memory. Therefore, the kernel is not subject to being swapped into or out of memory and cannot be violated by any process that resides in physical memory or be tampered with in any way. The loadable modules that make up the kernel can be loaded or unloaded as required.

These are represented as files, and as such are protected by the file protection mechanisms in the system.

NOTE

Note that the administrator need not do anything to enforce the secure reuse of system objects. This requirement is handled by the kernel automatically.

However, the administrator needs to be aware of some methods for reusing removable media such as disks and tapes. As physical objects, these are not under the control of the system software, and it is the responsibility of the administrator to manage their reuse in a secure fashion.

All removable media should be physically labeled with a description of the sensitivity of the data they contain. (See Chapter 13, “Maintaining an Enhanced Security System” for a discussion of security levels and sensitivity labels.)

No removable data storage device should allow reading or writing at a security level lower than that described on the physical label.

If a removable storage device needs to be re-used at a level other than that described on the physical label, the device should be bulk erased and re-labeled to prevent subsequent users from retrieving data from the device.

The Seven Service Subsystems

The security policy, and the system architecture, are conceptually and functionally interrelated. The remainder of this section discusses how the system architecture expresses the security policy in the design, implementation, and organization of its service areas and their constituent routines.

File Management Subsystem

The file management subsystem controls all interactions involving files. Users can create, access, edit, and delete files, as well as cause processes to interact with files. The allocation of space to new files and to existing files as they grow in size is handled automatically by the file management subsystem, which also manages free space.

A discussion of the operating system files will provide a basis for understanding the security implications of file system management.

The Operating System Files

A primary function of the operating system is to support file systems. Although the file tree on a running operating system looks like a single, uniform hierarchy, the file tree is actually composed of subtrees called file systems.

These file systems are mounted onto the root file system (designated by `/`) at special mount points that look like directories. These directories form the connection between the root file system and other mountable file systems. Such a directory is called the root of the file system that descends logically from it, and the file system is usually referred to by the name of the directory used as the mount point.

For example, the **`usr`** file system is the subtree under the **`/usr`** directory.

From a static point of view, a file system may be seen as a collection of files and information about those files on a disk. Dynamically, the **`mount`** command adds a file system to the file tree of a running system.

When the system is booted, it automatically mounts the root file system. Startup routines typically add other file systems, such as **`usr`**, **`home`**, **`var`**, **`proc`**, and **`stand`**.

Disk-based file systems maintain certain data structures on disk that contain information about the file system and its logical and physical structure. While these data structures differ between file system types, all file systems generally have:

- a superblock that contains file system specific and housekeeping information (such as file system size and status, number and status of the `inodes`, information nodes, in the file system, and number of free blocks)
- a series of blocks that identify the files in the file system (referred to most commonly as the **`ilist`** or list of `inodes`)
- a number of storage blocks that either contain data, a list of free blocks, or indirect addresses

An inode contains all the information about a file except its name, which is kept in the directory. Both directories and other files are contained in the storage block part of the file system.

File System Types

The preceding section on file basics describes features common to all operating system files, but the file subsystem allows different file system types to co-exist transparently on a single running system. The file system types supported in the operating system are:

cdfs	CD-ROM File System
fdfs	File Descriptor File System
fifofs	Pipe File System
memfs	Memory Based File System
namefs	Name File System
nfs	Network File System
procfs	Process File System
profs	Processor File System
sfs	Secure File System

specfs	Device File System
ufs	BSD UNIX File System
xfs	Resilient File System

- The **`sfs`** file system type is the only general purpose type that supports all of the access control file attributes available in the Enhanced Security set, specifically, Mandatory Access Control labels and Access Control Lists. These are described briefly later in this chapter. The special-purpose file systems **`cdfs`**, **`fdfs`**, **`memfs`**, **`procfs`**, **`specfs`**, **`fifofs`**, **`profs`**, and **`namefs`**, are special cases. The **`ufs`**, **`xfs`** and **`nfs`** types are available, but do not support the security mechanisms used by the Enhanced Security Set to control access. (See the *"How the System Handles Access Control"* section of this chapter for a discussion of access control.)
- The **`procfs`** file system is a file system that provides an interface to access the image of each active process in the system. Each active process is represented by a directory named by its process ID in the directory **`/proc`**. Processes can be inspected or changed by using standard file system calls. See online manual page **`proc(4)`**.
- The special **`specfs`** and **`fifofs`** file systems are not explicitly mounted and are not directly accessible to user programs. They contain standard code for dealing with devices and named pipes (FIFOs) and allow different file system types to share a common implementation of device files and pipes.

Note that the **`nfs`** file system type is not supported in an operating system with the Enhanced Security Set installed; this type does not support the access control mechanisms used by the Enhanced Security Set.

- The **`ufs`** and **`xfs`** file systems support the full traditional semantics of the operating system files.

Basic operations such as `open`, `close`, `read`, and `write` are mostly the same across file system types. File system types may differ, however, in performance characteristics, the length of filenames they allow, or access algorithms.

The mechanism that allows the operating system to support multiple file system types simultaneously is the File System Switch (FSS). Figure 11-2 illustrates the use of the FSS.

The code that supports file operations is divided into a file-system-independent part and a file-system-dependent part. To do the FS-dependent part, the kernel switches based on the file system type. This FSS architecture makes it easy to add new file system types with characteristics different from traditional time-sharing file systems.

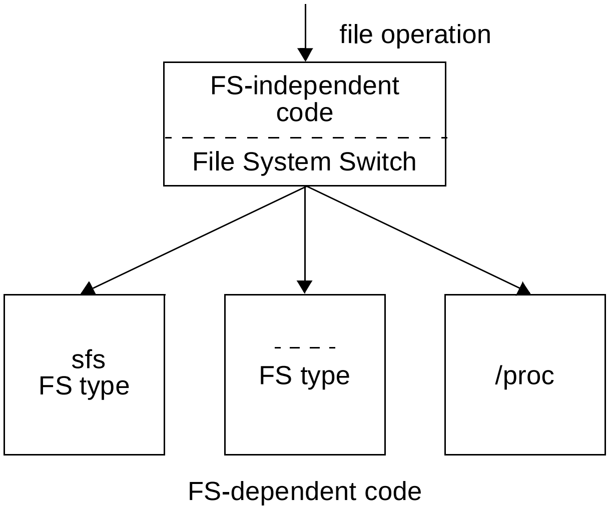


Figure 11-2. File System Switch

File Management and Security

For the enforcement of security, the most important things that the file management subsystem does are: controlling access to files; and guaranteeing the integrity of files. The file management subsystem identifies each file by its `inode`. The `inode` of a file on a secure file system (sfs) contains the following information:

- file owner
- file type
- access permissions
- Access Control List (ACL) information (see *"Discretionary Access Control"* in this chapter)
- pointer to additional ACL information that does not fit in the inode (see *"Discretionary Access Control"* in this chapter)
- security level information, also called Mandatory Access Control (MAC) information (see *"Mandatory Access Control"* in this chapter)
- addresses of the data in the file
- file size
- other bookkeeping information

Each file is identified by a unique `inode`. Thus, the file management subsystem enforces security by keeping files separate and by guaranteeing that any subject which tries to access a file has the proper authorization.

I/O Management

The Input/Output management subsystem controls all the input and output of the computer system. For the enforcement of security, the most important things that the I/O management subsystem does are: managing the transfer of data; and enforcing access controls (the DAC “and MAC” mechanisms) on data while it is being transferred. See the *“Mandatory Access Control”* and *“Discretionary Access Control”* sections in this chapter for more information on MAC and DAC, respectively.

During the transfer of blocks or streams of data, and during character I/O operation, each I/O transaction is completely separate from all others. It follows a well known and well defined path; thus, the integrity of all data is maintained during data transactions. Moreover, DAC and MAC permissions are enforced, thus securing the data from unauthorized access.

Kernel Utilities

The kernel utilities are low-level support routines for use by other parts of the kernel. This includes routines and header files that are used by many different parts of the kernel, but that are not classified into one of the other subsystems. They are general purpose routines which are well defined and as such maintain modularity and thus enforce security.

Memory Management

The operating system uses virtual address space for all user processes and for the kernel. Because two or more processes can share memory by mapping a portion of their virtual address space to the same physical memory, the memory management subsystem controls the contents of physical memory and maintains the data tables used by the hardware to translate from virtual to physical addresses.

By keeping control of memory, the memory management subsystem keeps processes separate in address space. By keeping track of the location of processes in memory, keeping processes separate in memory, keeping memory and the data tables that point to memory updated, maintaining the integrity of memory and data tables, and zeroing out or properly reinitializing memory before reuse (see *“Object Reuse”* in this chapter), the memory management subsystem enforces security.

Process Management

The process management subsystem controls all the actions of processes, including lightweight processes. The process management subsystem schedules and synchronizes processes, keeping them separate and keeping them from running into each other. The process management subsystem also manages inter-process communications (IPC), allowing processes to communicate only in the ways for which they have proper authorization.

System Services

The Systems Services manage system initialization and termination, timer services, and user services.

Access Control

The Access Control mechanisms are those parts of the system that administer and enforce Discretionary Access Control and Mandatory Access Control in accordance with the security policy. The access control subsystem also contains the auditing mechanism.

Identification and Authentication

Identification mechanisms are employed by a user to prove who the user is, and authentication mechanisms are used by the system to verify the user's claim of identity. A user gains access to the system by entering a login name that is verified by the system via a password. The compromise of this password could allow another person access to the system and to all resources authorized for the password's real owner.

In order for the system to be secure, the system must ensure that only authorized users can log in and that they log in only as they are authorized to log in. Identification is the mechanism by which, via the login name and level information, the system recognizes a user as legitimate for the operating system at a given security level. Authentication is the mechanism by which, via the password, the system verifies the identity of a user. Together, these mechanisms are known as Identification and Authorization (I & A).

When a user logs in and his or her identity is verified by the system, certain information about the user's access to information is also revealed, namely the access attributes of the user. Only if the user's access attributes meet the requirements for accessing an object can the user access the object.

The I & A mechanisms prevent unauthorized people from logging into your secure system, and they ensure that users log in only to areas for which they are authorized. These mechanisms supply the “who” information to the system so that the system can make decisions about and enforce the “who can access what” parts of the security policy. (See “*Discretionary Access Control*” and “*Mandatory Access Control*” in this chapter for information on how the system uses this information to supply security.)

The programs that enforce identification and authentication are:

- **`login`**
- **`passwd`**

The command that lets users find out about other users logged in on the system is:

- **`listusers`**

The administrative commands for managing identification and authentication are:

- **`defadm`**
- **`logins`**
- **`useradd`**
- **`userdel`**
- **`usermod`**

The library routines for managing identification and authentication are:

- `getpwent`
- `getpwuid`
- `getpwnam`

Because the all-powerful superuser privilege has been replaced by discrete process privileges, **`su`** is no longer always a means to obtain privileges. For a complete discussion of process privileges, see the *"Process Privileges"* section in this chapter.

For a more complete discussion of identification and authentication, see *"Identification and Authentication"* in the *"How the Components of the SystemWorkTogether"* section of this chapter.

How the System Handles Access Control

The security policy of the Enhanced Security Set prescribes a relationship between access rules and access attributes. The access attributes allow the system to define several distinct levels of authorization, and the access rules provide the mechanism for the system to prevent unauthorized access to sensitive information.

The system enforces access control by means of the Discretionary Access Control mechanism and the Mandatory Access Control mechanism.

Access to a file is determined by the file's absolute pathname. The kernel determines whether or not to allow a process the kind of file access requested (read, write, execute/ search) based on the user and group IDs associated with the process, the process's level, the privileges (if any) associated with the process, and the discretionary and mandatory controls associated with the file and all the directories that make up the absolute pathname of the file. These access checks are performed at the time the file is opened, rather than at the time a read or write is actually attempted.

For example, if the file **`/usr/src/cmd/mv.c`** is readable by a user, but the directory **`/usr/src/cmd`** (or any other directory in the path) is not searchable by the user (that is, the user does not have search permission on **`/usr/src/cmd)`**, then **`mv.c`** cannot be read.

When the system makes access checks to control access, Mandatory Access Control checks are made first, then Discretionary Access Control checks. However, because Discretionary Access Controls are conceptually easier to understand, they are discussed first.

Discretionary Access Control

Users on a computer system typically share commands, programs, library routines, and files. For example, all users need to share the user level commands in the operating system, such as **`ls`**, **`cp`**, **`ed`**, and so on. Also, users in the same group or in related groups often may want to share some of the same data or text files. The sharing of objects introduces some potential security violations. In order to maintain security, the security policy requires that the system oversee the sharing of objects in a known and controlled manner.

Discretionary Access Control (DAC) provides for the controlled sharing of objects among subjects. DAC is part of the "who can access what" mechanism. With DAC, the owner of an object can choose to grant access permissions to other users; that is, the segregation of information and the prevention of unauthorized access to information is set according to the discretion of the owner of the information.

The Enhanced Security Set has two complementary DAC mechanisms: the operating system file permission modes and Access Control Lists (ACLs). The operating system file permission modes are retained from previous releases of the operating system for compatibility. Administrators already familiar with the operating system file permissions will find that this mechanism still works as expected. ACLs are a feature in the Enhanced Security Set. They are designed to satisfy B2 requirements and to be compatible with the operating system file permission modes.

A given combination of permission mode bits on a file are directly translated into a basic ACL for that file; it provides identical protection. The ACL on a file can be displayed by invoking the **`getacl`** command. The owner of the file can add more users and groups to the basic ACL by invoking the **`setacl`** command. Refer to the “Using the File System” chapter in the *User's Guide* and the **`getacl(1)`** and **`setacl(1)`** manual pages for more information concerning these commands.

This ACL scheme supports finer control than file permissions alone by providing the ability for the owner of an object to grant or deny access by other users to the granularity of a single user. All DAC information may be changed in one atomic operation with the **`setacl`** command, avoiding the possibility of an intermediate insecure state.

The operating system's ACLs also allow specification of access rights to members of groups as defined to the system in the administrative file **`/etc/group`**. ACLs can be arbitrarily large; that is, the number of ACL entries is not limited by the system. The system administrator can set the maximum number of entries per ACL by setting a tunable parameter. (Naturally, as ACLs get larger, processing gets slower, which induces a practical limit on the number of ACL entries.)

Much information about a file's accessibility is given by its mode, which tells what kind of file it is, and its discretionary permissions, as shown in Figure11-3.

The first character of the mode gives the file type: either a regular file, symbolic link, directory, pipe, character device, or block device.

The next nine characters of the mode are three sets of three permissions. For the owner of the file, the group of the file, and for all others, the mode tells whether a user can read, write, and execute the file. (For directories, execute permission tells whether the user can search the directory.)

As stated above, the file mode indicates the maximum discretionary permissions for a file system object and provides the basic ACL for the file; these permissions may be further restricted if additional entries are made in the ACL associated with that file.

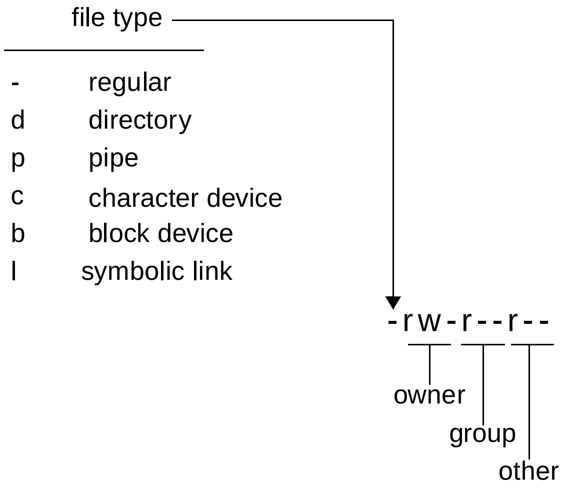


Figure 11-3. File Mode

The eleventh character of the mode indicates whether a basic or extended ACL is associated with a file; if this character is a space, only the basic ACL exists for the file; if this character is a plus sign (+), one with additional entries exists. Both can be examined and modified using the commands `getacl` and `setacl`.

To further explain the basic ACL, there is a direct mapping between the entries in the initial ACL and the file mode bits. When a file is created, the permission mode bits and the basic ACL are generated. The basic ACL that is generated at time of file creation has four entries, `user`, `group`, `class`, and `other`. The **`owner`** mode bits are always equal to a **`user`** ACL entry for the object's owner. The **`other`** mode bits are always equal to the **`other`** ACL entry, of which there is only one in any ACL. The **`group`** mode bits are initially equal to the **`class`** ACL entry, of which there is only one in any ACL and to the **`group`** ACL entry for the owning group. The significance of the **`class`** entry is discussed later in this section.

The basic or initial ACL can be extended by specifying additional **`group`** and **`user`** entries. Permissions for multiple groups can be specified in **`group`** entries, while additional **`user`** entries can be used to grant or deny access for specific logins.

The ACL can have additional entries based on the **`default`** ACL entries associated with the directory in which a file is created. These **`default`** directory ACL entries indicate the ACL entries that are added to any file created in that directory.

The following is a brief description summarizing how an ACL for a file is generated based on the information discussed. Whenever a file is created, the system initializes an ACL for the file that contains a **`user`** entry for the owner permissions, a **`group`** entry for the owning group permissions, a **`class`** entry for the owning group permissions, and an **`other`** entry for the other group permissions. Additional entries may be added by the user, or as a result of default entries specified on the parent directory.

The **`getacl`** command reports the entries in the ACL. As indicated, each ACL has at least four entries, one each corresponding to the file mode permissions for **`owner`**, **`group`**, **`class`**, and **`other`**.

File permission bits for user and group are translated into special cases of these entries:

- The bits representing owner permissions are represented by a **`user`** entry without a specified user ID.
- The bits representing group permissions are represented by a **`group`** entry without a specified group ID.

In an ACL, there must be one each of these special user and group entries. There may be any number of additional **`user`** entries and **`group`** entries, but these must all contain a user ID or group ID, respectively. There is only one **`other`** entry in an ACL, representing the permission bits for permissions to be granted to other users. The following is an example of the output of the **`getacl`** command for a file named **`junk`** owned by **`user_1`** in **`group_1`** whose permission mode bits are `-rw-r--r--:`

```
# file: junk
# owner: user_1
# group: group_1
user::rw-
group::r--
class:r--
other:r--
```

If **`user_2`** and **`user_3`** and **`group_2`** are added to the ACL by using the **`setacl`** command, **`getacl`** would produce the following output:

```
# file: junk
# owner: user_1
# group: group_1
user::rw-
user:user_2:r--
user:user_3:r--
group::r--
group:group_2:r--
class:r--
other:r--
```

The mode bits on the ACL **`class`** entry are significant. The **`class`** entry mode bits are determined by the **`group`** mode bits for the file. Therefore, the **`group`** entry for the owning group and the **`class`** entry in the basic ACL are identical. When only a basic ACL exists for the file, you can think of the **`group`** and **`class`** bits as being the same. Once additional users and groups are added to the ACL, the owning **`group`** bits take on a separate identity from the **`class`** bits. If the **`chmod`** command is invoked to modify DAC permission bits when additional ACL entries exist, it effectively modifies the **`owner`**, **`class`**, and **`other`** mode bits. Please note that in the case of the DAC group permission bits, it is the **`class`** bits that are modified and not the owning **`group`** bits in the ACL entry. The only way to change ACL entries (except for the ones representing **`owner`** and **`other`**) is by using the **`setacl`** command.

The purpose of the **`class`** entry bits is to define the maximum permissions available to the users and groups that may be added to the ACL. For example, if the group permission bits for a file are `r--`, the output of the **getacl** command would show a **class** entry in the ACL with those same permissions associated with it. All additional users and groups will effectively have no permissions in excess of `r--` regardless of what permissions are indicated in their ACL entry. The specified permissions in the ACL entry for a user or group can only serve to further restrict permissions since the **class** entry derived from the **group** permission bits effectively sets the upper bound for the permissions on additional users and groups. In the above example of an ACL with additional entries, if **user_2** were added to the ACL with `r-x` permission bits, the ACL displayed by the **getacl** would look as follows:

```
# file: junk
# owner: user_1
# group: group_1
user::rw-
user:user_2:r-x#effective:r--
user:user_3:r--
group::r--
group:group_2:r--
class:r--
other:r--
```

However, the effective permissions for **user_2** would be `r--` as determined by the mode bits in the **class** entry.

ACLs are implemented through the secure file system type, **sfs**. See *“The Operating System Files”* and *“File System Types”* in this chapter, and the “Using the File System” and “Managing Files Securely” chapters of the *User's Guide.*

Other information about the file is also stored in the mode; for a complete description, see **chmod(1)** and **chmod(2)**.

Discretionary Access Checking Algorithm

The kernel performs access checking every time a subject attempts to open an object for access.

The subject, and process, has a security level, a user ID, and a primary group ID (and possibly supplementary group IDs), associated with it. The object has a set of permission bits, a security level, and possibly an ACL.

First, starting with the leftmost component of the file's pathname, security levels are checked. The kernel checks security levels based on a “read-down, write-equal” mandatory policy. (See *“Mandatory Access Control”* following this section for a discussion of security levels and Mandatory Access Control checks.)

Then, a discretionary access check is performed to determine if the process requesting access to a file has permission to access the file in the mode (read, write, and/or execute/search) requested. Each access mode requested is checked separately according to the steps in the algorithm that follows:

```
if effective uid of process matches owner id on object
    if requested access mode matches bits set in
            the user entry representing the owner
        then the requested access is granted

else if effective uid of process matches the uid in an
                 additional user entry
    if requested access mode matches bits set in that user entry
                         and matches bits set in the class entry
        then the requested access is granted

else if any group in the group set of process matches the owner gid or
                the gid of any additional group entry
    if requested access mode matches bits set in any combination
                of one or more group entries
                and matches bits set in the class entry
        then the requested access is granted

else if requested access mode matches a bit set in the other entry
    then the requested access is granted
else the requested access is denied
```

Note that in the step checking for a match against group, the permissions for all **`group`** entries are effectively `or`*ed* together. For example, if a user is a member of groups A and B, and requests read or write access to a file which allows read access to group A and write access to group B, the requested read or write access will be granted.

Also note that before DAC grants permission based on additional user or group entries, the access must be allowed by the **`class`** entry, that is, the **`class`** entry masks off permissions for additional **`user`** and **`group`** entries. See “Using the File System” in the *User's Guide* for more details. These checks are performed on every component of the pathname, including the object itself.

Objects with ACLs

ACLs are associated with each file system object on a Secure File System (**`sfs`**)and IPC object. ACLs for file system objects are stored in the associated `inode`. ACLs for IPC objects are stored in an internal structure associated with the instantiating of the IPC object.

DAC Commands and System Calls

The commands that a user can invoke to manipulate and read DAC permissions are:

- **`getacl`**
- **`setacl`**

The system calls that a program can invoke to manipulate and read DAC permissions are:

- `acl`
- `aclipc`
- `chmod`
- `chown`

- `msgctl`
- `semctl`
- `shmctl`

The library function for reading and sorting ACL information is:

- `aclsort`

The commands and system calls that are affected by both DAC and MAC are discussed in the *"Common Access Control"* and *"DAC and MAC Effects on Other Commands and System Calls"* sections of this chapter. For a more complete discussion of DAC, see *"Managing Files Securely"* in the "User's Guide."

Mandatory Access Control

In order to maintain strict separation of subjects and objects, the security policy requires that the sharing of objects should not be left solely to the discretion of the owner of the object. Mandatory Access Control (MAC) also provides for the controlled sharing of objects among subjects. Unlike DAC, however, MAC is controlled by the administrator and enforced by the system, and it is independent of the object owner's ability to grant discretionary access.

The system enforces MAC by means of a security level recorded in a sensitivity label. This security level is the result of the finer levels of division and the correspondingly more stringent access checks required. Every subject has an assigned security level; so does every object. The security level of an object is the same as that of the subject that created it. The security level is a combination of a hierarchical classification (for example, **`Classified`**, **`Secret`**, **`Top Secret`**) and zero or more non-hierarchical categories (for example, **`Project Alpha`**, **`Project Sigma`**, **`Project Phi`**). Thus, a user logged in as **`Top Secret;Phi`** will create files and programs that will also have **`Top Secret;Phi`** in their sensitivity labels. Security levels are given in the form:

classification`;`*category_1*`, . . . ,` *category_n*

For a more complete discussion of MAC, see "Managing Files Securely" in the *User's Guide* and the "Mandatory Access Control" chapter in this guide.

Mandatory Access Control Checks

Access to objects by subjects is constrained by the dominance relationship between MAC security levels. A subject can access an object for reading if and only if its security level dominates the security level of the object. Security level S1 is said to dominate security level S2 if the hierarchical classification of S1 is greater than or equal to that of S2, and if the non-hierarchical categories of S1 include those of S2 as a subset.

In order to read an object, the subject's clearance must dominate the level of the object. For one level to dominate another, two conditions must be met:

- the classification of the first must be at the same or a higher hierarchical classification than the classification of the other
- the set of categories of the first must be a superset of the set of categories of the other

In the case of read access for IPC objects and pipes (named and unnamed), the meaning of dominance is restricted to the level of the subject being equal to that of the object.

In order to write an object, the subject's level must be equal to that of the object.

In order to search or execute an object, in general, the subject's clearance need only dominate that of the object; that is, this is implemented following the same rules as read access. Search or execute access does not apply to pipes or IPC objects.

Accessing objects contained in multilevel directories is a special case. Access depends first on which multilevel mode you are in. If you are in virtual mode, your level must be the same as that of the object for you to access it. This is a by-product of your view of the multilevel directory when you are in virtual mode. If you are in real multilevel directory mode, the access decision remains unchanged from the policy stated above.

The kernel enforces MAC. There are seven subdivisions of the MAC software components. These are:

- defining security levels, classifications, and categories
- dealing with subject levels
- dealing with object levels
- dealing with multilevel directories
- dealing with device labels
- dealing with mounted file system range
- system calls and commands needing to be modified to handle DAC and MAC information.

Each of these subdivisions will be discussed separately.

Level Definition and Initialization

In order for MAC restrictions to function effectively, administrators must be able to enable and disable classifications and categories, thus defining the levels on the system and their range. The creation of new subjects, and the creation or modification of objects is limited to levels within the defined range.

Moreover, in order to be able to use MAC security level information, users must be able to determine the security levels — classifications and categories — currently supported by the system.

The commands which allow administrators to define and initialize MAC security level information are:

- **`chlvl`**
- **`lvldelete`**
- **`lvlname`**
- **`maccnv`**

NOTE

> Although the **`chlvl`** command can be used by administrators with the proper privilege to change the level of a mounted file system, it should not be used to change the level of the root vnode of the file system while the file system is mounted. Rather, to change the level of the root of a file system, an administrator should (1) unmount the file system, (2) change the level via **`chlvl`**, and (3) remount the file system.

The command that allows users to determine MAC security level information is:

- **`lvlprt`**

The system calls that allow users to compare MAC security level information are:

- `lvldom`
- `lvlequal`

The functions that allow users to print MAC security level information and to convert from full level names to aliases and vice versa are:

- `lvlin`
- `lvlout`

The function that allows users to validate security levels is:

- `lvlvalid`

Subject Levels

When a user logs in, a shell process is created. This process is assigned the security level of the user as established by the **`login`** and **`password`** programs. Each succeeding process or file inherits the security level of the shell. Users can display the security levels of their processes; administrators can display the security levels of all processes running on a system. Unprivileged users can get information only on processes that they own and dominate.

The commands that allow both administrators and non-privileged users to determine information about subjects' MAC security levels are:

- **`ps`**
- **`whodo`**

The system call that a program can invoke to get its own process level is:

- `lvlproc`

If the calling process is privileged, `lvlproc` can also be used to change that process's level.

Object Levels

Objects are those parts of the system that contain or receive information. Objects include regular files, special device files, peripheral devices, memory, symbolic links, unnamed pipes and named pipes (FIFOs), regular and multi-level directories, and processes.

The privileged command which users can invoke to change the level of a file within their authorized levels is:

- **`chlvl`**

The system calls which programs can invoke to get or set the levels of objects (including devices) are:

- `flvlfile`
- `lvlfile`
- `lvlipc`

Multilevel Directories

A multilevel directory (MLD) is a directory, such as **`/tmp`**, that allows untrusted processes to create files at different security levels within that same directory. An untrusted process will be able to access those parts of the MLD and its contents that have the same security level as the untrusted process. A process in real mode can also read files that it dominates in an MLD. MLDs exist largely to provide compatibility with previous releases of the operating system.

The system calls that a process can invoke to create and find information about MLDs are:

- `mkmld`
- `mldmode`

The function for determining information about MLDs is:

- `stat`

The command for dealing with MLDs is:

- **`mldmode`**

Secure Device Handling

The system must also restrict the levels of information being stored on or sent to allocated devices. To enforce MAC on devices, the system uses the mechanism of device labels.

Administrators need to be able to determine the allocation of secure devices. The commands for administering secure device levels are:

- **`devattr`**
- **`getdev`**
- **`putdev`**

The command for administering secure device levels is:

- **`admalloc`**

Users and processes must be able to work with secure device labels. The system calls for manipulating device labels are:

- `devstat`
- `fattach`
- `fdevstat`
- `getmsg`
- `getpmsg`
- `ioctl`
- `open`
- `putmsg`
- `putpmsg`

The system calls that change data but not the attributes of the device are:

- `mmap`
- `read`
- `write`

The functions for manipulating secure device labels are:

- `devalloc`
- `devdealloc`

Mounted File System Range

The system also applies MAC to mounted file systems in order to restrict the security level of information stored or sent to the file systems. The system specifies the level range for the duration of the mount.

The administrator command for dealing with mounted file system range is:

- **`mount`**

The system call for dealing with mounted file system range is:

- `lvlvfs`

Common Access Control (CAC)

The access control policy of several features and subsystems combines the rules of DAC and MAC to such a degree that the descriptions of their functionality do not fall neatly into either class. This section describes such subsystems.

Trusted Import/Export

Data can be imported into or exported from a system running the operating system with the Enhanced Security Set installed by either disk or tape (hereafter referred to as the archive medium). However, the only trusted mechanism is via specially-provided utilities.

On an operating system installation, the normal way of doing backups and restores is to use the commands **find** and **cpio** or **volcopy.** On a system running the operating system with the Enhanced Security Set installed, the **tcpio** (trusted cpio) command is used with the **find** command. Using the **tcpio** command instead of **cpio** allows you to save and restore the MAC and DAC attributes of files. You can still use the **volcopy** command to save file systems when the Enhanced Security Set has been installed, since this command saves the entire file system image.

To perform a trusted back-up, the administrator would typically first use **find** to determine the pathnames for all files satisfying certain criteria. **find** understands the additional MAC information and provides the ability to select files based on level or level range. The user then pipes these pathnames to **tcpio** to be written to the archive medium.

tcpio saves (on the archive medium) critical system-wide information that is used, upon restoring the data on the archive medium, to validate the MAC, user ID, and group ID data. For each file, **tcpio** saves the label associated with each file, as well as the file data.

Before writing a file or its associated per-file information to the archive medium, **tcpio** checks the level of the file against the level range of the device on which the archive medium is mounted. The file will not be written to the medium if its level is not within the device range of the medium.

To restore files from a **tcpio** archive, an administrator must ensure that the device on which the medium is mounted has a device level range covering the range of levels of data on the archive medium. No check is made of the level of the file being extracted against the device level range; however, the level of each file is checked against the level range of the file system to which it is being written. If the level of the given file is not within the range of the target file system, the file is not extracted — a special operation to override this restriction is required of the administrator.

During the extraction, the validation data is read first; then file data is extracted file-by-file. However, if the system database files have changed since the medium was written, the files are not extracted, by default. By using one of the options **-T** or **-n**, the administrator can force the extraction in various ways (remap the changed IDs to the specified definition, force the extraction regardless of mismatch of IDs, use the ID definition contained in the validation data stored on the archive, and so on).

NOTE

Please note that using the remapping options to **tcpio** may violate the system security policy.

Note that the **tcpio** utility may be used by non-privileged users. However, allowing users to use tape utilities of any kind runs the risk of permitting violations to the system security policy in that files at one level may be stored to tape and then restored at a different level. This is prevented by making archive devices unavailable to non-privileged users.

When archive devices are used by a non-privileged user, no MAC information is saved, since a non-privileged user can operate (that is, write) only at the level of the current user process. When read back in, files are created at the level of the user executing the process. The **tcpio** utility may be useful to non-privileged users, because it is the only means provided to non-privileged users to save and restore files with their security attributes.

Chapter 18, "Trusted Backup and Restore" in this book describes in more detail how to perform the procedures mentioned in this section.

LP Print Service

The LP print service meets the Department of Defense requirements for handling human-readable output.

When a user submits a print request, the sensitivity level associated with the job depends on the level of the user submitting the request. The request is assigned the same level as that of the user's process.

In deciding whether or not to accept the print request, the LP software checks the level of the job against the level range of the printer. If this range check fails, the request is not accepted.

The sensitivity level of the job is printed at the top and bottom of each page, unless overridden using the **-o** nolabels option to the **lp** command. The use of this option is an auditable event. If the text of the level is extraordinarily long, the level plus as many categories as possible will be printed, with the overflow condition indicated with ellipses. If truncation occurs, an audit record is automatically generated.

The level of the job plus a unique (random) job identifier assigned by the LP system are always included on the banner and trailer pages for the job. There is no truncation or overriding of levels. If necessary, multiple pages are produced. The unique job identifier is kept internal to the LP system. It is never reported to the user except on the banner and trailer pages. This is done to prevent spoofing of the identifier by an unscrupulous user.

The **lpstat** command reports the status of jobs waiting in the print queue. When used by an unprivileged user, only jobs at that user's current level are reported. An option enables appropriately privileged administrators to see the levels of all the jobs in the queue. [See **lpstat(1)**.]

More information on using printers on a system with the Enhanced Security set installed can be found in Chapter 15, "Administering Printers, Terminals, and Devices" and the "Advanced Print Service" chapter in volume 2 of System Administration.

DAC and MAC Effects on Other Commands and System Calls

In order to accommodate the addition of MAC and the addition of ACLs to DAC in the Enhanced Security Set, some system calls and commands have been modified to handle security level and ACL information. These modifications are transparent to users and administrators, but the information that they supply is subject to DAC and MAC restrictions.

The following commands will be affected when the Enhanced Security Set is installed:

- **at**

- **batch**
- **cp**
- **cpio**
- **crash**
- **crontab**
- **df**
- **du**
- **find**
- **fsck**
- **ipcs**
- **ln**
- **ls**
- **mail**
- **mailcheck**
- **mailx**
- **mesg**
- **mkdir**
- **mkfs**
- **mknod**
- **mv**
- **ncheck**
- **news**
- **ps**
- **sysdef**
- **volcopy**
- **write**

The following system calls have been modified:

- access
- exec
- mkdir

The **cron** clock daemon will also change its behavior to accommodate MAC and DAC.

The effects of MAC and DAC on some specific commands listed here are discussed in the following three sections.

Mail Subsystem - mail, mailx, and mailcheck

When sending mail with either **`mail`** or **`mailx`** under the operating system with the Enhanced Security Set installed, the mail message is assigned the level of the process which sends it. **`/var/mail,`** the directory in which mail messages are stored, is a multilevel directory. A user may have mail messages waiting at several different levels.

When reading mail with either **`mail`** or **`mailx,`** only mail messages at the user's current level will be shown. The command **`mailcheck`** is provided to give the user the ability to check for the presence of mail at levels dominated by the level of the user's current process but not to read it.

To read mail stored at other levels, the user must log in at each appropriate level. All files created by the mail system on behalf of the user, including **`mbox`** and **`dead.letter`**, are created at the user's current level.

System-wide News - news

MAC places restrictions on the way that **`news`** works. The directory **`/var/news`** is at MAC alias `SYS_PUBLIC`. Placing it at this level will allow all users on the system, both administrative and non**`-administrative`**, to read news, since users at all levels dominate `SYS_PUBLIC`.

However, in order for someone to write into **`/var/news`**, the user would have to be at `SYS_PUBLIC`, since the security policy requires that levels be equal for write access. But no user is assigned to level `SYS_PUBLIC`. For a user to post news, the user will have to mail the message to the Site Security Officer, who will read the message to check the sensitivity of its contents. If the Site Security Officer decides that the news can be posted for everyone to read, the SSO can post the message in **`/var/news`** while at `SYS_PUBLIC`.

File Manipulation - cp, mv and ln

These commands retain their semantics as on UNIX Systems. The assignment of the owner, group, and permission bits remains unchanged.

However, these commands have additional rules for handling the MAC and DAC attributes of files. A summary of the rules appears in the following table:

When performing this kind of **cp**, **mv**, or **ln**	the DAC attributes of the destination file are	the MAC attributes of the destination file are
Case 1: **cp** to an existing file in the same or a different file system	retained	retained
Case 2: **cp** to a new destination file, or **mv** to a new or existing destination file in a different file system	assigned to have the owner and group of the invoking subject, and the access permissions of the source file	assigned to be the level of the invoking subject
Case 3: **mv** or **ln** to a new or existing destination file in the same file system (no new object is created)	assigned to be the same as the source file's	assigned to be the same level as the source file

File Listing - ls

The **ls** command enables users to determine DAC and MAC information. When **ls** is used to display the mode information of a file, as with the **-l** option, a "+" is appended to the display of the mode bits to indicate the presence of additional ACL entries. (The command **getacl** is used to display the ACL.)

Two options, **-Z** and **-z**, are provided to display respectively the fully qualified level of the file and the alias name of the level of the file(s).

For more information on MAC, see Chapter 17,"Administering Mandatory Access Control and Multilevel Directories" and "Managing File Systems Securely" chapters of the *User's Guide.*

Process Privileges

Having a privilege means having the ability to override access controls to perform a sensitive system operation.

In order for the system to be secure, the security policy requires that this single superuser privilege be subdivided into a set of privileges known collectively as process privileges. Privileges are no longer associated with user IDs, but with processes and executable files. The system still recognizes **root** as the all-powerful user, by associating all privileges with any process with a user ID of `0`.

Any user-level command, kernel module, or system call that requires a privilege in order to be executed is considered part of the system. Similarly, any system data file, directory, device special file, named pipe, or symbolic link that requires a privilege in order to be accessed is considered part of the system.

This functionality is supported by the Super User (SUM) policy module. For systems with the Enhanced Security set installed, the Least Privilege (LPM) policy module provides a more restrictive policy that specifies that processes execute with the least amount of privilege to function correctly, and no more.

A process that accesses these privileged components of the operating system will require the appropriate privilege. Thus, any process that runs with privileges is considered part of the system, because the privilege it has gives it the authority to perform some sensitive action. A misused privilege could compromise security. Thus, privileges and privileged processes must be carefully controlled.

The operating system distinguishes between general users, who have no access to commands with process privileges, and privilege users and administrators, who have such access via the Trusted Facility Management mechanism. (See Chapter 10, "Trusted Facility Management" in this book.)

The division of the ability to perform certain privileged tasks is necessary to enforce the principle of least privilege. According to that principle, a process should never have more privileges than it needs at a given time.

Each executable file can have two sets of privileges associated with it; they are fixed (SUM and LPM) and inheritable (LPM only) privileges. These sets are disjoint, that is, a privilege can not be defined as both fixed and inheritable for the same file. If an executable file does not require any privileges then both sets are empty.

Each running process can have two sets of privileges associated with it; they are maximum and working. The maximum and working set of privileges for a new process are identical.

The following explains how file and process privileges are related. The executable file's privileges are used to pass on privileges to the process when that program is executed, that is, via an `exec` system call. The file's privileges behave as follows:

- Fixed privileges are always given to the new process, independent of the calling or parent process's privileges (SUM and LPM). Inheritable privileges exist in the new process only if they existed in the previous process (LPM only). The inheritable privilege set is not supported when using the SUM policy module. The process's privileges are always considered to be inheritable when using the SUM module.

CAUTION

> Privileges associated with a file are removed when the validity information for the file changes (for example, when the file is opened for writing or when the modes of the file change). This removes the file privileges and the privileges must be set again in order for the command to run with privilege. This validity checking is optional. Refer to the **`filepriv(1M)`** and **`initprivs(1M)`** online manual pages for further information.

The combination of the fixed privileges on the executable file and the calling process's privileges become the maximum set of privileges for the new process when using the SUM privilege policy module.

The combination of the fixed privileges on the executable program file and the inheritable privileges on that file that are in common with the calling process's privileges becomes the maximum set of privileges for the new process when using the LPM privilege policy module. The process's privilege sets are defined as follows:

- The maximum set contains all the privileges granted to the process either as fixed (SUM and LPM) or inherited (LPM only) privileges.
- The working set contains all the privileges currently being used by the process. This set can be adjusted before making sensitive system calls, using the **`procprivl(2)`** system call, to enforce the principle of least privilege.

To summarize, after a `fork`, the privileges of the parent and child processes are identical; however, when an `exec` system call is performed, the privileges of the resultant process are determined from the maximum set of privileges of the process performing the `exec` and from the privileges associated with the executable file.

The following steps occur to determine the new maximum and working sets:

A. The maximum set of privileges of the calling process is intersected with the inheritable privileges of the file being executed. Privileges common to both sets are inherited (LPM case). When the SUM module is installed, all of the privileges in the maximum set of the calling process are inherited.

B. The union of the privileges from step A and the fixed privileges of the executable file becomes the maximum and working sets of the new process.

Process privileges and the principle of least privilege enhance the access control the system exercises over the execution of sensitive actions, providing enhanced security.

When the Enhanced Security package is installed, a calling process may access another process only if the calling process has DAC and MAC access and has a maximum set of working privileges which dominates the maximum set of working privileges of the target process.

The administrative commands that manage privileges on files are:

- **`filepriv`**
- **`initpriv`**

The system call that manages file privileges in the kernel privilege table is:

- `filepriv`

The user-level system calls for dealing with process privileges are:

- `exec`
- `fork`
- `procpriv`
- `procprivl`

For more information on process privileges and the principle of least privilege, see the *"Privilege Mechanism"* section of Chapter 9, "Administering Privilege" in this book and also the *"Security Considerations"* section in the "Directory and File Management" Chapter of the *PowerMAX OS Programming Guide.*

How the System Protects Itself

In order for the system to be trusted, it must protect itself from compromise, just as it must protect all other trusted components of the secure system. The system protects itself by means of the access isolation mechanism.

The access isolation mechanism segregates people who log in to the operating system with the Enhanced Security Set installed into two large groups; general, non-privileged users, who log in at level alias name `USER_LOGIN`, and system administrators, who log in at level alias names `SYS_PRIVATE` or `SYS_OPERATOR`. These concepts are discussed in the *"Access Isolation"* sub-section.

Access Isolation

The system contains many files and directories that need protection from unauthorized user access. This protection should prevent both unauthorized disclosure and unauthorized change. However, this protection should allow authorized administrators to access system programs and data files. The system access isolation mechanism provides this protection via the Mandatory Access Control mechanism (MAC).

The MAC mechanism provides the means to control access of different groups of users to data. Access isolation uses MAC mechanisms to control access of system files and commands. The simplest example of this control capability is shown in Figure 11-4 using levels with only two categories. Please note that this is a highly simplified version of access isolation. Its purpose is to demonstrate the concept, not the actual access isolation used in the operating system with the Enhanced Security Set installed. The actual relationship between security levels that provides access isolation is much more complicated.

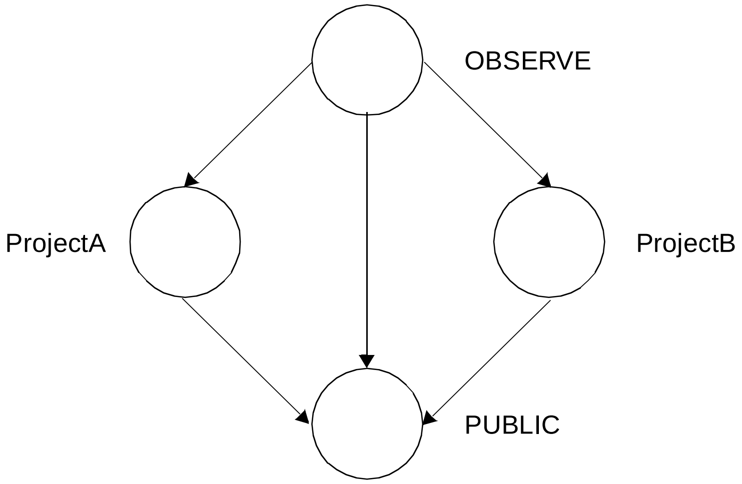


Figure 11-4. Simple Access Isolation Example

In Figure 11-4, there is an arrow from any level to any other level if and only if the first dominates the second. (Any level dominates itself, but the arrows from any level to itself are omitted for simplicity.)

In this example, one level is reserved for ProjectA and one level for ProjectB. Users are assigned either of these two levels. The system reserves for administrative use the OBSERVE and PUBLIC levels.

Note that the two groups of users are completely isolated from each other. ProjectA does not dominate ProjectB and ProjectB does not dominate ProjectA. They can both read PUBLIC data. But they can not write PUBLIC data since the security policy requires MAC levels to be equal for writing. Also, OBSERVE can read but not write data at ProjectA, ProjectB, or PUBLIC.

Of course, this is just a highly simplified example of access isolation. You will not find these terms in the actual system. In fact, there is no OBSERVE level, because an administrator logging in to that level would be able to read all data on the system.

The framework for protecting the system is similar to this simple example, but it contains many more levels with more complex domination relationships among them. These levels are necessary because of the need to separate the files and tasks of administrative roles as well as the need to separate the system from general users. (See Chapter 10, “Trusted Facility Management” in this book.)

Figure 11-5 conceptually depicts the access isolation levels used to protect the system.

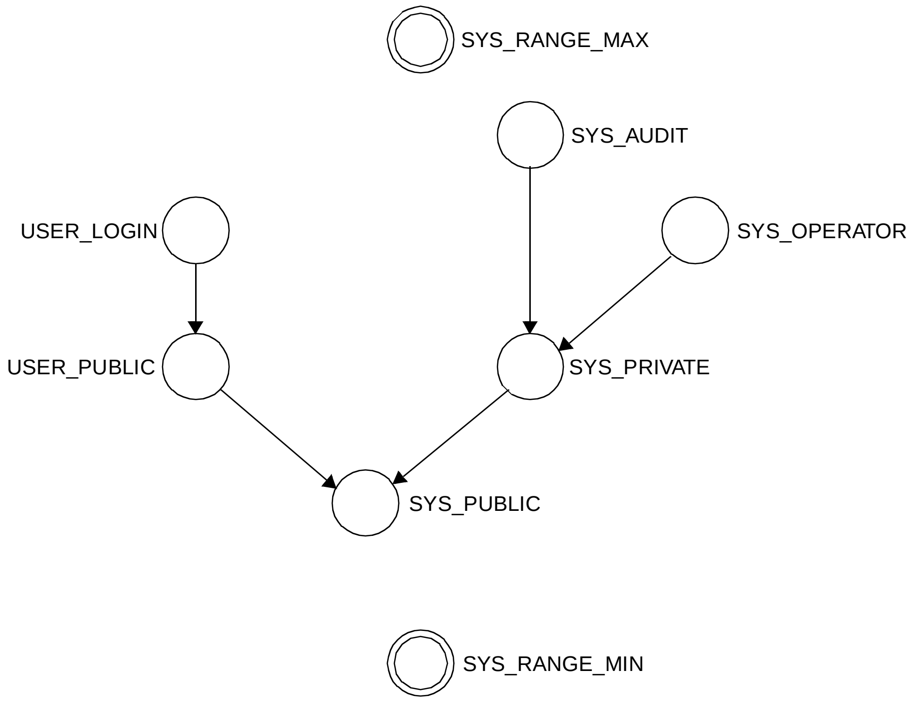


Figure 11-5. Operating System Access Isolation Architecture

Assigning classifications, categories, and alias names to the levels defined by this scheme yields the following mapping:

Alias Name	Security Level
SYS_PUBLIC	system:
SYS_PRIVATE	system:private
SYS_OPERATOR	system:private,operator
SYS_AUDIT	system:private,audit
USER_PUBLIC	user:
USER_LOGIN	user:login
SYS_RANGE_MAX	range_max:ALL
SYS_RANGE_MIN	range_min:

Figure 11-5 can be viewed as a grouping of system levels and user levels whose separation is controlled and enforced through use of the Mandatory Access Control mechanism. (See the *"Mandatory Access Control"* section in this chapter.) Thus, the Mandatory Access Control mechanism as used in the access isolation scheme keeps system administrators and general users separate from each other, maintaining security. General users log in at the level `USER_LOGIN`, and administrators log in at the levels `SYS_PRIVATE`, or `SYS_OPERATOR`.

Levels Provided by Access Isolation

Administrative Levels

In Figure 11-5, the system levels `SYS_AUDIT`, `SYS_OPERATOR`, `SYS_PRIVATE`, and `SYS_PUBLIC` all share the same hierarchical classification **`system`**, with dominance determined by the assignment of different categories.

- The level `SYS_PUBLIC` is dominated by both the other system levels and the user levels. `SYS_PUBLIC` is dominated by `USER_PUBLIC` because the hierarchical classification **`user`** is greater than the hierarchical classification **`system`**. Because neither level has any categories, the hierarchical classification alone determines level dominance. `SYS_PUBLIC` is defined for data files for which read access must be permitted at all levels but for which write access is denied to unprivileged users. An example of such a data file is the publicly readable password file **`/etc/passwd`**. This level is used for executables that must be accessible to all users.
- The level `SYS_PRIVATE` is created by adding the category **`private`** to the classification **`system`**.. Because no user level has the **`private`** category assigned, it is not dominated by any user level. This level is provided for use by the Security Operator, the Site Security Officer, and the Auditor. The

SYS_PRIVATE level is restricted to system access only and may be assigned to both subjects and objects.

Objects whose access is restricted to system levels would be placed here. For example, files such as **/etc/shadow** as well as commands such as **lvlname** are located here. Users executing privileged tasks log in at this level, in order to prevent administrators from executing Trojan Horse programs with privilege.

- The level SYS_OPERATOR is created by adding the category **operator** to SYS_PRIVATE. This level is provided for use by the Operator of the system when performing routine maintenance procedures. Note that this level dominates SYS_PRIVATE, but it does not dominate SYS_AUDIT. This separation allows the operator to perform routine maintenance procedures but prohibits the operator without the macwrite privilege from modifying files at the SYS_PRIVATE level.

  Because the level SYS_OPERATOR does not dominate SYS_AUDIT, the operator without macread and macwrite privileges has no access to the audit trail.

- The level SYS_AUDIT is created by adding the category **audit** to SYS_PRIVATE. This level provides protection for the system audit trail commands and data files. Note that this level is not dominated by any system or user level. This protects the audit trail from unauthorized access by both system and user levels. See the *Audit Trail Administration* guide for details about auditing.

General User Levels

The levels USER_LOGIN and USER_PUBLIC, as the names imply, are provided for assignment to non-privileged subjects and objects.

- The level USER_LOGIN is provided as the default login level for all users. This is the only user login level provided in the delivered system.

  You should create appropriate levels for user logins at your site. These levels should dominate USER_PUBLIC and should not dominate any of the administrative levels.

- The level USER_PUBLIC is the user-level equivalent of the level SYS_PUBLIC. Publicly accessible, non-trusted objects would be placed at this level. Examples of files which would exist at USER_PUBLIC are executables such as **vi** and user-defined data bases. Note that sites assigning additional user login levels should ensure that they dominate USER_PUBLIC.

Maximum and Minimum Levels

The levels SYS_RANGE_MIN and SYS_RANGE_MAX together provide the bounding range for all levels on the system. These levels are not assigned to subjects or objects. As such, it is guaranteed that they will always provide lower and upper bounds for all labeled system objects.

Use of these levels provides the capability to support unbounded devices. An unbounded device is a device which is defined to be accessible at any level, in essence a device with no range restrictions. Examples of such a device would be the tape device used for archiving all the files on a system or a file system that is used by both administrative and non-administrative personnel.

Note that the principle that all levels may access an unbounded device is a property of the levels assigned to the device range and is not a property of the device itself. All mandatory access checks performed by a regular device are performed with an unbounded device.

Because the ranges assigned to an unbounded device always dominate the highest system and user levels and are dominated by the lowest system and user levels, the mandatory access checks always pass. Thus, a simple, straightforward method of providing unbounded devices is by defining two levels: one that dominates all levels on the system (SYS_RANGE_MAX) and one that is dominated by all levels on the system (SYS_RANGE_MIN).

Thus, the MAC mechanism provides a simple way to control access and protect the system from unauthorized disclosure and unauthorized change. Together with the privilege mechanism, access isolation forms a robust architecture for the system.

It also provides a way to separate general users from administrators while allowing users to access those shared operating system commands that they must use day-to-day.

Moving Files from One Level to Another

If you add files to the configured system, you will want to be aware of the risks and benefits of such a change. In general, moving a file from SYS_PUBLIC to SYS_PRIVATE, although technically a modification to the system, is a relatively safe action. Moving a file in this manner would prevent general users from accessing the file.

However, a move in the opposite direction, from SYS_PRIVATE to SYS_PUBLIC would be unsafe and is strongly discouraged. Such a move would mean that all users on the system could read the file, subject only to DAC restrictions. The MAC access check would always pass. Effectively, Mandatory Access Control for the file has been bypassed for read access.

For more information about adding trusted software to the system, see “Guidelines for Writing Trusted Software” in the *Compilation Systems Manual*.

Implications of Security for Administrative Roles

The security mechanisms supplied by the system will be effective only if the administrative personnel perform their jobs as defined by the security policy. The administrative responsibilities are not assigned to a single administrator. Rather, the responsibilities and tasks are divided into groups and assigned to roles. Each role should be assigned to a different person. By separating administrative functions and requiring different people to be responsible for different areas, the security policy enhances system security.

The system administrators responsible for the secure operation of the operating system with the Enhanced Security Set installed are the Auditor (AUD), the Operator (OP), the Security Operator (SOP), and the System Security Officer (SSO).

For a more complete discussion of these roles and the responsibilities associated with them, see Chapter 10, “Trusted Facility Management” in this book.

How the Components of the System Work Together

The previous sections explained what is meant by the term security and the components of the system were explained individually. Although some of the information in this section is similar to that in the previous section, the emphasis in this section is on details of how the components of the System work together to provide security by providing accountability. For example, this section both discusses how security levels of objects are established and provides details about the access checking algorithms.

The system must be able to account for any security-relevant actions taken by itself on behalf of users on the system. To ensure accountability, the system must perform stringent authentication and identification. The ability to audit security-relevant system events, through the installation of the Auditing Set, is also important. For details on these mechanisms, refer to this section.

In general, the steps in assuring security are as follows:

1. You identify and verify yourself to the system via the Identification and Authentication mechanisms, **`login`**, and the password check.
2. You establish a process on the operating system. The process uses your identity and the security level that has been assigned to you to make access control decisions via the Discretionary and Mandatory Access Control Mechanisms.
3. The access isolation mechanism, which is built on Mandatory Access Controls, keeps general users from accessing sensitive system files and directories to which system administrators must have access.
4. Privileges provide users the ability to override system restrictions.
5. The Audit mechanism, if installed, keeps track of sensitive operations and who performs them.

As the items in this list demonstrate, the system establishes a chain of control to ensure security. Anyone using the operating system with the Enhanced Security Set installed must access the login and password verification mechanisms in order to create a process on the system. Then, that process and all the processes it spawns must pass MAC and DAC checks before it is allowed to access any object.

Any task that requires privileges to override MAC and DAC checks must be done via the Privilege mechanism, which is under the control of the system.

Finally, sensitive operations can be monitored by the Audit mechanism. Thus, the chain of secure accountability is maintained.

Identification and Authentication (I & A)

Identification mechanisms are employed by a user to prove who the user is and authentication mechanisms are used by the system to verify the user's claim of identity. A user gains access to the system by entering a login name that is verified to the system via a password. Compromise of this password could allow another person access to the system and to all resources authorized for the password's real owner. A reliable I & A is essential for the proper functioning of the access control and auditing components of the system.

The algorithm for I & A checks the expiration date of a *uid* when a user is added to the system as well as when a user logs in. In addition, the system administrator is strongly encouraged to assign unique user logins, rather than permit the use of “group” logins, or the sharing of a login by more than one user.

Security Policy and the Access Control Mechanisms that Implement It

The system's security policy consists of the access rules that the system follows, plus the exceptions to those rules permitted to privileged processes in the system. With the Enhanced Security Set installed, the security policy conforms to the National Computer Security Center's criteria for a B2 level of trust. The access rules dictate the limits on non-privileged users of the system; the system's access mechanisms implement the various rules.

The DAC and MAC access mechanisms in the operating system control the access of subjects to objects, determining whether or not a subject may access a given object, and in what manner.

When performing access checks, MAC checks are performed first, followed by DAC checks.

Mandatory Access Control (MAC)

Mandatory Access Control is enforced by the system. A non-privileged user cannot affect or bypass this control.

MAC Concepts and Definitions

MAC on the operating system with the Enhanced Security Set installed is implemented by associating a label with each object and subject in the system, either explicitly or implicitly. Labels contain a sensitivity level, or just level, indicating the sensitivity of the data in the object, or the clearance of the subject, with which it is associated. The level is a tuple of one hierarchical component, or classification, and zero or more associated non-hierarchical components, or categories.

An ASCII name is associated with each classification and each category; it is defined by the system administrator for easier use and recognition by administrators and general users. Classification names may be something like “Project Alpha” or “Project Sigma.”

Categories, analogous to need-to-know areas, might have names such as “sales” or “salaries.”

The Enhanced Security Set support 256 classifications and 1024 categories. For a more complete description of how Mandatory Access Controls work, see the other *“Mandatory Access Control”* section presented earlier in this chapter. Also refer to, the “Managing File Systems Securely” chapter of the *User's Guide* and Chapter 17, “Administering Mandatory Access Control and Multilevel Directories” in this book.

Given these basics, MAC controls access to objects on the basis of the following rules and access checking algorithms.

MAC Access Rules

The objects under MAC control are:

- file system objects: regular files, device special files, symbolic links, regular and multilevel directories and FIFOs
- unnamed pipes
- IPC objects (shared memory, semaphores, message queues)
- processes

Several of the object types listed above, including multilevel directories and device special files, require special handling for MAC. They are discussed below in subsequent subsections.

The general rules for all objects are based on the concept of dominance.

- In order to read an object, the subject's level must dominate the level of the object. For one level to dominate another, two conditions must be met:
  - The classification of the first must be at the same or a higher hierarchical level.
  - The set of categories of the first must be a superset of the set of categories of the other.

NOTE

Level dominance is more qualified in the case of IPC objects and pipes. For IPC objects and pipes (named and unnamed), the level of the subject must equal that of the object before read is allowed.

- In order to write an object, the subject's level must be equal to that of the object.
- In order to search or execute an object, in general, the subject's clearance need only dominate that of the object; that is, this is implemented following the same rules as “read” access. Search or execute access does not apply to pipes or IPC objects.

- In the case of accessing objects contained in multilevel directories, access is contingent on the multilevel mode. For virtual mode, the subject's level must equal that of the object for all forms of access. For real mode, the typical file access rules apply.

Establishing the Levels of Objects and Subjects

There are two situations in which the level of an object may be set:

- Creation of an object. Creation of an object is a special case of write. The level of a newly-created object is set equal to that of the process that creates it.
- Administrative change. An administrator with appropriate privileges can change the level of an existing object. The administrator must take care that, when changing the level of an object, information is not unwittingly being downgraded.

The level of a subject (that is, process) is generally set by the calling process. The two general cases are:

- At login, the user's process is created on behalf of the user by the **`login`** system process. The user's level is determined by the **`login`** process: it is either explicitly specified to **`login`** by the user as part of the **`login`** command, or, if the user chooses not to specify a level, **`login`** takes it from a system database containing default levels for all users of the system. After this and all other user parameters are established, **`login`** becomes a process for the user with the correct level.

  If either the specified or default level is outside the system's current login level range, the login fails. (Checks against login level range are made only at login time.) If the specified or default level is not a level that is authorized for that user, the login fails.
- A child process (created by `fork` and `exec` sequence) or a new process image (created by `exec`) of an existing process takes the level of the calling process. The only exception is in the case of an appropriately privileged process, which may create a process at a level different from its own (as in **`login`**).

An appropriately privileged process may change its own level. Only processes that are part of the system are trusted to do this.

File System Ranges

Each file system image on disk can potentially contain data at different levels. When a file system is mounted, it is mounted on a device special file with an associated level range. Once mounted, the system ensures that the integrity of the levels of the data on the file system is maintained; that is, writes to the file system that are outside the level range of the device will fail. Read access to data whose level is dominated by the low end of the range is allowed, but write access is not allowed. Access to data whose level is not dominated by the high end of the range is not permitted.

Default level ranges for file systems are maintained in the administrative file **`/etc/vfstab.`** The level range can be set at file system mount time by use of options to the

`mount` command. The actual level ranges for file systems that are mounted are maintained in the administrative file **`/etc/mnttab.`**

CAUTION

> The mount point of the file system is the floor security level of that file system. If it is necessary to change the floor security level, it is necessary to unmount the file system, change the security level of the mount point, and then remount.

Multilevel Directories

The directory model in the operating system with the Enhanced Security Set installed provides additional security. MAC requirements prohibit unprivileged users from creating files at different sensitivity levels within the same directory. That is, unprivileged users may only create files at the same level as the level of the parent directory (and at their current login level). However, certain directories are used as “public” directories and require that an unprivileged user have write access to them regardless of the user's level. Multilevel directories solve this problem.

To a general user, a multilevel directory looks and acts like a directory whose contents are all at that user's level. To a second user at a different level, that same multilevel directory would appear to contain a different set of files. This is because each user sees an “effective” directory consisting only of objects at the user's level. Figure 11-6 demonstrates how processes at different levels “see” directories and files at different levels on a multilevel directory.

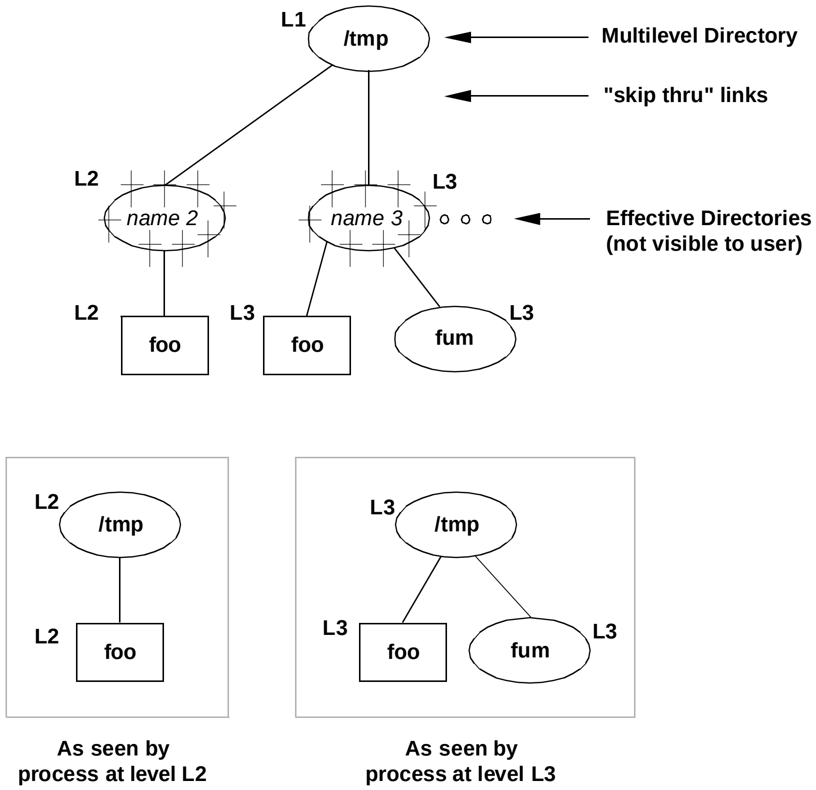


Figure 11-6. Example of a Multilevel Directory

On a secure system (that is, a system running the operating system with the Enhanced Security Set installed) certain directories, such as **`/tmp`** and **`/usr/tmp`**, must be multilevel directories. Other directories may be created as multilevel directories at administrative discretion.

A general user cannot create a multilevel directory; that is a privileged operation.

Secure Device Handling

The two major goals in adding MAC to device handling for the operating system with the Enhanced Security Set installed are to:

- minimize driver changes
- retain, for unprivileged processes, support for the UNIX device paradigm that operations on devices are the same as those on files

There are several layers of protection for devices. The first is the classification of devices as either public or private. This classification has nothing to do with the DAC or MAC controls associated with the device; it is an internal state kept in a table in the kernel. As delivered, all devices on the system are **`private`** unless otherwise specified by device drivers. This means only administrators with appropriate privileges may access them.

At system startup time, the administrator has the option of re-classifying devices to be **`public`**, or potentially accessible by a non-privileged user, subject to MAC and DAC controls. Devices typically designated as public include terminals, tape drives, and floppy drives.

MAC provides another layer of protection for devices. As with regular files, every device special file has a label associated with it. In addition, each device (represented by one or more device special files) also has a minimum and a maximum level associated with it. This device level range determines the range in which the device can operate.

Devices are categorized as either single-level or multi-level:

- Multi-level devices are devices that may contain data at many levels; that is, they contain data with embedded MAC information. These devices can be opened only by processes with the appropriate privilege. Such processes include the kernel and system processes and administrators with special privilege. Examples of devices that fall into this category are disks and memory devices.
- Single-level devices are those devices that process only one level of data at a time. Such devices include terminals, and tape drives and floppy drives when used in the appropriate mode. For a device to be used as public resource, it must be a single-level device. An administrator with appropriate privileges may use these devices as multi-level devices; as, for example, when creating a backup tape for the system.

By default, every device on the system is a system private resource.To permit access to a device by a general user requires that a trusted program grant unprivileged access to that device.

Normally a user will access a terminal device at login time. The user attempts to log in on a particular terminal at a particular level. If that level does not fall within the device level range for the terminal being used, the login fails. If the login succeeds, the device-specific label is set to the login level specified by the user.

To use a tape or floppy device, or to get access to a terminal device other than the one used for logging in, the user must request the administrator to allocate the device. The administrator attempts to allocate the device to the user at a particular level. If that level does not lie within the target device's level range, the request fails. If it does, the device is allocated to the user. The user becomes the device owner, the DAC file modes are set to 600, the device level is set as specified in the allocation command, and the administrator informs the user of the success of the operation. The user can then use the device at will, assuming that the user's current level is equal to the allocated level.

There are a few devices that fit neither of the two categories and require special handling. These include **`/dev/null,`** the universal bit bucket; **`/dev/zero,`** the universal source; and **`/dev/tty,`** the general terminal device that provides access to specific terminal devices. Since data does not flow through any of these devices, all users may access these devices at any time.

The operating system with the Enhanced Security Set installed supports the conceptual classifications of devices as static and dynamic. The class of dynamic devices is provided for compatibility with systems, such as MLS, that provide support for multiple labels on windowing terminals.

Upon installation of the Enhanced Security Set, all devices are initially static. A device may be designated as dynamic at the discretion of an appropriately privileged administrator. The level of a static device in the public state is not permitted to change while it is open or mapped; the level of a dynamic device may change at any time.

This ability to change the level of an open device does not violate the security policy of the system because of the way MAC checks are made: For static devices, checks are made only at the time the device is opened or when an `ioctl` is issued; for dynamic devices, checks are made not only when the device is opened, but also for subsequent operations to the device (`read`, `write,` `ioctl`, and mmap).

Discretionary Access Control (DAC)

Discretionary Access Control is a means of controlling access to objects that is exercised at the discretion of its owner. For each object owned by a user, the user can designate which other users on the system have what types of access.

The set of regular UNIX permission bits associated with each file is a DAC mechanism. The the operating system Enhanced Security Set offers an additional mechanism: Access Control Lists (ACLs). Previous file permission modes are retained for compatibility. Users already familiar with UNIX file permissions will find that this mechanism still works as expected. See *"Discretionary Access Control"* earlier in this chapter for a description of how these mechanisms work.

Privilege

The system architecture is required to enforce least privilege among system modules when the Enhanced Security Set is installed. Least privilege is the principle that requires that each subject in a system be granted the most restrictive sets of privileges needed for the performance of authorized tasks. Figure 11-7 demonstrates how privileges are inherited by processes when using the Least Privilege policy module (LPM).

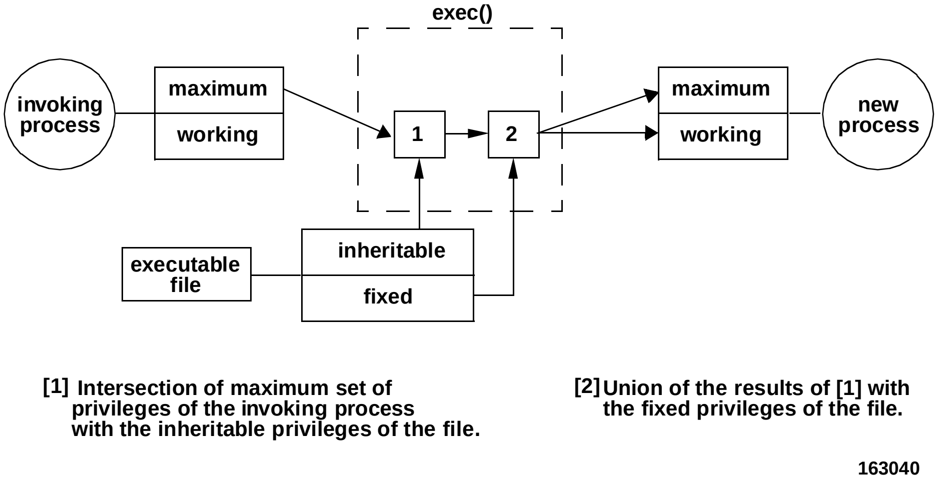

exec()
invoking process
maximum
working
1
2
maximum
working
new process
executable file
inheritable
fixed
[1] Intersection of maximum set of privileges of the invoking process with the inheritable privileges of the file.
[2] Union of the results of [1] with the fixed privileges of the file.
163040

Figure 11-7. Privilege Inheritance Mechanism (With LPM Module)

The figure above illustrates the way in which privileges are passed from one process to another. Privileges in the calling process's maximum set are given to the new process only if the privilege is in the inheritable set of the new process (that is, its executable file). In this way, the inheritable privileges on an executable file serve to limit the privileges in the maximum set of the calling process that are available to the new process. This inheritance mechanism prevents unneeded privileges from being passed to the new process. Inheritable privileges are not supported when using the Super User (SUM) privilege policy module. The maximum set of process privileges are always considered as being inheritable when using the SUM module. Fixed privileges are always given to the new process.

The maximum and working sets of privileges of a new process are identical. The working set represents those privileges currently being used by the process and can be adjusted to enforce the principle of least privilege. The working set then becomes a subset of the maximum set.

In an operating system with the Enhanced Security Set installed, the key to the implementation of administrative least privilege is the diversification of what was superuser or “root” in previous releases. The authority to perform distinct administrative functions with different levels of privilege has been divided among distinct administrative functions and roles.

Trusted Facility Management (TFM)

The security policy requires that the system have two kinds of trusted user classes: administrator and operator. When used with privileges and proper system configuration, the TFM feature provides all necessary mechanisms that define administrative roles, as required by the security policy.

The responsibility for performing various system maintenance, configuration, and administration tasks is divided over several roles. The intent of this division of responsibility is to assure that administrative personnel do only those tasks for which they are trained and responsible. Moreover, the principle of least privilege is maintained by giving each role only those privileges needed to perform the necessary and appropriate tasks.

For further discussion of roles, see Chapte r10, “Trusted Facility Management” in this book.

Audit

The operating system provides an audit mechanism capable of recording and reporting all security-related events that occur on the system. The security and design goals of the operating system audit mechanism are to:

- Provide a degree of security auditing that satisfies the B2 criteria, both in terms of events and information that are audited, and in terms of the protection afforded the auditing mechanism;
- Provide a flexible system that allows the administrator to trade off the benefits of auditing against the drawback of performance loss incurred by increased auditing, as appropriate to the needs of the specific site and installation.

All security-related events that occur on the system can be audited, including those events identified as being associated with covert channels. A certain subset of events deemed critical to the integrity of the audit log (and other events deemed necessary to maintain the traceability of these events) is always audited whenever auditing is enabled. These events are called `fixed` events. Other events are auditable at the discretion of the system administrator; these are called `selectable` events.

The audit system is always present when the operating system Enhanced Security Set are installed. The administrator sets selectable events to be audited with the **`auditset`** command. The set of selectable events so chosen will begin being audited the next time auditing is enabled. The administrator enables or disables auditing with the commands **`auditon`** and **`auditoff`**.

When auditing is enabled with **`auditon,`** fixed events plus any selectable events chosen with **`auditset`** are audited. **`auditset`** may also be used after auditing is enabled to specify additional events to be audited or to de-select events that no longer require auditing. User-specific audit masks may be designated for each user by using the **`useradd`** or **`usermod`** commands. These masks are permanent. Whenever auditing is enabled and the user is logged on, events specified in these masks will be audited. To temporarily audit additional events for a user, the **`auditset`** command can be used to select the desired additional events. The events selected with **`auditset`** apply only to the user's current login session; if the specified user is not logged on, these events will not be recorded.

For each auditable event, audit records are maintained in an audit log file that is accessible only by the system and appropriately privileged administrators. Each audit record contains a time stamp, the user identity, the object name, the level of the process (subject) causing the event, the privileges used, an identification of the type of event, and an indication of the success or failure of the event, as well as other information specific to the event type.

The audit log is protected by taking appropriate advantage of the various mechanisms offered on the operating system with the Enhanced Security Set installed (including DAC, MAC, and the access isolation scheme) and by the fact that privilege is required to manipulate the audit mechanism. See the *Audit Trail Administration* guide for more information.

System Startup and Security

This section describes a typical sequence of events in a representative UNIX installation. An administrator may tailor configuration and startup files to change this sequence to suit local needs.

NOTE

Note that a system administrator must define user login security levels after the Enhanced Security Set are installed and the system is rebooted. Refer to Chapter 14, “User Accounts and Group Management” in this book for details.

Also note that, after the system is installed and configured, a system administrator need do nothing special to get the system to the secure operating state; the default system state, as the system is delivered, is the secure, multi-user state.

System Startup

When a system is booted, a computer-dependent boot program reads in and starts the kernel, **`/stand/unix`**. The kernel does some initialization, creates a few processes that do housekeeping and basic operations, and then executes process 0, which helps schedule processes. Process 0, like all the processes started from the kernel at startup time, runs as long as the system is up. The code for process 0 is entirely in the kernel; process 0 has no user context and never runs in user mode.

Several processes are started by the kernel. These include **`init`**, as process 1, and `sysproc`, as the system daemon process. This includes threads for each system daemon: the page out daemon, the file system flush daemon, the asynchronous I/O daemon, and the free-buffer-pool daemon.

Process 1, **`init`**, is responsible for starting all user processes in the system. There are two pieces of **`init`**. The kernel **`init`** routine executes the program **`/sbin/init,`** which is a privileged executable file on secondary storage that runs in user space. Except for the root directory, **`/sbin/init`** is the only file known inside the kernel. The **`/sbin/init`** command is described on the **`init(1M)`** online manual page.

The file **`/etc/inittab`** controls **`/sbin/init. inittab`** specifies which processes are to be run in which system state. System states provide a way of organizing the standard operations of the system.

The UNIX system typically has at least the following states: single-user state (state s or S), multi-user state (state 2), networking state (state 3), and shutdown state (state 0). Other states include administrative state (state 1), and reboot state (state 6).

Single-user state is used by a system administrator to perform tasks that preclude ordinary user processes, such as configuration changes or file system repair. Typically, in single-user state only the console accepts logins, file systems are not mounted, and services (such as LP) are not started.

Multi-user state (state 2) is the conventional state for multi-user interactive service. In multi-user state, login and other services are started, and user file systems are mounted.

A system goes into shutdown state (state 0) when an administrator brings it down (although this can also happen as a result of an error in the auditing system, if it is configured to do so).

The reboot state (state 6) is used to halt and reboot the operating system. It is also used to configure a new bootable operating system if the system configuration does not match the currently executing operating system's configuration. (The two systems might fail to match, for example, if a new options card has been added or a new driver has been added.)

The **`init`** program controls state transitions and determines which processes run in a particular state based on the contents of the **`inittab`** file, described on the **`shutdown(1M)`** and **`inittab(4)`** online manual pages.

Initialization scripts usually do routine maintenance and housekeeping when a system makes state transitions. See **`rc0(1M),`** and **`rc2(1M)`** online manual pages.

See Chapter 3, “Booting and System States” in this book for a more detailed description of system states and state changes.

Terminal Port Monitor

The terminal port monitor, **`ttymon`**, supplies the standard login service. See **`ttymon(1M)`** online manual page **`ttymon`** is typically specified in **`inittab`** for all states that are to accept logins. **`ttymon`** replaces the **`getty(1M)`** function of earlier releases.

`ttymon` reads port monitor administrative files to get information about communication lines. It initializes the lines and then listens for someone to try to use them. It is **`ttymon`** that implements the trusted communication path between terminals and the system, and starts the service that is configured for that line. For normal interactive use, **`ttymon`** detects the SAK and starts the **`login`** program.

login

The **`login(1)`** program identifies and authenticates users. It asks for a login name and password and validates these and any other inputs (such as a sensitivity level) given at the login prompt.

Login information is listed in the file **/etc/passwd**. See the **passwd(4)** online manual page. Note that group information is given in the file **/etc/group**, but this information is not relevant to security in terms of identification and authentication.

The **login** command checks the password entered against the encrypted passwords in the file **/etc/shadow**. See **shadow(4)** online manual page (The **passwd** file is usually readable by everyone; the **shadow** password file is readable only by privileged processes.)

If a valid login name and its password is entered, **login** executes the program specified in the **passwd** file. Typically this is the user's shell, or command interpreter, but any command may be executed on login. The administrator controls what gets executed.

This program's process is given the user's default Mandatory Access Control level, unless the user requested another valid level at the login prompt. The requested level must be a level assigned to the user. This level is propagated to all processes spawned and all objects created during the user's terminal session. (The only way to change the user's current level is to log out of the system and log in again at a different level. Note that the level of a process can be altered only through the use of privilege via a well defined trusted user-level interface.)

The simplest way for an administrator to limit what a user can do is to give that user a restricted program in the **/etc/passwd** file. Such a program might, for example, prevent the user from starting other programs or creating files. The program might also be a special-purpose application rather than a general command interpreter.

The administrator can also limit users' access in the assignment of login levels.

The Operating System Shell

A shell is a command interpreter that forms the basic interactive interface to the system. The shell's basic functions are to run executable files and to provide programming features such as variables, functions, I/O, and flow control.

The shell in use is called the Bourne shell. Two copies of this shell exist on the secure system. One copy resides in the file **/usr/bin/sh** and is intended for unprivileged interactive and interpretive use. Another version of the Bourne shell resides in the file **/sbin/sh** and is intended for the execution of trusted administrative shell scripts.

/sbin/sh and **/usr/bin/sh** both are at the SYS_PUBLIC level to prevent unauthorized modification of these files. **/sbin/sh** has only inheritable (no fixed) privileges. Thus, even an administrator cannot obtain a privileged version of this shell. The shell runs with no privileges of its own and its ability to perform sensitive operations is limited by the privilege(s) of the invoking process.

Note that no shell has fixed privileges or performs sensitive operations. The functionality that would be provided by such an administrative shell is provided by the **tfadmin** command, which is used by administrators to gain required privileges to perform sensitive operations. It is **tfadmin** that spawns a child process to do the work of a command (such as **mount**) that the user is attempting to execute. In this regard, **tfadmin** acts as a command processor rather than a command interpreter.

In order for a shell script to propagate privileges whether they are acquired by way of **tfadmin** or **filepriv**, the script file must begin with a line of the form:

```
#! pathname [arg]
```

where pathname is the path of the interpreter (usually a shell), and arg is an optional argument.

/usr/bin/sh has no file privileges (fixed or inheritable) associated with it.

The Bourne shell is documented on the **sh(1)** online manual page. Other shells (such as csh, ksh, and jsh) exist on the system.

So far as the kernel is concerned, a shell is just like any other process in user space. A shell does not enforce access policy or have any special privileges. Like any other user process, it runs with the privileges of the process which invokes it. That is, the system enforces access restrictions on the shell as it does on any process.

The shell maintains an environment made up of a set of variables and values for each. These variables may be conventional names used by the shell or other programs, system-defined variables, or user-defined variables. For example, two of the most important variables used by the shell are PATH, a list of directories that the shell searches for executable programs, and HOME, the name of the user's home directory. The **env** command lists the user's environment.

When it starts, the shell reads a system startup file, **/etc/profile**, which gives the user an initial environment supplied by the system administrator. Then it reads a personal startup file, **$HOME/.profile**, which users may tailor to their own preferences. See **profile(4)** online manual page.

The typical operation of the shell after it sets up the user's environment is to enter an infinite loop, reading user input and executing it. The user may leave the shell by entering the shell **exit** command. If a user exits the top-level shell (the shell spawned when the user logged in), the login session is ended and the user must go through the login procedure again to get a new shell.

Summary

The diagram of Figure 11-8 summarizes a simple startup sequence for a few standard UNIX system processes. For the sake of brevity, the events shown have been simplified; in actuality, **init** starts more processes and services than those shown here.

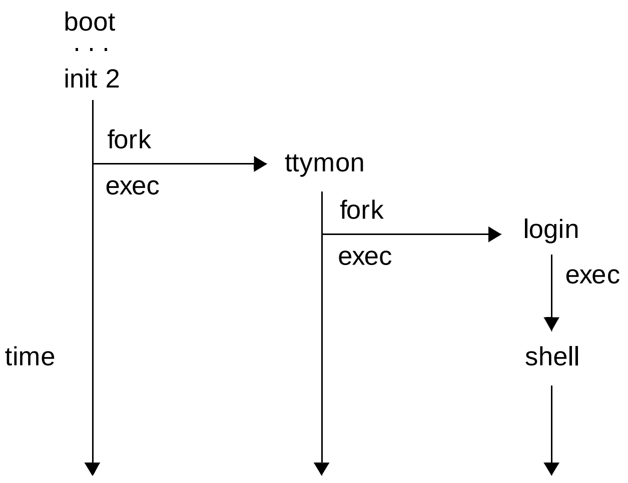


Figure 11-8. Simple Startup Sequence

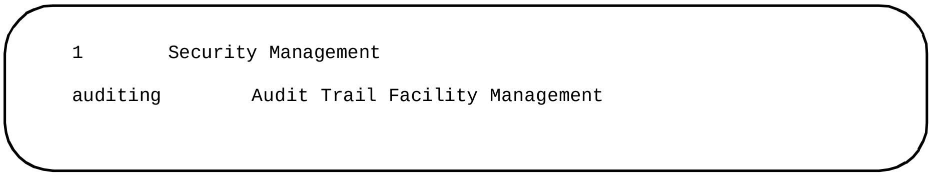
After the system boots (that is, after the bootable operating system is loaded and executed), it starts the **init** process. A typical interactive system eventually goes to system state 2, multi-user state, in which **init** starts login service to the console by doing a fork, followed by an exec of the **ttymon** program, which monitors terminal communication lines. Other system ports are monitored by the **sac** program, which is also started at this time (this program is left out of the diagram for simplicity). The **sac** program starts up **ttymon** for each of the ports it monitors.

At this point init, **sac**, and **ttymon** processes are all running on the system. When someone requests service on a port, **ttymon** forks and executes the **login** program on that line. If the login is successful, **login** executes the user's shell.

init runs as long as the system is up. **ttymon** runs as long as the system is in a system state that includes it, managing terminal lines, starting user sessions, and cleaning them up when they finish.

Security and the sysadm Interface

To help you do administrative work easily, the UNIX system provides a menu interface which can be accessed through the **sysadm** command. There is no single administration menu dedicated to system security. However, some security related tasks can be done through the **users** (short for User and Group Management) and systemsetup (or System Setup) menus. The security (short for Security Management) and users menus can help you establish and maintain security on your system.To access the security menu, type sysadm security; the menu will appear on your screen as follows:



```
1           Security Management

auditing          Audit Trail Facility Management
```

Screen 11-1. Main Menu for Security

NOTE

> The selection for auditing will only appear if you have installed the Auditing Set.

By selecting an item from this (or any) `sysadm` menu, you invoke a series of screens in which further menus or prompts for information are displayed. By making more selections and/or entering responses to the prompts, you can specify the task you want done and the way in which you want it done.

If you prefer not to use the menus, you can do the same tasks by issuing shell commands. See the relevant chapters in this book for more information. Keep in mind that some tasks cannot be done through the menus.

12
Installing Software on an Enhanced Security System

12
Installing Software on an Enhanced Security System

Introduction

This chapter gives administrative users specific guidelines on how to install, configure, and run the operating system with the Auditing and Enhanced Security Utilities installed. It illustrates the intended use of the features available to administrators.

While a carefully planned and properly implemented installation is the first step in assuring the security of your system, administrators must also recognize the vulnerabilities that can arise due to the misuse or careless use of administrative authority, both at installation time and during the normal operation of the system. It is the intention of this chapter to provide you with an awareness of the proper security administration of the system and minimize the chance that you or another administrator will inadvertently compromise system security.

This chapter assumes that you have some knowledge of the administration of computer systems in general, and of trusted systems and computer security in particular. To help the reader identify key concepts and terms, a *"Glossary"* is provided in this book that contains definitions of general administrative and computer terms, as well as definitions of security-specific terms.

This chapter, and the readings cited in the *"Before You Begin"* section of this chapter, will give you the information you need to make effective use of the system's features, control access to administrative functions and data files, and protect user and system security.

While these readings are not intended as a primer on trusted systems or computer security, they do provide detailed and accurate information on how to configure, install, and operate the system in a secure manner.

The following section provides specific instructions and guidelines for installing the system and configuring the installed system for secure operation. The section after provides guidelines for installing add-on packages and third party software onto an Enhanced Security System.

Setting Up the Enhanced Security System

This section provides detailed instructions on how to set up an Enhanced Security System.

Before You Begin

There are several items that must be completed in advance to help ensure successful installation and operation of your system. It is recommended that you install Enhanced Security at the time you initially install the OS on your computer. If this is not the case, it is recommended that you back up any user data you wish to save, and then re-install the entire system.

Who Should Install the System?

The process of installing the system software and then preparing the system for general use should be performed by the Trusted Systems Programmer (TSP); this individual is responsible for all computer operations that occur outside the normal, multi-user, secure operating state of the system.

During the setup procedure, the TSP has unrestricted access to the system and for this reason should be a competent computer professional who has a solid knowledge of computer systems in general and computer security considerations in particular.

The installation and configuration of the system is not an operator-level task; it will influence directly both the security of your system and your user community's ability to make the best use of its facilities.

Prerequisite Reading

Read this entire chapter, and all the chapters and sections in the list given below, before beginning the process of setting up your computer. Failure to do so may result in an improper installation, or one that does not meet the needs of your user community. These sections will help you to determine and document the setup of your machine, including:

- security levels, classifications, and categories
- user logins and login parameters
- password-related parameters
- auditable events and auditing-related defaults
- administrative access to the system

Read the following sections of the indicated manuals completely, and this entire section, before beginning the process of setting up your computer. (Unless otherwise indicated, the chapters and sections cited appear in this book.)

- Read the following chapters of the *User's Guide:*
  - *“What Is the Operating System?“*
  - *“Basics for Operating System Users”*
  - *“Using the File System”*

  The above chapters provide important information on the design of the system and its security features.

- From Chapter 11, “Introduction to Security” read:
  - *“Security and Your System“*
  - *“Implications of Security for Administrative Roles”*

  These sections tell you about the design of the secure system and its intended use, and the expected activities of system administrators.

- Read Chapter 17, *“Administering Mandatory Access Control and Multilevel Directories”* in this book, with special attention to the subsections
  - *“Classifications, Categories, Levels, and Aliases”*
  - *“Predefined Classifications, Categories, Levels, and Aliases”*
  - *“Restrictions on Classifications, Categories, Levels, and Aliases”*
  - *“Adding Classification and Category Names”*
  - *“Defining a New Security Level”*
  - *“Adding an Alias Name”*

  These sections provide you with the information you need to define the security levels for your system.

- Read the following sections of Chapter 14, “User Account and Group Management” of this book:
  - *“Overview of User and Group Management”*
  - *“Suggestions for User and Group Management”*
  - *“Controlling Access to the System and Data”*

  These sections provide the information you need to define the user logins and groups for your system.

- Read the *Audit Trail Administration Guide*, with special attention to the “Enabling Auditing” and “Auditable Events” chapters.

Other Preparatory Items

Several other items should be checked before beginning installation:

- The type of file system you choose is important. If you wish to support Access Control Lists (ACLs), you must choose the **`sfs`** file system type when installing. If you wish to support Mandatory Access Control (MAC), you must choose the **`sfs`** type.
- Consider your disk partitioning. The auditing feature can require very large amounts of hard disk space if you will have a great deal of activity on your system and you are auditing all or most events. You may wish to allocate more than the default amount of space to your **`/var`** partition.

- If you have user data present, you will have to back up the data before installation, and then restore it once installation is complete.
- Compatibility between the Enhanced Security Utilities Package and the X11-based graphics packages is not provided.
- Installation of any of the following packages negates the B2 certifiability of the Enhanced Security Utilities Package, and may give undefined results.

Motif Run-TimeMotif Development	
Applications and Demos	XWIN GWS Fonts
Windowing Korn Shell	TypeScalar Fonts
Desktop Manager	Graphics Utilities
Networked Graphics	XWIN GWS Development
MooLIT Development	Desktop Manager Development
ATM Basic Fonts	Graphical Applications Set

Conversely, all functionality in the packages listed above may not function as expected when run on a system with the Enhanced Security Utilities Package installed.

Isolating /tmp from Administrative Files

This directory is used for temporary files, both by the system and by user applications. To ensure that these temporary files are properly protected, you must override the default partitioning, and either allow **`/tmp`** to be created as a directory in the root file system, or create **`/tmp`** as a separate file system **`sfs`** (for ACLs and MAC labels). To allow applications that assume the existence and general accessibility of **`/tmp`** to run correctly on the secure system, the file system on which this directory resides must have a level range of `SYS_PUBLIC` to `SYS_RANGE_MAX`.

The root file system is installed with a level range of SYS_PUBLIC to SYS_RANGE_MAX. This, in effect, imposes the level range requirements of **`/tmp`** on the entire root file system.

You could repartition your hard disk so that **`/tmp`** resides on its own file system, and has a level range of `SYS_PUBLIC` to `SYS_RANGE_MAX`.

Doing this allows you to place a more restrictive level range on the root file system of `SYS_AUDIT` to `SYS_PUBLIC`.

You can create a separate partition for **`/tmp`** either during installation or anytime thereafter. Because it is much more difficult to do this after installation, we recommend you do so during installation. For instructions, see the section *"Viewing/Changing Disk Partition"* in the *Release Notes* for this release.

If you do choose to repartition your hard disk after installation, you will need to perform the full setup procedure (as detailed in this chapter) again, performing a full backup of your system before beginning, reinstalling the system completely, and reloading any files peculiar to your site from backup.

Installation Procedure

You should follow the installation procedure as documented in the *Release Notes* to install the Operating System packages you wish onto your system. You may have other application packages that you wish to install that are not in the Operating System Package set. These should be installed after you follow the instructions in the following section, *"Configuring the System for Operation"*, also see Chapter 7, "Installing Add-on Software" for instructions on loading additional packages after the Operating System software has been installed and the system has been configured.

Configuring the System for Operation

Once you have installed the system software, you are ready to set the system up for operation. This section leads you through this procedure.

Before You Begin

You should have installed all of the desired packages from the Operating System, including the Enhanced Security Set. You should not yet have installed any add-on packages not part of the Operating System.

Procedure

To configure an Enhanced Security system for operation, perform the following steps in single-user non-secure mode:

1. Read **`root`**'s mail; the installation scripts should have sent mail confirming the installation of the packages, and this mail should be read and deleted so that no installation-related mail remains in **`root`**'s mailbox. The following screen shows you how to do this:
2. If you will be using the Mandatory Access Control feature, define the security classifications, categories, levels, and aliases necessary for your site. While the system is delivered with a set of predefined classifications, categories, levels, and aliases, you will probably want to define new ones to enhance data security on your system.

The **`lvlname`** command, with various options, is used to define new classifications, categories, levels, and level aliases. Carefully read and follow the directions given in the section *"Classifications, Categories, Levels, and Aliases,"* in Chapter 17, "Administering Mandatory Access Control and Multilevel Directories" to add the required new classifications, categories, levels, and level aliases.

CAUTION

Do not change or remove any of the predefined classifications, categories, levels, or level aliases delivered on your system.

```
# mailx <RETURN>
mailx version 4.2mp  Type ? for help.
“/var/mail/root”: 11 message 11 new
>   1 root              Wed May  6 21:25    7/163
    2 root              Wed May  6 21:43    7/178
    3 root              Wed May  6 21:52    7/183
    4 root              Wed May  6 21:57    7/172
    5 root              Wed May  6 22:06    7/159
    6 root              Wed May  6 22:11    7/182
    7 root              Mon Aug  3 14:47    7/181
    8 root              Mon Aug  3 14:51    7/160
    9 root              Mon Aug  3 15:05    7/162
   10 root              Mon Aug  3 15:19    7/163
   11 root              Mon Aug  3 15:53    7/169
? <RETURN>
.
.   Repeat until last message is displayed.
.
Message 11:
From root Mon Aug  3 15:53 EDT 1993
Content-Type: text
Content-Length: 93
Status: R

Installation of Enhanced Security Utilities on hal as package
instance <es> was successful.

? d 1-11 <RETURN>
? h <RETURN>
No applicable messages
? q <RETURN>
#
```

Also, see the **`lvlname(1M)`** online manual page.

3. Add groups to your system with the **`groupadd`** command. Enter a **`groupadd`** command for each group that you want to add. Each command line must be of the form:

   `groupadd [`**`-g`** *gid*`]` *group_name*

where *gid* is the group number you want to assign (other than the next sequential group number available) and *group_name* is the name of the group.

4. Add user logins to your system with the **`useradd`** command. Enter a separate **`useradd`** command for each user you want to add.

Some points that must be considered:

- You may want to define a specific audit event mask for one, some, or all users. See the *“Setting Audit Criteria for Users”* section of the “Enabling Auditing” chapter in the *Audit Trail Administrator's Guide.*
- If you will be using Mandatory Access Control, you need to define the levels at which each user can log in using this command.
- Users that will be assigned particular roles as administrators need to be members of particular groups. See *“Administering Group*

Memberships for Users" in Chapter 14, "Managing User Accounts and Groups" in this book.

Chapter 14, "Managing User Accounts and Groups" in this book describes how to use the **useradd** command.

See the **useradd(1M)** online manual page for a complete description of all the options and arguments accepted by the **useradd** command.

You must also assign groups to users that will hold administrative roles. See *"Administering Group Memberships for Users"* in Chapter 14, "User Account and Group Management" in this book for more information.

5. For each user login created in the previous step, and for the **root** login, use the **passwd** command to set up the desired password aging and create a password for each login. The **passwd** command must be executed twice: once to set the password aging options and again to actually assign a password to the user. Chapter 14, "Managing User Accounts and Groups" in this book describes how to perform this task.

6. Lock the **lp** and **setup** logins. Enter two separate **passwd** commands, as shown in the screen below:

```
# passwd -l lp <RETURN>
# passwd -l setup <RETURN>
```

These logins are now locked and cannot be used to log in to the system. However, these logins are necessary, primarily to own certain system files, and to help provide subsystem separation.

For example, the Line Printer Subsystem requires that a user be the **lp** user to perform certain operations. This is done by having certain programs execute set-uid to **lp**, so that they become the **lp** user only for the time that is strictly necessary. This way, there is no need for anyone to be able to log in as **lp**, and hence the login should be locked.

7. Create any new roles required with the **adminrole** command, and assign user logins to administrative roles using the **adminuser** command. There are several administrative roles defined by the system; an administrative user should be assigned to each. Typically, a user should hold no more than one administrative role, to help provide accountability, but this may not always be possible. The pre-defined roles are: OP (operator), AUD (auditor), SOP (security operator), and SSO (site security officer). These provide for B2 operation, but should be filled whether or not your system will be run with B2 functionality. At a minimum, the SOP and AUD roles must be assigned to a user. If this is not done, it will not be possible to properly administrate your system in multi-user mode.

See Chapter 10, "Trusted Facility Management" for guidelines on assigning users to roles (including group membership requirements for the different roles), and for a list of the activities that each role is authorized to perform on the system.

8. Enable the port monitors on your system. Follow the steps outlined in Chapter 5, "Managing Ports" of this book.

9. If you desire, define Secure Attention Keys (SAKs) for all terminal lines (**`/dev/term/*`**). See Chapter 15, “Administering Printers, Terminals, and Devices” of this book.

10. Set up any printers following the directions given in Chapter 15, “Managing Terminals, Printers, and Other Devices” in this book. Remember to set the desired level ranges for the printer devices. Also see the **`lpadmin(1M)`** and **`putdev(1M)`** online manual pages for more information.

    Note that you must still enable the printers as shown in the section *“Enabling Printers”* after you bring the system up in the multi-user state.

11. At this time, you should install any other packages that your user community will require. See the *“Adding Third Party Software to an Enhanced Security System”* for information on doing this.

12. You can use the **`cpio`** command at this time to load data onto the system, in the event that you are, for example, upgrading an existing system from a previous release of the Operating System.

    If the data being imported consists of user files only, then you need to import the data and move it to appropriate user directories; then, you must assign an appropriate level to each file so that the intended user(s) of the files can access them.

    If programs and their associated data files are installed from a backup medium, you need to determine an appropriate location and security level for them so that users can access them. It is possible that some of these files will need privilege to function properly; it is up to you as administrator to determine which privilege(s) are required and assign them to the file.

    It is important to understand that installing files at `SYS_PUBLIC` and `SYS_PRIVATE`, and/or associating privilege(s) with files, essentially makes these files part of the Trusted Computing Base. Placing files at a level where an administrator can execute them, or allowing a file to possess privileges, provides a possible avenue for attack by a malicious user.

    If you do not assign a level to a file, the file will be assigned the level `SYS_RANGE_MAX` on creation in single-user mode; when the system transitions to multi-user state, only users with `macread` and/or `macwrite` privileges will be able to access these files. Use the **`chlvl`** and **`filepriv`** commands to assign appropriate levels and privileges, if necessary, before going to the multi-user state.

After Completion

After completing this procedure, you may reboot your system into multi-user mode. See Chapter 3, “Booting and System States” in this book for information on how to do this.

Enabling Printers

The first time you bring the system up into the multi-user state, the administrator whose login is assigned to the OP role must log in and enable the printers you set up previously in this procedure.

The printers must be enabled in multi-user state, since the commands used to accomplish printer enabling require that the **lpsched** daemon is running.

For example, if the system had two printers, called *user1* and *admin1*, the administrator would execute the following commands to enable the printers: **$ tfadmin accept user1 admin1** <RETURN>
$ tfadmin enable user1 admin1 <RETURN>

Once these commands are executed, the lp scheduler (lpsched) will enable the printers on each reboot automatically, until a **disable** or **reject** command is executed for one or both of the printers. If this is done, the administrator in the OP role will need to execute the **accept** and/or **enable** commands again in multi-user state to re-enable the printer(s).

Setting the Audit Criteria for the System

Once the system has come up in the multi-user state, the user assigned to the AUD role should establish the auditing criteria for the system. While the system is delivered with a default set of system-wide audited events, you may want to add or remove events for system-wide auditing, or establish object-level audited events. The **auditset** command is used to set these auditable events. See the "Enabling Auditing" chapter of the *Audit Trail Administration* guide for instructions on choosing and setting auditable events. To display the default audited events, enter:

```
tfadmin auditset <RETURN>
```

The auditing of events will be started automatically each time the system changes to multi-user state (state 2).

Changing or Modifying Files on an Enhanced Security System

Incorporating Modified Kernel Modules into the System

If a problem is found in a kernel module, the module must be restored from backup media. In the case of a fix delivered to you by the vendor, the fix will be distributed on either disk or tape.

This module must be loaded on to the hard disk properly, or the security of the system may be compromised. This section presents a general procedure for restoring a module and applying fixed modules to the system.

Before You Begin

To begin, you must be logged in as an administrator (SOP or SSO) that can execute the **`shutdown`** command.

Procedure

To install a modified kernel module, perform the following steps:

1. Prepare the system to reboot using non-secure kernel and bring the system down; see the Chapter 3, “Booting and System States” in this book for information on how to do this.
2. Reboot the system into single-user mode. Chapter 3, “Booting and System States” explains the procedure to be followed. Mount all the file systems by entering:

```
# mountall <RETURN>
```

3. Insert the tape or disk containing the fixed TCB module into the tape or disk drive as appropriate; make sure to engage the locking mechanism on the device.
4. Unless the tape or disk is accompanied by other instructions, enter the following at the # prompt:

```
# tcpio -iduvI device_name
```

where *device_name* is the name for the tape drive.

The system will print out the name of each file installed.

5. After the **`tcpio`** command is completed and the # returns, enter the following:

```
# initprivs <RETURN>
```

This will check and set (if necessary) all privilege information associated with all files on the system, including any you just installed.

6. Bring the system back to multi-user mode, following the procedure in Chapter 3, “Booting and System States” in this book.

Changing Levels of Existing Commands

As shown in Chapter 13, “Maintaining an Enhanced Security System”, certain commands (or options of commands that require privilege) are restricted to execution in single-user mode. You may want to make some of these commands executable in multi-user state for your convenience.

The **`mkfs`** and **`filepriv`** commands are good examples. While these executable files are installed on the system with all the privileges needed to perform the functions for which they were designed, their privileged use in multi-user state is prevented by the system design.

This means, in the case of **`filepriv`**, that users can use the command to query the privileges on files they can access in multi-user mode, but cannot set privileges on files. This is accomplished by installing the file at `SYS_PUBLIC`, so that it is accessible to all users (including administrators); however, the file is not assigned to any role in the Trusted Facility Management (TFM) database, so no user or administrator can execute it with privilege to set privileges on executable files.

In the case of **`mkfs`**, this means that the command cannot be used at all in multi-user state. Since the executable is installed at `USER_PUBLIC`, users can access the executable, but cannot execute it with privilege; administrators are prevented from accessing the executable at all since they are logged in at `SYS_PRIVATE` or `SYS_OPERATOR` in multi-user state.

Procedure

To make commands like **`mkfs`** or **`filepriv`** available to administrators in multi-user state, perform the following steps:

1. Change the level on the executable to `SYS_PRIVATE` or `SYS_PUBLIC` (if it is not installed with either of these levels). A level of `SYS_PUBLIC` allows any user to access the executable, while a level of `SYS_PRIVATE` allows only administrative logins logged in at the same level to access it.

2. Add the command to one or more roles (as appropriate for your site) in the TFM database, if it is not already assigned to one or more roles. Appropriate privilege must be given to the entry in the TFM database to execute the command.

Use the **adminrole** command (in single-user mode) to list the role definitions on your system. Refer to Chapter 10, “Trusted Facility Management” for the use of the **adminrole** command.

Adding Third Party Software

In general, the packages you may want to add to the Enhanced Security system fall into two broad categories:

- packages whose files need to be installed at SYS_PRIVATE and/or SYS_PUBLIC; these files essentially become part of the Trusted Computing Base, and as such provide a possible route of attack on your system by malicious users. These files may also require privilege(s) in order to function properly.
- packages whose files can be installed at USER_PUBLIC or other levels above USER_PUBLIC; these files (such as a spreadsheet program) are not part of the Trusted Computing Base and as such do not pose as much of a threat to system security, as administrators generally cannot execute such programs, and they generally do not require privilege.

It is important to determine the appropriate privileges and levels for all files installed by a package before it is installed.

If the software has been enhanced to work on top of the Enhanced Security Utilities the installation software delivered with the package should assign privileges and levels to the files it installs.

If the software has not been enhanced, you will need to use the **chlvl** and **filepriv** commands to assign appropriate levels and privileges.

NOTE

Please note that all software should be added in single-user mode. If you are first turning on your machine or if you are rebooting, boot from **m\OS** and go to **init** level 1 before adding any software.

Determining File Levels

It is a good idea to list, before installing, the files to be installed by a particular software package, and determine appropriate levels for each file.

If the **pkgadd(1M)** command is used to install software, it will, by default, install software at the level USER_PUBLIC when the **chlvl** command is available. Of course, if the package has been specifically created for use with the Enhanced Security Set installed, it may already contain levels for the files it installs, which will override this default. The **pkgadd** command should always be used in single-user mode.

NOTE

> If **pkgadd** is used to install a pre-Release 4 package, the files installed are given the SYS_RANGE_MAX level, unless the package includes an **rlist** (a list of files contained in the package), in which case the files are installed at USER_PUBLIC.

The **cpio(1)** command, by default, installs software at the level SYS_RANGE_MAX when executed in single-user mode. This means privilege will be required to access the files. Otherwise, you must change the levels of the files yourself, using the **chlvl** command. If you are in the multi-user state, and the **cpio** command is used, it will, by default, install software at the level of your current process.

If the levels that will be set upon installation are not acceptable, you will have to determine the correct levels and set them yourself using the **chlvl** command. The following guidelines should be followed for setting the file levels:

- In general, if the files are for administrative use only, they should be installed at the SYS_PRIVATE MAC level.
- If the files are to be accessible to both users and administrators, they should be installed at the SYS_PUBLIC MAC level.
- If the files are to be used by non-administrative users only, they should be installed at the USER_PUBLIC MAC level.

Determining File Privileges

Most software, such as spreadsheets or word processors, can be installed at the USER_PUBLIC level with no privileges on the executable files delivered with the package.

Some software, however, may require privilege in order to function properly. This is generally to satisfy one of two needs:

- an extraordinary access requirement (the need to read, write, or execute files owned by a particular user, or at a particular level). An example might be a command that writes to a log file at SYS_PRIVATE, which is normally inaccessible to regular users. The dacread, macread, dacwrite and macwrite privileges are used to resolve access problems.
- the need to override a system restriction. A common example would be the need to use the sysops privilege to override a system imposed limit, such as the ulimit or number of blocks in a file system that can be consumed by a user process. Adding sysops to the executable as a fixed privilege with the **filepriv** command, enables the executable to override such limitations.

Determining the privilege(s) needed by an executable is a difficult task, but one that must be done if the security of the system is to be upheld.

If Source Code Is Available

If source code is available for a command, examine the source to determine what privileges are needed for the proper functioning of the command. The following guidelines point out some of the items to examine:

- System calls issued by the command may require one or more of the privileges listed on the system call's online manual page.
- Library routines may also require privilege; the system calls listed in the library function's *"SEE ALSO"* section should be scanned for possible privilege requirements.
- Pay attention to the execution flow. An executable file or shell might require privilege solely because it executes something that requires those privileges.

Whether or Not Source Is Available

There are other guidelines that should be considered whether or not source is available.

- If an executable fails consistently with the error `Permission denied`, then one or more of the privileges `dacread`, `dacwrite`, `macread`, and/or `macwrite` will be required [see **`intro(2)`**]. Typically the MAC override privileges are needed when an executable needs to access resources at non-dominated MAC levels.
- If the file is installed as set-user-ID, it probably requires privilege. If the user ID is non-root, (such as is done for several of the LP commands, which are installed set-user-ID to **`lp`**), the need for privilege is probably for access reasons. Otherwise, if the executable is installed to run set-user-ID to **`root`**, it may be because a system restriction must be overridden. If the need for privilege is only for access reasons, it may be enough to place a `setuid` privilege on the file. Otherwise, the correct combination of privileges must be deduced through other means, including examining the source, if available, or using the **`truss`** command, as detailed below.
- Run the application while logged in at a different level than that at which the application is installed, specifically, a level that dominates the level at which the application's files are installed. For example, if an application is installed at `USER_PUBLIC,` execute it while logged in at `USER_LOGIN`. This will point out potential access problems. In general, you should execute the application while logged in at the level that typical users of that application will be logged in when executing the application.

Procedure

To determine privileges needed when source code is not available, perform the following steps:

Step 1: While in single-user mode, execute the **`truss`** command [see **`truss(1)`**] to obtain the system calls used.

Screen 12-1 shows an example of the output displayed from executing **`truss`** on the **`mkfs`** command. The output is captured in a file, **`/tmp/truss.out`**, which is then examined.

NOTE

When executing **`truss`**, the specified command is executed so the system calls can be traced. Be careful not to overwrite or destroy existing files when using the **`truss`** command.

Step 2: Examine each system call for all privileges associated with it by referring to the system call online manual pages. The error code descriptions on the page mention the privilege(s) required to avoid the error conditions, with the exception of some override privileges, such as `dacread`, `macread`, `dacwrite`, and `macwrite`. See **`intro(2)`** for a complete list of privileges.

Step 3: While in single-user mode, assign privileges to the command on a trial-by-error basis until you find the correct combination of privileges necessary to execute the command while supporting the least privilege concept.

Step 4: Ensure that the executable is at the correct MAC level.

As a last resort, you can assign `allprivs` to the executable, guaranteeing that it will not fail because of a lack of privilege; however, doing so violates the concept of least privilege and may give the executable enough privilege to inadvertently damage the data on your system.

Adding System Commands

The following procedure shows you how to add a command to make it accessible to an administrator:

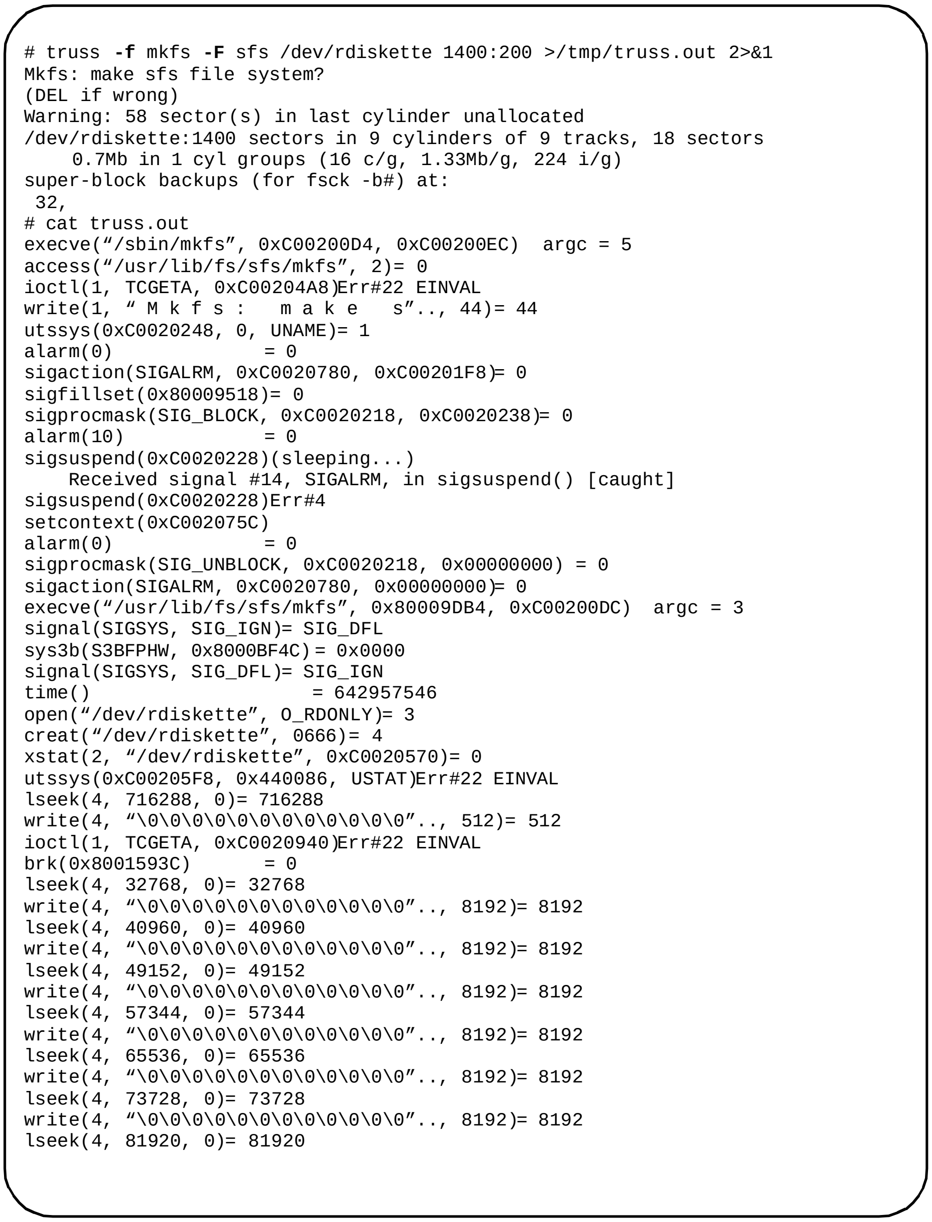
Step 1: Bring your system down from multi-user state to single-user mode.

Step 2: Assign privileges to the command which enable it to execute without error. This is accomplished by:

- determining which privileges are needed (see the section entitled *“Determining File Privileges”),* or
- assigning `allprivs` to the command.

CAUTION

Assigning `allprivs` to a command may compromise the security of your system. Privileges should not be assigned to a program without a full understanding of its actions.



```
# truss -f mkfs -F sfs /dev/rdiskette 1400:200 >/tmp/truss.out 2>&1
Mkfs: make sfs file system?
(DEL if wrong)
Warning: 58 sector(s) in last cylinder unallocated
/dev/rdiskette:1400 sectors in 9 cylinders of 9 tracks, 18 sectors
    0.7Mb in 1 cyl groups (16 c/g, 1.33Mb/g, 224 i/g)
super-block backups (for fsck -b#) at:
 32,
# cat truss.out
execve("/sbin/mkfs", 0xC00200D4, 0xC00200EC)  argc = 5
access("/usr/lib/fs/sfs/mkfs", 2)= 0
ioctl(1, TCGETA, 0xC00204A8)Err#22 EINVAL
write(1, " M k f s :   m a k e   s".., 44)= 44
utssys(0xC0020248, 0, UNAME)= 1
alarm(0)              = 0
sigaction(SIGALRM, 0xC0020780, 0xC00201F8)= 0
sigfillset(0x80009518)= 0
sigprocmask(SIG_BLOCK, 0xC0020218, 0xC0020238)= 0
alarm(10)             = 0
sigsuspend(0xC0020228)(sleeping...)
    Received signal #14, SIGALRM, in sigsuspend() [caught]
sigsuspend(0xC0020228)Err#4
setcontext(0xC002075C)
alarm(0)              = 0
sigprocmask(SIG_UNBLOCK, 0xC0020218, 0x00000000) = 0
sigaction(SIGALRM, 0xC0020780, 0x00000000)= 0
execve("/usr/lib/fs/sfs/mkfs", 0x80009DB4, 0xC00200DC)  argc = 3
signal(SIGSYS, SIG_IGN)= SIG_DFL
sys3b(S3BFPHW, 0x8000BF4C)= 0x0000
signal(SIGSYS, SIG_DFL)= SIG_IGN
time()                    = 642957546
open("/dev/rdiskette", O_RDONLY)= 3
creat("/dev/rdiskette", 0666)= 4
xstat(2, "/dev/rdiskette", 0xC0020570)= 0
utssys(0xC00205F8, 0x440086, USTAT)Err#22 EINVAL
lseek(4, 716288, 0)= 716288
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 512)= 512
ioctl(1, TCGETA, 0xC0020940)Err#22 EINVAL
brk(0x8001593C)       = 0
lseek(4, 32768, 0)= 32768
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 8192)= 8192
lseek(4, 40960, 0)= 40960
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 8192)= 8192
lseek(4, 49152, 0)= 49152
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 8192)= 8192
lseek(4, 57344, 0)= 57344
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 8192)= 8192
lseek(4, 65536, 0)= 65536
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 8192)= 8192
lseek(4, 73728, 0)= 73728
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 8192)= 8192
lseek(4, 81920, 0)= 81920
```

Screen 12-1. Executing truss on a Command

```
write(4, “\0\0\0\0\0\0\0\0\0\0\0\0”.., 8192)= 8192
lseek(4, 24576, 0)= 24576
write(4, “\0\0\0\0\0\0\0\0 & RC0EA”.., 8192)= 8192
lseek(4, 0, 0)       = 0
write(4, “\0\0\0\0\0\0\0\0\0\0\0\0”.., 8192)= 8192
lseek(3, 24576, 0)= 24576
read(3, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048) = 2048
lseek(4, 24576, 0)= 24576
write(4, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048)= 2048
lseek(4, 98304, 0)= 98304
write(4, “\0\0\004\0\f\001 .\0\0\0”.., 8192)= 8192
lseek(3, 24576, 0)= 24576
read(3, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048) = 2048
lseek(4, 24576, 0)= 24576
write(4, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048)= 2048
lseek(3, 32768, 0)= 32768
read(3, “\0\0\0\0\0\0\0\0\0\0\0\0”.., 8192) = 8192
lseek(4, 32768, 0)= 32768
write(4, “\0\0\0\0\0\0\0\0\0\0\0\0”.., 8192)= 8192
lseek(3, 24576, 0)= 24576
read(3, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048) = 2048
lseek(4, 24576, 0)= 24576
write(4, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048)= 2048
lseek(4, 106496, 0)= 106496
write(4, “\0\0\002\0\f\001 .\0\0\0”.., 1024)= 1024
lseek(3, 24576, 0)= 24576
read(3, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048) = 2048
lseek(4, 24576, 0)= 24576
write(4, “\0\0\0\0\0\0\0\0 & RC0EA”.., 2048)= 2048
lseek(3, 32768, 0)= 32768
read(3, “\0\0\0\0\0\0\0\0\0\0\0\0”.., 8192) = 8192
lseek(4, 32768, 0)= 32768
write(4, “\0\0\0\0\0\0\0\0\0\0\0\0”.., 8192)= 8192
lseek(4, 8192, 0)= 8192
write(4, “\0\0\0\0\0\0\0\0\0\0\010”.., 8192)= 8192
lseek(4, 90112, 0)= 90112
write(4, “\0\0\002\0\0\0 I\0\0\0DD”.., 1024)= 1024
lseek(4, 16384, 0)= 16384
write(4, “\0\0\0\0\0\0\0\0\0\0\010”.., 8192)= 8192
fsync(4)             = 0
close(3)             = 0
close(4)             = 0
write(1, “ W a r n i n g :   5 8  “.., 214)= 214
_exit(0)
#
```

Use the **filepriv -i** command [see **filepriv(1M)**] to set the required privileges. The following screen shows an example of assigning inheritable privileges to the **mkfs** command.

```
# filepriv -i dev,sysops,macread,macwrite /sbin/mkfs
# grep mkfs /etc/security/tcb/privs
29352:64311:641836815:%inher,dev,sysops,macread,macwrite:/sbin/mkfs
```

In the case of **sysadm** screens, all related task forms and files need to be given the appropriate privileges.

Step 3: Use the **chlvl** command [see **chlvl(1M)**] to change the level of non-TCB commands to `SYS_PRIVATE` or `SYS_PUBLIC`. The following screen shows an example of changing the level of the **mkfs** command from USER_PUBLIC to SYS_PRIVATE.

```
# ls -lz /sbin/mkfs
-r-xr-xr-x    4 bin      bin        29244 Mar 28 19:54 /sbin/mkfs  USER_PUBLIC
# chlvl SYS_PRIVATE /sbin/mkfs
# ls -lz /sbin/mkfs
-r-xr-xr-x    4 bin      bin        29244 Mar 28 19:54 /sbin/mkfs  SYS_PRIVATE
```

The level at which you want commands to reside depends on who you want to be able to access the command. Assigning a command a level of

SYS_PRIVATE allows only those users logged in at SYS_PRIVATE (with appropriate privileges) to access it; a command with a level of SYS_PUBLIC can be accessed by anyone with appropriate privileges.

In the case of **sysadm** screens, all related directories, task forms and files need to be changed to SYS_PRIVATE.

Step 4: Populate the appropriate user/role in the TFM database with the command name and required privileges. See the section entitled *“Administrative Logins and Roles”* in this chapter for instructions on populating the TFM database.

Step 5: In the case of **sysadm** screens and shell scripts, use the TFADMIN environment variable as described in *Programming with System Calls* to ensure that commands that require privilege are executed with the necessary privileges; of course, the executing user must appear in the TFM database.

13
Maintaining an Enhanced Security System

13
Maintaining an Enhanced Security System

Introduction

This chapter covers procedures and guidelines you should use in maintaining your Enhanced Security System. The specific topics covered include:

- Crontab entries — special procedures need to be followed on Enhanced Security systems to ensure the secure operation of the **`cron`** daemon.
- Removable media — a section of this chapter talks about guidelines you should follow for handling physical, removable media, such as disks and tapes.

A number of other procedures are described in Chapter 19, “Security Procedures” in this book.

Auditing

The *Audit Trail Administrator's Guide* contains a complete description of the auditing subsystem, including a description of the audit trail reporting tool (**`auditrpt`**), a description of the tool's output, a list of the default auditable events, a complete description of all audited events, and the location and maintenance of the audit log.

Editing System Files

In the event that it is necessary for you to edit a file (at any level) while logged in as an administrator, use only the trusted editor, **`/usr/bin/ed`**. This is the only editor supplied with the system that can inherit privilege, which may be necessary to access files and to write out changes. Also, some editors, notably **`vi(1)`**, do not preserve MAC levels on files if used in single-user mode or by a user logged in at a level different than that of the file being edited.

Regular users, of course, may use any editor to edit their files.

Note that any file created (as opposed to changing an existing file) using **`ed`** while in single-user mode will be given the level `SYS_RANGE_MAX` when the file is written to disk; the level of the file should be changed to an appropriate level (using the **`chlvl`** command) that reflects the sensitivity of the file's contents. If the file's level is not changed from `SYS_RANGE_MAX`, it will not be accessible to any process when MAC is running, unless the process has the appropriate privilege (generally `macread` and/or `macwrite`).

If you use **`ed`** to make changes to a file that already exists, the level will be unchanged.

See the **`ed(1)`** online manual page.

Administering crontab Entries

The **`cron`** daemon provides a facility for users and the system to run programs without requiring that they be associated with a login session, and allows these commands to be executed periodically at specified times. Special care must be taken to administer the crontab entries on an Enhanced Security system.

Modifying User crontab Entries

Administering user **`crontab`** entries requires some special procedures when:

- removing a crontab entry or entries for another user
- editing a crontab entry for another user

The following two sections show you how to administer user **`crontab`** entries, and why it might be necessary to do so.

Removing User crontab Entries

When deleting a user's login from the system, there is no automatic way to delete that user's **`crontab`** entries. The following manual steps must be performed to remove them. This procedure may also be used to delete an individual **`crontab`** entry that is causing system trouble (for example, by grabbing excessive system resources).

Before You Begin

The following steps must be performed by an administrator logged in at SYS_PRIVATE and assigned to the SSO role. an administrator logged in as **`root`**:

Note that the *level* referred to in the procedure below is not the fully qualified level name or alias, it is the *Level Identifier* (LID). See the “Mandatory Access Control” chapter in this book for more information on how to identify the LID for a given level.

Procedure

To remove some or all of a user's crontab files, perform the following steps:

Step 1: Execute:

```
cd /var/spool  <RETURN>
```

Step 2: Change the current **mldmode** to **real**:

```
mldmode -r  <RETURN>
```

Step 3: Look for directories at all levels named after the user's login:

```
tfadmin ls cron/crontabs/*/login <RETURN>
```

```
ls cron/crontabs/*/login <RETURN>
```

Step 4: To remove all the **crontab** files for the user, execute:

```
tfadmin rm cron/crontabs/*/login <RETURN>
```

```
rm cron/crontabs/*/login <RETURN>
```

To remove a single file, assuming you know which one to delete, based on the level, execute:

```
tfadmin rm cron/crontabs/level/login <RETURN>
```

```
rm cron/crontabs/level/login <RETURN>
```

where *level* matches the level of the **crontab** file you want to delete.

Step 5: To return to **virtual mldmode**, enter:

```
mldmode -v  <RETURN>
```

Editing User crontab Entries

Instead of deleting a user **crontab** file, which may be a drastic step in some cases, you could edit the file and alter the actions of the user's **crontab**. This requires bringing the system down to single-user mode, however, and may not be practical under all circumstances.

Before You Begin

Note that the *level* referred to in the procedure below is not the fully qualified level name or alias, it is the *Level Identifier* (LID). See the “Mandatory Access Control” chapter in this book for more information on how to identify the LID for a given level.

Procedure

To edit an entry in a user's crontab, perform the following steps:

Step 1: Bring the system down to single-user mode; follow the instructions in Chapter 3, “Booting and System States” in this book.

Step 2: Execute:

```
cd /var/spool <RETURN>
```

Step 3: Look for a directories at all levels named after the user's login:

```
ls -z cron/crontabs/*/login  <RETURN>
```

Step 4: Edit the appropriate crontab file using the trusted editor, **ed**:

```
ed cron/crontabs/level/login  <RETURN>
```

where *level* matches the level of the **crontab** file you want to delete.

See the **ed(1)** online manual page for instructions on using the editor; see the **crontab(4)** online manual page for a discussion of the format of **crontab** files.

Step 5: After you have finished editing the file, restore the level information to the file by executing:

```
chlvl level cron/crontabs/level/login  <RETURN>
```

where the *level* assigned to the file matches the *level* in the pathname and the level displayed for the file in Step 2, above.

Step 6: Return to the multi-user state by following the instructions given in the section *“Entering the Multi-User State During Boot Up”* in Chapter 3, “Booting and System States”.

Modifying Package crontab Entries

If packages have been uninstalled from the system and they have entries in the**crontabs** directory, the only way the administrator can remove these entries is by using manual steps similar to those described for removing or editing user **crontab** entries.

The packages that create **crontab** entries when they are installed are:

- Essential Utilities (`core`),
- Basic Networking Utilities (`bnu`),
- Software Distribution Service (`dist`),
- LP Print Service (**`lp`**),
- Message Logging and Monitoring (`msgmgt`),
- System Performance Analysis Utilities (`perf`),
- Simple Mail Transfer Protocol Utilities (`smtp`).

There are two effective directories under the **crontabs** multilevel directory with **crontab** files that are either created or updated when the above-mentioned packages are installed on the system. Figur e13-1 shows the effective directories under **crontabs** and the files that contain package **crontab** entries.

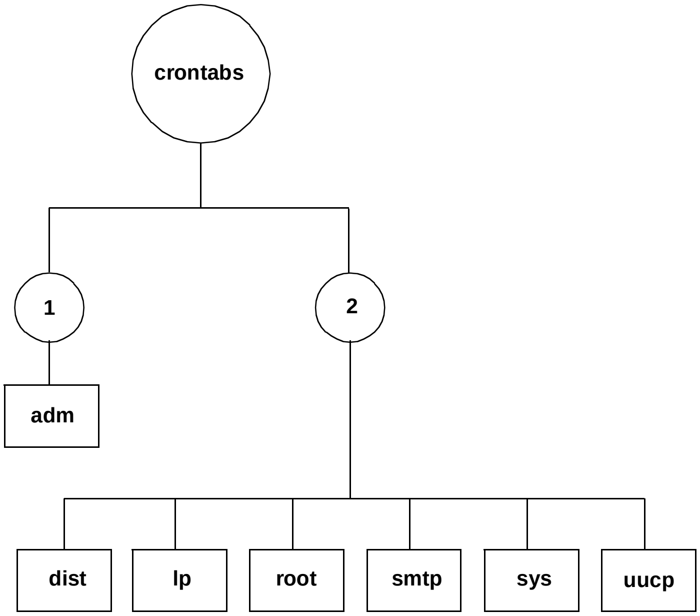


Figure 13-1. Package crontab Entries

To remove a package with the **pkgrm** command, it is necessary to be in single-user mode. After removing the package and before going to multi-user mode, you should remove any **crontab** entries for that package.

The `dist`, **lp**, and `smtp` packages create **crontab** files with the same name as the package name. If any of these packages are removed, you would remove the **crontab** file of the same name. The `bnu` package creates the **uucp** file in the **crontabs/2** directory and also appends an entry to the **root crontab** file. In this case, you would remove the **uucp** file and edit the **root** file. The other packages append entries to one or more of the **adm**, **root**, and **sys crontab** files. In this case, you would edit the **crontab** file of the appropriate login name to remove the entry from the file. Refer to the **postinstall** script for the package for more information concerning the **crontab** entries that a package makes on installation.

Removing Package crontab Entries

You should remove any **crontab** entries for a package when you remove the package.

Before You Begin

This procedure should be performed in single-user mode. See “Booting and System States” chapter in this book for information on how to place your system into this state.

Procedure

In order to remove a package **`crontab`** entry, perform the following steps:

Step 1: Execute:

```
cd /var/spool/cron/crontabs <RETURN>
```

Step 2: Determine the effective directory that contains the package **`crontab`** entry to be removed (refer to Figure 13-1 and change into that directory:

`cd` *level* `<RETURN>`

where *level* is either `1` or `2`.

Step 3: Remove the package **`crontab`** entry:

`rm` *login* `<RETURN>`

where *login* matches the name of the **`crontab`** file you want to remove.

Step 4: If you have no further tasks to perform in single-user mode, you can reboot your system into multi-user mode. See Chapter 3, “Booting and System States” for information on how to do this.

Editing Package crontab Entries

You may wish to alter the entries installed by a package. This may be because you wish the commands to run at a different time, or more or less frequently than specified by the default value.

Before You Begin

This procedure should be performed in single-user mode. See Chapter 3, “Booting and System States” in this book for information on how to place your system into this state.

Procedure

In order to edit a package **`crontab`** entry, perform the following steps:

Step 1: Execute:

```
cd /var/spool/cron/crontabs <RETURN>
```

Step 2: Determine the effective directory that contains the **`crontab`** file to be edited (refer to Figure 13-1 and change into that directory:

`cd` *level* `<RETURN>`

where *level* is either `1` or `2`.

Step 3: Edit the appropriate **`crontab`** file using the trusted editor, **`ed`**:

`ed` *login* `<RETURN>`

where *login* matches the login filename of the **`crontab`** file you want to edit.

See the **`ed(1)`** online manual page for instructions on using the editor; see the **`crontab(4)`** online manual page for a discussion of the format of **`crontab`** files.

Step 4: After you have finished editing the file, restore the level information to the file by executing:

`chlvl` *level login* `<RETURN>`

where the *level* assigned to the file matches the level indicated by the effective directory of the file as shown in Figure 13-1, and *login* is the name of the file you just edited.

Step 5: If you have no further tasks to perform in single-user mode, you can reboot your system into multi-user mode. See “Booting and System States” chapter for information on how to do this.

Using Removable Media

The following guidelines concerning the use of removable media that must be observed to safeguard the security of your system and any data stored on the media:

- All removable media must be physically labelled with a description of the sensitivity of the data contained on the media.
- All removable media should be stored in a physically secure place commensurate with the sensitivity of the data.
- No removable media should be provided for reading or writing at a security level lower than that described by the physical label, to protect against the inadvertent downgrading of information.
- If a removable storage media needs to be re-used at a level other than the one described on the physical label, the device should be bulk-erased and relabelled to prevent subsequent users from retrieving unauthorized data from the device.

Other Security Procedures

Other procedures you should perform to insure correct, secure system operation are documented elsewhere in this book. A quick reference to them follows:

- Check the state of your Mandatory Access Control database. See Chapter 17, Administering Mandatory Access Control and Multilevel Directories" chapter for information.
- Verify that no unexpected multi-level directories exist on your system. The directory **`/etc/security/MLD`** contains files from system packages listing all of the MLDs that were installed. See Chapter 17, Administering Mandatory Access Control and Multilevel Directories" chapter in this book for more information.
- Check the state of the SAK keys on your terminal lines. See Chapter 15, "Administering Devices, Printers, and Terminals" chapter for more information.
- The administrator serving in the AUD role should follow the procedures in the *Audit Trail Administration* guide to maintain the auditing subsystem.
- Other procedures you should follow can be found in Chapte r19, "Security Procedures" of this book. It contains general procedures you should implement that are applicable to systems with or without the Enhanced Security Utilities installed.

14
User Account and Group Management

14
User Account and Group Management

Introduction

One of the most vital links in system security is managing the user accounts and groups on your system. This chapter covers items of particular security relevance when creating new accounts and managing existing accounts. It should be read in concert with Chapter 4, “Creating and Managing User Accounts”. This chapter contains guidelines on items involved in managing and creating user accounts of particular security relevance. These items include:

- special administrative logins — what they are and how to manage them
- user passwords — setting them, and controlling how they are changed
- login levels for user accounts — guidelines for setting them, and how to set them
- privileges — assigning privileges using the Trusted Facility Management (TFM) roles and commands to users for use with the **`tfadmin`** command
- groups assigning groups to a login
- file creation mask — the system allows you to set a default file creation mask that applies to files created by users on the system

Administering Special Administrative and System Logins

A good mix of system use and system security is available to you with the use of special system logins that can be password-protected. These logins allow privileges to be split into smaller domains so fewer people have access to the entire system.

Access to these accounts is typically achieved in one of two ways:

- logging in to the account
- using the **`su`** command during a login session to become the desired user

Both of these access methods generally require the knowledge of the password for the account.

Limit distribution of the passwords for the following system logins only to those who need to know them.

Login	UID	Use
root	0	On a system that is not running with the default Privileged User Module, allow the user access to the entire system. See Chapter 5, “Administering Privileges” of this book for more information on privilege modules. Because the **root** login has no restrictions and overrides all other logins, protections, and permissions, the password for this login should be very carefully protected. On a system running with LPM, **root** does not automatically override other restrictions and protections, but the login still owns many of the most important and sensitive system files, and the password is still used for single-user mode. Protection of this password is therefore still very important.
sys	3	Owns many system files. Generally not necessary to be able to login to this account.
bin	2	Owns most of the commands. Generally not necessary to login to this account.
adm	4	Owns certain administrative files. Generally not necessary to login to this account.
uucp	5	Owns the object and spooled data files for uucp. Generally not necessary to login to this account.
nuucp	10	Used by remote computers to log on to the system and start file transfers. Local users, including administrators, should not need to login to this account. The password for this account must be known to remote sites that wish to transfer files or run commands. Therefore, particular care should be used in distributing this password, and it should be changed regularly.
daemon	1	System daemon login; controls background processing. Generally not necessary for users to login to this account.
lp	71	Owns the object and spooled data files for **lp**. Generally not necessary for users to login to this account.

Figure 14-1. System Logins and Uses

Most of these system logins allow a user access to critical portions of the operating system. There are several things you can do to protect these logins:

- All of these logins should have passwords assigned to them. Once this is done, any user attempting to use one of these accounts as a login is prompted for the password. See the *“Setting and Administering Passwords”* section of this chapter for more information on setting passwords and password guidelines.
- Consider locking some or most of these accounts. This will prohibit users from logging in on these accounts. However, users can still use the **su(1)** command to change their user and group IDs to that of another user,

provided they know the password. Many of these accounts are locked by default; this usually means no user should need to log in as one of these accounts. You should consider carefully if unlocking such an account is necessary. Logging in as **`root`**, or the ability to use the **`su`** command in a current login session, is almost always sufficient.

- Periodically check the **`loginlog`** and **`sulog`** files for suspicious activity. See Chapter 19, "Security Procedures" for information on how to do this.
- Verify that the **`/etc/default/su`** file does not contain a `PROMPT=` line. If this file contains a line of the form `PROMPT=no,` then any user running the **`su`** command will be able to change user and group IDs within a login session or **`cron`** job without being prompted for a password. This is a serious security violation, particularly in systems that are not running with the Enhanced Security Utilities installed.
- The Enhanced Security Utilities, specifically using the Least Privilege Module (LPM), can provide additional protection for these accounts. LPM requires that the user possess the `setuid` privilege to successfully use the **`su`** command, in addition to any other requirements (e.g. password). Possession of this privilege is controlled through the use of the TFM Database. See Chapter 10, "Trusted Facility Management" and Chapter 9, "Administering Privilege" of this book for more information.

Assigning Special Administrative Passwords

After you have set up your system, assign passwords to the special administrative and system logins. For instructions, see Chapter 4, "Creating and Managing User Accounts".

The Importance of the root Password

While the **`root`** login is not privileged in any way, it should not appear in the TFM database in any administrative role. This login still owns many sensitive files on the system; the password for this login is used for single-user mode, which places the system in a relaxed state of security for administrative tasks. Therefore, it is still a good idea to tightly control access to this login. If the **`root`** login's password is forgotten while the system is running, a new password can be established for **`root`** by the site security officer (SSO), using the **`passwd`** command.

Managing Passwords

To log on to the system, a user must enter both a login name and a password. Although logins are publicly known, passwords must be kept secret. To enhance the security of your system and data, ask your users to change their passwords occasionally. For a high level of security, normal users should do so about every six weeks, or whenever a user suspects their password may have been compromised by disclosure, inadvertent or otherwise, to an unauthorized party. System administration logins should be changed monthly or whenever a person who knows key passwords leaves the company or is reassigned.

Although voluntary compliance with this practice is desired, the operating system provides a mechanism, called password aging, to force compliance.

Choosing a Password

Most security break-ins of computer systems involve guessing the password of a valid login. While the **`passwd(1)`** command has some criteria for making a password hard to obtain using mechanical means, a clever person can sometimes guess someone's password just by knowing something about that person's habits and interests.

- Bad choices: names of family members or pets, car license numbers or telephone numbers, Social Security number, employee number, names related to a person's hobbies or interests, currently popular words (such as slang from TV shows), seasonal themes (such as “turkey” in November or “superbowl” in January). Also, any variations on this by substitution or addition of a special character.
- Good choices: puns, foreign words, reversed words (yekrut for turkey), or nonsense words (Mhallifwwas — Mary had a little lamb, its fleece was white as snow).
- Add a non-alphabetic character in the middle of the password (be careful about special characters such as `#` and `@` and control characters). Substitute a number for a similar letter (for example `0` for `o`, `3` for `e`, `1` for `l` or `i`).

Passwords must be constructed to meet the following requirements:

- A password must have at least six characters. Only the first eight characters are significant.
- A password must contain at least two alphabetic characters and at least one numeric or special character. Alphabetic characters can be upper case or lower case.
- A password must differ from the user's login name and any reverse or circular shift of that login name. For comparison purposes, the case of letters is ignored.
- A new password must differ from the old by at least three characters. For comparison purposes, the case of letters is ignored.

Examples of valid passwords are: `mar84ch,` `Jonath0n,` and `BRAV3S`. See Chapter 4, “Creating and Managing User Accounts” for more information about choosing and protecting passwords.

Password Aging

The password aging mechanism provides a method to force users to periodically change their passwords, in accordance with guidelines that you set. With this mechanism, you can force users to change their passwords periodically. You can also stop users who change passwords at your behest, and then immediately change them back to their previous value by requiring a specific interval to pass between changes.

It is strongly recommended that you utilize this mechanism to help strengthen the security of your system. You can activate password aging for selected logins by using the **`passwd(1)`** command.

The password aging information requires setting the following parameters for each login:

min the minimum number of days required between password changes

max the maximum number of days the password is valid

warn the number of days before the password expires that the user will begin receiving messages at login time warning that the account's password is about to expire.

As a result of using **`passwd`** to change the password, the following parameter will also change:

lastchanged the number of days between January 1, 1970, and the date the password was last modified

Displaying Password Information

Password and aging information can be displayed using the **`-s`** option of the **`passwd`** command.

Before You Begin

You must be in the SSO role to perform this action.

Procedure

To display password and aging information, perform the following steps:

1. For information on a single user, *user1*, type:

   ```
   tfadmin passwd -s user1
   ```

2. For information on all users, type:

   ```
   tfadmin passwd -s -a
   ```

Interpreting Password and Aging Information

For example, if you type:

```
passwd -s sms
```

the following information will appear if there is password aging.

```
sms PS 06/23/92 14 84 7
```

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

- login name (`sms`)
- password status (`PS`). The `PS` field may contain the following codes:

NP	no password for this login
LK	login is locked
PS	anything else

- date the password was last changed (`06/23/92`), also called *lastchanged*
- minimum number of days after *lastchanged* before the user can change the password (`14`)
- maximum number of days after *lastchanged* until the user will be forced to change the password (`84`)
- number of warning days before the password must be changed (`7`)

Thus, the information obtained for this example shows there is a password for the login `sms` that cannot be changed before July 6 and must be changed by September 15, 1992. On September 8, 1992, this user will begin seeing a warning message that the password will expire and should be changed.

A privileged user can use the **`-a`** option to see this information for all users:

```
passwd -s -a
```

Changing a Password

If, for example, a user forgets their password, the SSO can change another user's password.

Procedure

To change the password for user *login_name*, perform the following step:

1. Type:

   ```
   tfadmin passwd login_name
   ```

   Because this command can only be run by a privileged user, no prompt for the old password is given. Instead, the privileged user is prompted to enter the new password twice, to ensure it is typed accurately.

Turning On and Setting Password Aging

Password aging for an account is turned on by setting the *min* and *max* fields to appropriate values. The *warn* field can be set to provide advance warning to the user when their password is about to expire.

Before You Begin

You must be in the SSO role to perform this action.

Procedure

To turn on password aging, perform the following steps:

1. Use the **-n** and **-x** options to **passwd** to set the *min* and *max* values for the user. Type:

   ```
   tfadmin passwd -n min -x max login_name
   ```

2. If you wish the user to be warned beginning *warn* days before expiration that their password is about to expire, type:

   ```
   tfadmin
   passwd -w warn login_name
   ```

Note that the above steps can be combined into one command if desired.

Example

To turn on aging, set *max* to 84 and *min* to 7 days, respectively, and start reminding the owner to choose a new one fourteen days before the password expires:

```
tfadmin passwd -x 84 -n 7 -w 14 login_name
```

Starting 14 days before *max*, the user will see the message:

```
    Your password will expire in 14 days
```

The number decreases daily, until the password expires or is changed.

Forcing A Password Change

You may wish to force a user to change their password the next time they log in, if, for example, you suspect the current password may have been compromised.

Before You Begin

You must be in the SSO role to perform this action.

Procedure

To force a user to change the password at the next login session, perform the following step:

1. Type:

```
tfadmin passwd -f login_nameLocking a Password
```

Locking a Password

You may lock a login on the system. This does not allow anyone to log in to the account.

Before You Begin

You must be in the SSO role to perform this action.

Procedure

To lock a login and disallow anyone to use it to log in to the system, perform the following step:

1. Type:

```
tfadmin passwd -l login_name
```

Accounts that are locked in this manner will show a value of `LK` in the password status field.

Disallowing Password Changes

You may disallow password changes on accounts on your system. This forces users to use the current password, and ensures they will not change it to some other, possibly less secure value.

Before You Begin

You must be in the SSO role to perform this action.

Procedure

To disallow password changes on an account, perform the following step:

1. Change the *min* and *max* password aging values for the account so that the value of *min* is greater than the value for *max*. Type:

```
tfadmin passwd -n min -x max login_name
```

Note that password aging is disabled for such an account.

Example

To disallow changing the password for *login_name*, set *max* to 7 and *min* to 10 days.

```
tfadmin passwd -x 7 -n 10 login_name
```

Because *min* is greater than *max*, the password is locked and cannot be changed but the user can still log on to the system. Only a privileged user can change this password.

Turning Off Password Aging

Setting the *max* password aging value for a login account to **`-1`** turns off password aging. This means that there are no restrictions as to when, and how often, a user may change the password for their login.

Before You Begin

You must be in the SSO role to perform this action.

Procedure

To turn off password aging, perform the following step:

1. Type:

```
tfadmin passwd -x -1 login_name
```

Turning off password aging is not recommended.

More Information on Passwords

For more information, see the **`passwd(1)`**online manual page.

Setting Up Passwords and Aging Criteria for a New User Account

When a new user account is created, you should immediately give it a password and set the desired password aging values for it.

Before You Begin

It is assumed you are performing this action as part of the initial installation procedure.

Procedure

To set up the desired password and aging characteristics for a new account, perform the following steps:

1. Invoke the **`passwd`** command to set up the password aging characteristics and force the user to change their password the first time they log in. For example,

   `passwd` **`-f -n`** *min* **`-w`** *warn* **`-x`** *max login_name*

2. Create a new password for the login. Type:

   `passwd` *login_name*

3. Provide the new password when prompted. The system requests the password twice to make sure it is received incorrectly.

4. Provide the user with the password you have just assigned to their account.

This will allow the user to log in, and immediately force the password to be changed.

Dial-up Passwords

A dial-up password is an additional password users must enter before they are allowed access to a system. Use of the dial-up password is an option available to system administrators who want to tighten security on their systems.

Dial-up passwords can be assigned and changed only by administrators. When used, they should be changed at least once a month.

Creating a Dial-up Password

When you first establish a dial-up password, make sure you 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.

To protect your system with a dial-up password, you must create two files, **`/etc/dialups`** and **`/etc/d_passwd.`** The modes of the files should be `600`, and they should have owner and group set to **`root`**.

`/etc/dialups` contains a list of the terminal devices (ports) requiring the extra security of a dial-up password. (These are actually modem ports on the system.) It should look similar to this:

```
/dev/tty0_00    ( 0_00 )
/dev/tty0_01      |  |
/dev/tty0_02      |  |__ port number
  .   .    .      |_____ adapter number
  .   .    .
  .   .    .
/dev/tty0_07
```

/etc/d_passwd contains the encrypted password and the login programs requiring the user to enter a password before they can be invoked. It looks similar to this:

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

When a user attempts to log on onto any of the ports listed in **/etc/dialups**, the **login** program looks at **/etc/d_passwd** and may prompt the user for a second password. Whether a second password is requested depends on (1) which login shell is specified by the user's login entry in the **/etc/passwd** file, and (2) whether this login shell has an entry in **/etc/d_passwd**. The basic sequence is illustrated by Figur e14-2.

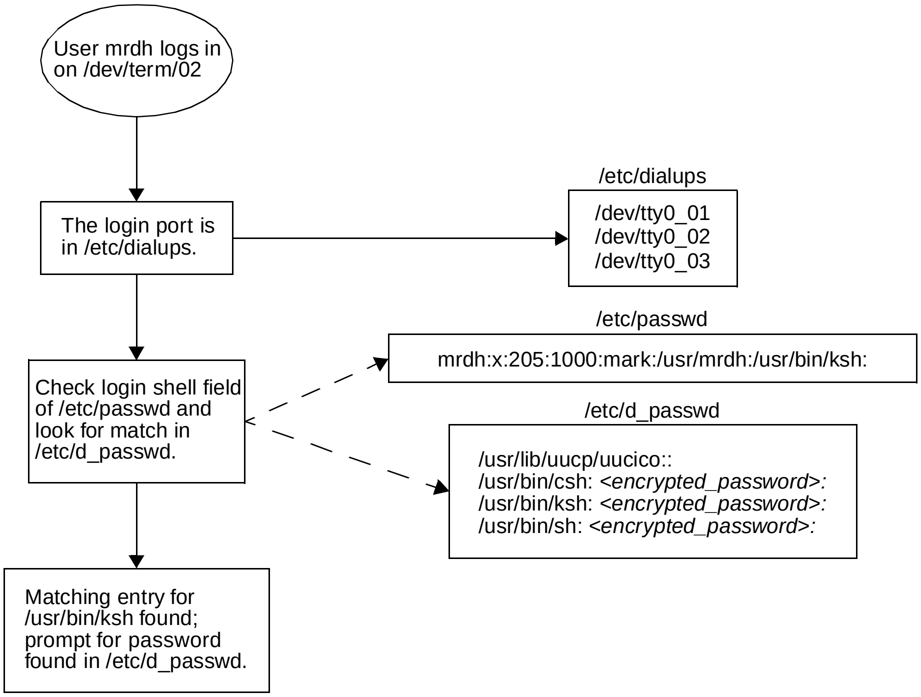

User mrdh logs in on /dev/term/02
The login port is in /etc/dialups.
/etc/dialups
/dev/tty0_01
/dev/tty0_02
/dev/tty0_03
/etc/passwd
mrdh:x:205:1000:mark:/usr/mrdh:/usr/bin/ksh:
Check login shell field of /etc/passwd and look for match in /etc/d_passwd.
/etc/d_passwd
/usr/lib/uucp/uucico::
/usr/bin/csh: <encrypted_password>:
/usr/bin/ksh: <encrypted_password>:
/usr/bin/sh: <encrypted_password>:
Matching entry for /usr/bin/ksh found; prompt for password found in /etc/d_passwd.

Figure 14-2. Basic Dialup Password Sequence

Because most users will be running a shell when they log on, the `/etc/d_passwd` file should contain entries for all shell programs. Some of the most commonly used shells are `uucico`, `sh`, `ksh`, and `csh`.

Each entry in /etc/d_passwd has two fields, separated by colons. The first field contains the name of the login program requiring a dial-up password. The second field contains the encrypted password.

In the example above, the first field in the entry for uucico is empty. This allows remote systems to call your system, via uucp, without having to know the dial-up password. The uucp subsystem is relatively secure, if properly administered, and usually does not need a dial-up password. However, if you are concerned about security here, it would be wise to require a password for uucp, too. The dial-up password need not be the same for uucp as for ksh, sh, and so on.

The entry for /usr/bin/sh defines the default dial-up password. If a user's login program is not listed in /etc/d_passwd, or if the login shell field in /etc/passwd is empty, the password defined in this entry will be used.

If there is no entry for /usr/bin/sh, users whose shell field in /etc/passwd is null or does not match any entry in /etc/d_passwd will not be prompted for a dial-up password.

Note the /etc/d_passwd file could be used to temporarily disable dial-up logins by putting an entry such as:

/usr/bin/sh:*:

by itself in the file.

Creating a Dial-up Password

To create a dial-up password, follow this procedure.

1. Using **useradd**, add a “dummy” user, say dummy.
2. Give it a password with the **passwd(1)** command.
3. Capture the encrypted password from **/etc/shadow** by typing

   ```
   grep dummy /etc/shadow > dummy.temp
   ```

4. Using **userdel**, delete the dummy user.
5. Edit **dummy.temp** and delete all fields except the encrypted password. Fields are delimited with a colon (:) and the password is the second field.
6. Edit the **/etc/d_passwd** file and read in the encrypted password from your **dummy.temp** file as the password field.

NOTE

If you have installed the Encryption Utilities, you can use the **makekey** utility to generate a password, and read that password into the **/etc/d_passwd** file as the password field.

Setting a Login Expiration Date

Occasionally, you may want to create a temporary login that “expires” after a short period. (Once it expires, its owner can no longer log on without your help.)

Before You Begin

This task must be performed by a user in the SSO role.

Procedure

To set an expiration date for a login,

1. type:

   ```
   tfadmin useradd -e mm/dd/yy login_name
   ```

   where *mm*/*dd*/*yy* is an absolute date:

   mm is a one or two-digit number representing the month (1-12)

   dd is a one or two-digit number representing the day of the month (1-31)

   yy is a two-digit number representing the year (00-99)

Extending a Login Expiration Date

If you decide to extend the expiration date of an existing login, you can do so with the **`usermod`** command.

Before You Begin

This task must be performed by a user in the SSO role.

Procedure

To extend an expiration date, perform the following step:

1. type:

   ```
   tfadmin usermod -e mm/dd/yy login_name
   ```

   where *mm*/*dd*/*yy* is an absolute date:

   mm is a one or two-digit number representing the month (1-12)

   dd is a one or two-digit number representing the day of the month (1-31) *yy* is a two-digit number representing the year (00-99)

(See the **`usermod(1M)`** online manual page for complete details.)

Deactivating a Login

It may be useful to know that a user has not logged in to the system for a while. You can set the "inactive" field of a login; if a user does not log on for the specified number of days, the login becomes inactive, and the user is prevented from logging in until the administrator resets the login.

Before You Begin

This procedure must be performed by a user in the SSO role.

Procedure

To set the inactive field,

1. type:

   ```
   tfadmin usermod -f n login_name
   ```

   where *login_name* will be considered inactive *n* days after *lastlogin*.

   You can also set the inactive field when adding a new user with **`useradd(1M)`**.

Reactivating a Login

Before You Begin

This procedure must be performed by a user in the SSO role.

Procedure

To reactivate an inactive login, complete the following steps:

1. Check the current inactive value:

   ```
   tfadmin logins -a -l login_name
   ```

   The first number on the second line of the output of this command is the inactive field.

2. Set the inactive field to zero:

   ```
   tfadmin usermod -f 0 login_name
   ```

3. Have the affected user log on again so *lastlogin* will be updated.
4. Reset the inactive field once again:

   ```
   tfadmin usermod -f n login_name
   ```

   where *n* is the number of days the login will remain active.

Displaying Login Information

The amount of warning time, the number of days before deactivation, and the expiration date can be displayed using the **`logins`** command. It displays a variety of information about the users on your system. Of interest here are the **`-a`** and **`-x`** options. For information on other options and uses of **`logins`**, see Chapter 4, "Creating and Managing User Accounts" and the **`logins(1M)`** online manual page.

Administering User Login Levels

On a system with the Enhanced Security Utilities installed, and the Mandatory Access Control (MAC) feature running, each user must be assigned one or more authorized login levels. You should assign a default security level to the user and possibly one or more other valid login levels. These levels will be the only levels at which the user will be allowed to log in to the system.

The user will be logged in at the default level unless the user specifies another valid level. If you assign the user more than one login level, the user may change the default login level. The user may specify the security level as either a full level name or a valid level alias. The specified level must dominate **`LVLLOW`** and be dominated by **`LVLHIGH`** if both of these values have been defined (see the *"Login Level Range"* section in this chapter).

Assigning Multiple Levels to Logins

A few additional steps by the administrator are required when assigning more than one login level to any user. The steps taken involve creating a home directory whose level is dominated by all other levels at which the user is able to log in.

The following two sections explain the different scenarios for users with multiple login levels.

Adding an Administrative User with Multiple Levels

If you want to add an administrative login to the system and allow that user to log in at either the level appropriate for their administrative role or a non-administrative login level, you must:

- specify a home directory level of SYS_PUBLIC via the **-w** option to **useradd** or **usermod**
- create one subdirectory under the user's home directory for each level at which the user is allowed to log in and assign one of the login levels to each directory
- copy the default **.profile** found in the home directory to each of the subdirectories. Note that the user will have to manually execute the **.profile** when logging in

An example illustrates why these steps are necessary. Suppose a user is added to the system as follows:

```
tfadmin useradd -m -h SYS_PRIVATE -h USER_LOGIN -v
USER_LOGIN \
-g audit -g users -w SYS_PUBLIC smith <RETURN>
```

If the home directory were assigned a level of either SYS_PRIVATE or USER_LOGIN, the user would not be able to log in at the other level, since these levels are disjoint (one does not dominate the other) and the user would not be able to change directory to their home directory when logging in at one of the levels. (To change directory to a given directory, the level of the current process must dominate the level of the given directory.)

Assigning a level of SYS_PUBLIC to the home directory solves this problem since both SYS_PRIVATE and USER_LOGIN dominate SYS_PUBLIC.

This does, however, introduce a further complication concerning the user's **.profile**, which is created in their home directory. While this file is created from the default system profile in **/etc/profile** and is executed upon login, the user will never be able to change it. This is true because the access control policy of the system requires that the level of the current process equal the level of the directory in which the user is attempting to write (or alter) a file. In fact, the user will never be allowed to create or alter any files in their home directory.

By creating a directory under the user's home directory for each level at which the user can log in, you allow the user to change directory to the subdirectory appropriate for the current login session, giving them the ability to create and modify files as they want.

The steps listed above are required for any login that is given the ability to log in at a level on the “SYS_ side” of the level structure and another disjoint level, such as USER_LOGIN.

Adding a Non-Administrative User with Multiple Levels

The same steps outlined in the previous section for adding an administrative user with multiple levels are required for adding a non-administrative user with multiple logins, if the login levels assigned via **useradd** are disjoint, except that the level assigned to the home directory should be USER_PUBLIC. The same motivations for this as outlined in the previous section apply.

However, if the levels assigned to the user are not disjoint, then the home directory need only be assigned the lowest level at which the user is allowed to log in.

For instance, if a user is added with the ability to log in at USER_1, USER_2, and USER_3, and the level USER_1 is dominated by the other two levels, the home directory could be assigned a level of USER_1; however, subdirectories for the other levels must still be created, and the user would not be able to create or modify any files in the home directory when logged in at the other levels.

It is recommended that the user create a **.profile** in each of the subdirectories that they will need to execute manually should they want certain commands to be executed every time they log in at those levels. The **.profile** found in the home directory, however, will always be executed, and some commands it attempts to execute may not be appropriate for the level at which they have logged in.

For this reason, it is recommended to assign the home directory a level ofUSER_PUBLIC in this case, and create a subdirectory for each login level.

Login Level Range

You can restrict the levels of new logins on the system by using the **defadm** command to set the **LVLHIGH** and **LVLLOW** values in the **/etc/defaults/login** file. When the system is delivered, both **LVLLOW** and **LVLHIGH** are undefined, so all login levels are allowed. Both must be defined to restrict the login level range.

During the login sequence, the level at which the user is attempting to log in is compared to the low and high values of the login level range. If the user's level does not dominate **LVLLOW** or is not dominated by **LVLHIGH,** the user is not allowed to log in.

CAUTION

> Be careful when setting the level range that you do not accidentally “lock yourself out” of the system. Make sure the level at which you log on to perform privileged administrative actions is within the newly defined login range.

The login level range affects only new login sessions; user processes may exist outside the login level range and may create child processes that inherit the parent's level. If you want to ensure there are no user processes outside the level range, you should reboot the system or make the change in single-user mode.

Screen 14-1 shows an example of setting and displaying the login level range.

```
$ defadm login LVLLOW LVLHIGH
defadm: WARN: name LVLLOW does not exist
defadm: WARN: name LVLHIGH does not exist
$ defadm login LVLLOW=SYS_PUBLIC LVLHIGH=secret:ALL
$ defadm login LVLLOW LVLHIGH
LVLLOW=SYS_PUBLIC  LVLHIGH=secret:ALL
$
```

Screen 14-1. Displaying and Setting the Login Level Range

The administrator displays the current login range, then changes it, and confirms the change. This change will allow users to log in at levels between SYS_PUBLIC and secret:ALL.

NOTE

In addition to setting a login level range for the system as a whole, you can set level ranges for specific terminals. A user's login level must be within both the system login level range and the terminal level range. See Chapter 15,"Administering Printers, Terminals, and Devices" for detailed information on terminal level ranges.

Assigning Administrative Roles

To enable administrative users to perform their duties, they generally must be assigned to a particular administrative role. Each role has associated with it a set of privileged commands. These commands can be executed by users assigned to that particular role. The commands will execute with the privileges defined for each in the role. Users may be assigned to more than one role. There are four pre-defined roles supplied with the system:

- auditor (AUD)— controls the auditing subsystem and prints the auditing reports
- operator (OP)— performs regular, low sensitivity administrative tasks, such as maintaining the LP subsystem, perform backups of user data, and more.
- security operator(SOP) — encompasses the operator role, but also has responsibility for other tasks that are more security sensitive, such as backing up system data, or mounting a file system.
- site security officer (SSO) — encompasses the SOP role, but also includes tasks that have a high security sensitivity, such as adding users, assigning passwords, assigning clearances to objects, and set ACLs on objects.

Roles may be created, modified, or deleted with the **adminrole(1M)** command. Role definitions and information on users' assigned roles and individual commands are contained in the TFM database.

The **adminuser(1M)** command is used to assign users to roles, or individual privileged commands to users. In general, the assignment of individual privileged commands to a single user is not recommended. Maintenance of the system is usually eased by incorporating a specific command or set of commands into a new role. The new role can then be assigned to as many users as desired.

Note that roles need not be confined to administrators. Individual users or groups of users that require privileged use of a program or set of commands, whether system commands, or an application, can be assigned to a role created specifically to meet their needs. This avoids having to grant privilege on the programs involved to everyone.

Specifics on the **adminuser** and **adminrole** commands and the TFM database can be found in Chapter 10, “Trusted Facility Management” in this book. However, there are some guidelines on assigning roles to users.

Managing Groups

Your system uses the concept of “group membership” as one method of access control. Each file on the system is a member of a particular group. Members of that group may have special access permissions. Establishing and maintaining user group assignments is one way you can control user access to specific directories and files on your system.

Administering Groups

Groups are administered on the system using the **groupadd(1M)**, **groupmod(1M),** and **groupdel(1M)** commands. You can change the group for a file using the **chgrp(1)** command. See Chapter 4, “Creating and Managing User Accounts” in this book for information on how to use these commands. Some guidelines should be kept in mind when administering groups on your system:

- The groups delivered with your system should not be deleted.
- System files should generally not be reassigned to another group. In some cases, such as for the **lp** group, membership in a particular group is essential for the system to function correctly.

Administering Group Memberships for Users

The **useradd(1M)**, and **usermod(1m)** commands are used to assign one or more groups to a user. See Chapter 4, “Creating and Managing User Accounts” for more information on these commands. The following guidelines should be applied to such operations:

- Logins assigned to administrative roles must be assigned group IDs as follows:

Role	Groups Required
OP	sys, lp
SOP	sys, adm, lp
SSO	sys, adm, bin, tty, lp
AUD	audit

The group memberships shown above are important since administrators gain access to some commands (such as the **enable** and **disable** commands) through group membership, rather than the inclusion of the commands in a role definition. Such commands are executed without privilege since they are not in a role definition.

- Do not assign any user membership in the following groups: **root**, **uucp**, **mail**, **nuucp**, **daemon**, **or priv**.

Administering the File Creation Mask

The default permissions used for files are determined by the values assigned to the file creation mask. The **umask** command assigns values to the mask. The system profile may call this command to set the default mask, which users may redefine for themselves. More information on the file creation mask can be found in Chapter 4, “Creating and Managing User Accounts”.

Changing the File Creation Mask

For maximum security, it is recommended that you set the default mask in the system profile. It should be set to a value of 077. New directories will then be created with read, write, and search permissions for the directory owner only. New files will then be created with read and write permissions for the owner only. This will particularly help naive users from inadvertently granting others access to files they create. More experienced users may wish to redefine the mask.

Before You Begin

This action should be performed in single-user mode by the Trusted System Programmer.

Procedure

To change the default mask, perform the following steps:

1. Open the **`/etc/profile`** file for editing, using your editor.
2. Add a line to the file of the form:

   `umask` *n*

   where *n* is the desired value. For maximum security, a value of `077` is recommended.
3. Write the changes back out to **`/etc/profile.`**

15
Administering Printers, Terminals, and Devices

15
Administering Printers, Terminals, and Devices

Introduction

This chapter covers the particular security aspects of administering devices. This includes the following:

- General Device File Protection — describes the system protection mechanisms that you should apply to devices, including the use of the Device Database (DDB)
- File Systems and Storage Devices — how to administer the secure device characteristics associated with these devices
- Terminal Devices — the function of the Secure Attention Key (SAK), how to set and view it for individual terminals
- Printers — how to set up printers in a secure environment
- Connecting New Devices — steps to follow to connect new devices to your system when it is running with the Enhanced Security Utilities installed

General Device File Protection

Device files on your system are protected by the same mechanisms that the system uses to protect other files. These include:

- file permission bits
- owner and group settings
- Access Control Lists (if the Access Control List Utilities have been installed)
- Mandatory Access Control (MAC) label (if the Enhanced Security Utilities have been installed)

Device Security Mechanism

Mandatory Access Control (MAC) and Discretionary Access Control (DAC) restrictions affect access to devices. However, additional access restrictions apply to devices. These restrictions prevent a non-privileged process from opening (or otherwise accessing) a device that has not been allocated for public use and restrict changes of the security level associated with the device.

All storage and I/O devices are protected by a range of security levels, which restrict what data can be stored on the device and the levels at which it can operate. For each device on the system, the security level range is defined by a minimum level and a maximum level that are used to enforce access restrictions. These restrictions ensure that a device receives only data that is appropriate for its location and configuration. For example, you would not want to print highly sensitive information on a printer located in a public area, accessible to everyone on the site. The level range of the device is designed to protect the sensitivity of the data flow on a device.

By default, every device on the system is a system private resource and can be accessed only by processes with the appropriate privileges. Explicit action by a trusted program is needed to grant unprivileged access to a device.

A device can be accessed by multiple pathnames residing on the file system. This allows a device with different characteristics to be accessed in different ways. For example, it is possible to have different pathnames for the same tape drive when it operates at different speeds. A mapping of the device pathnames to physical devices resides in the Device Database (DDB). Information on security characteristics of devices is also stored in the DDB, which is described under *"Maintaining the Device Database"* later in this chapter. The DDB also includes authorization information on device usage.

The security attributes in the DDB define the characteristics the device will have when it is allocated for use by the operating system's kernel. During allocation, the **`admalloc`** command validates any information passed against the information in the DDB. Therefore, the device must be defined in the DDB before it can be allocated. Once the device is allocated, the security characteristics are maintained in kernel data structures.

Basic Device Access Control

Because a physical device is represented by one or more device special files (DSFs), the device is always protected by the DAC and the MAC mechanisms as they apply to files.

The MAC and DAC information for the device is taken from the device special file used to access the device. The MAC and/or DAC information can be different on each device special file, and changing the DAC or MAC information on one device special file does not affect the access restrictions on other device special files that can be used to access the device.

For example, there can be different device special files for a tape device, one used when the tape operates at one density, another for a different density, etc. Each device special file would refer to the same physical device, but could have different access permissions; one set would not affect the other.

```
ls -l /dev/rmt/0h /dev/rmt/0m

crw-------   2 root     sys       17,127 May 20  1993 /dev/rmt/0h
crw-------   2 root     sys       17,128 May 20  1993 /dev/rmt/0m
```

To provide appropriate protection for the devices themselves, the system relies on device security attributes stored in the DDB. The device allocation routines consult the Device Database to obtain the list of device special files that map to the same physical device. The allocation routines make sure that none of the device special files are in use before they allocate the device and set the MAC and DAC attributes on one or all of the device special files.

Device Security Attributes

Associated with each device special file are several security-related attributes. These attributes, which are maintained by the kernel, provide an administrator a mechanism to regulate access to devices.

Four security attributes are associated with each device special file. The per-device attributes are:

release flag

This attribute shows how all the device security attributes are associated with a device. The release flag shows whether the device is allocated and the way in which it is allocated. The flag can have one of three values:

- persistent

  This value shows that the security attributes are set explicitly and remain associated with the device special file while the system is running or until the attributes are explicitly changed. The device is allocated, and the security attributes that the device now has will remain until explicitly changed.

- lastclose

  This value shows that the security attributes are set explicitly and remain associated with the device until the last reference to the device is closed. This value is only possible when the device has been allocated. Once there are no outstanding references to the device, the security attributes will be changed to those defined in the DDB for this device.

- system

  This value shows that the security attributes are set by the system. If the driver flag was set to be **initpub**, then the state of the device is set to **public**. Otherwise, it is set to **private**. The mode is set to **static**, and the **hilevel** and **lolevel** of the level range are set to level identifier (LID) values of zero (0) to show that the device does not have any applicable level range. The device is not allocated specifically to a user.

`state` — The state attribute must either be **`private`** or **`public`**. A device state of **`private`** shows that the device is a private Trusted Computing Base (TCB) resource and that unprivileged access to the device is denied. A device state of **`public`** shows that unprivileged access to the device is allowed. The state is changed from **`private`** to **`public`** when the device is allocated for unprivileged access (by, for example, **`admalloc`** or **`login`**), and is changed from **`public`** to **`private`** when the device is deallocated (again by, for example, **`admalloc`** or **`login`**).

level range — The device level range is represented by a **`hilevel-lolevel`** pair. The level range constrains the allowed values for the security level of the device and should be based on the physical constraints of the device (such as device location). The high level of the device level range must dominate the low level of the device level range. The device level (as set in the device special files for the device) must be contained in the level range.

`mode` — The device mode should always be **`static,`** which prevents changing the MAC level of the device if the device state is **`public`** and there are active I/O connections to the device. For all other cases, MAC level change is allowed.

The other possible value for the device mode, **`dynamic`**, is provided only for those sites that require dynamic changes in MAC levels while a device has open I/O connections. When the device mode is **`dynamic`**, MAC access checks are performed for each I/O operation. Thus, if the level of the device is changed, processes accessing the device must continue to pass MAC access checks. A device mode of **`dynamic`**, is provided as a means by which multi-window terminals can run several windows at different MAC levels. Generally, in all other cases, the mode should be **`static`**.

Logical and Secure Device Aliases

The DDB defines the attributes of all devices configured on the system. Each device is known to the system by an alias unique in the DDB. This alias is limited to 64 characters (`MAX_ALIAS`) and should contain only alphanumeric characters and the special characters underscore (_), period (`.`), dollar sign ($), or minus sign (`-`).

Two types of device aliases can be defined in the DDB.

secure device alias

This is the name of a physical device configured on the system. Each secure device alias defines one independent set of security attributes that governs that physical device. These security attributes are defined in the `DDB_SEC` file. The secure device alias can also define device special files that map to it and non-security attributes.

logical device alias

This is the name of a logical device that defines only non-security attributes and maps to a secure device alias that defines the security attributes. A logical device alias does not define any security attributes of its own. Therefore, many logical aliases may define different sets of non-security attributes, yet share only one set of security attributes defined by the secure device alias to which they map. A logical device alias can also define device special files that map to it.

Almost all aliases defined in the DDB are secure device aliases, that define only one set of security and one set of non-security attributes. However, you may define additional logical device aliases that have their own non-security attributes but share the security attributes defined for the secure device alias to which they map.

Device Attributes

The DDB (described in *"Maintaining the Device Database")* contains entries for device aliases. Each entry provides values for a set of attributes for the device represented by a particular alias. See the section *"Creating a Device Entry"* in the "Managing Storage Devices" chapter in volume 2 of "*System Administration*" for a description of all possible attributes.

The device aliases and their attributes should be created by the device installation script, if written for Release 4 or a later. However, you may need to modify the security attributes of all device aliases, based on the security environment in which the system is installed. The DDB must be updated with the alias names and attributes of any new devices added to the system.

NOTE

By reserving a device, you do not allocate it for use; the device must be allocated with **`admalloc`** before it can be used.

Managing Device Attributes

Setting and Examining Non-Security Device Attributes

Non-security device attributes can be maintained through the use of the **`putdev(1M)`** and **`getdev(1M)`** commands. These commands allow you to create, delete, modify, and view entries in the device database. Guidelines on the use of these commands can be found in the *"Maintaining the Device Database"* section in the "Managing Storage Devices" chapter in volume 2 of *"System Administration*".

Setting Security Device Attributes

These attributes can be set through the use of the **putdev** command. Guidelines on the use of this command can be found in the *"Maintaining the Device Database"* section in the "Managing Storage Devices" chapter in volume 2 of *"System Administration*".

Examining Security Device Attributes

The device security attributes and the device use count can be examined with the **devstat** command.

Before You Begin

This action can be performed by an administrator in the OP, SOP, or SSO role.

Procedure

To view the security attributes associated with a device, perform the following step:

1. Use the **devstat(1M)** command. Type:

   `tfadmin devstat` **-z** *device_name*

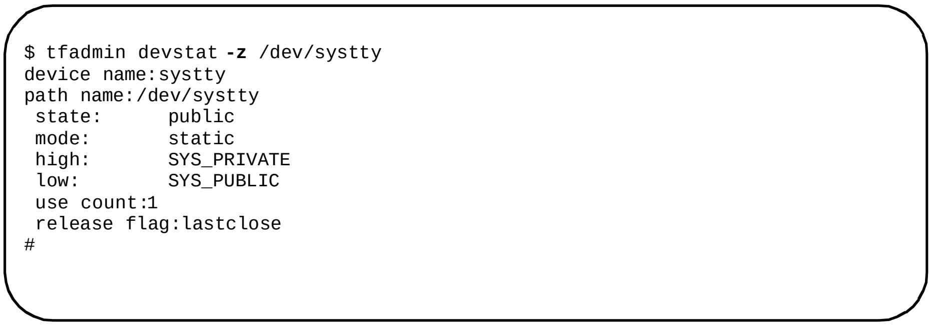
   or

   `tfadmin devstat` **-Z** *device_name*

   depending on whether you want the level alias names or the full level names, respectively, for the MAC levels. Omit the *device_file* to view the entire database. The use count is a flag (values `0` and `1`) that shows if the device is in use (file descriptor open or `mmap` active). A device can be specified as either an absolute pathname to a device special file or a device alias. (See *"Maintaining the Device Database"* in the "Managing Storage Devices" chapter in volume 2 of *"System Administration*". If a device alias is specified, then the current security attributes for all pathnames defined for the specified device are displayed.

Example

The following example shows the use of the **devstat** command.



```
$ tfadmin devstat -z /dev/systty
device name:systty
path name:/dev/systty
 state:      public
 mode:       static
 high:       SYS_PRIVATE
 low:        SYS_PUBLIC
 use count:1
 release flag:lastclose
#
```

Screen 15-1. Displaying Device Security Attributes

Device Allocation

Device allocation programs modify the device security attributes of one or more device special files to allow unprivileged access to one or more users. The device allocation programs check the DDB to determine or verify the security attributes and user ID for the device. The device allocation programs also check the device to determine what device special files need to be allocated for a specified logical or secure device alias.

The general purpose device allocation program is **`admalloc`**, which is used to allocate most devices for users. The **`login`** process allocates login terminals for users automatically during login.

admalloc

The **`admalloc`** command is used both to allocate and to deallocate devices. In its simplest invocation, **`admalloc`** allocates the specified device for the invoking user ID and group ID at the level of the invoking process and places the device in public state if the device attribute for `state` in the DDB is **`public`** or **`pubpriv`**. The following example shows the use of **`admalloc`** without any options to allocate a device.

```
tfadmin admalloc /dev/rsave
```

The device specified may be a secure device alias, a logical device alias, or a device special file.

The **`-m`** option of **`admalloc`** restricts the device state to **`private`**. In this state, only a process with the appropriate privilege has access to the device. After the following command is executed, the physical device represented by the alias `tty21` will be in the **`private`** state; unprivileged processes will not be able to access it.

```
tfadmin admalloc -m tty21
```

The **`-r`** option of **`admalloc`** sets the device range to the specified MAC security levels. This option takes, as an argument, two levels separated by a minus sign. The first level is the high level of the level range; the second, the low level of the range. The high level must dominate the low level, and the new range must be contained in the range specified in the DDB. The levels must be specified as either security level aliases or fully qualified level names. (For information on security levels, see Chapter 17, “Administering Mandatory Access Control and Multilevel Directories” in this book.)

In the following example, a tape device is allocated with a level range of `Confidential` (a level defined by the security administrator) to `SYS_PUBLIC` (a predefined level). This will allow the tape to be used to back up a file system that has `Confidential` as the highest level for data stored on it.

```
tfadmin admalloc -r Confidential-SYS_PUBLIC tapedrive1
```

NOTE

> When you allocate a device for trusted backup and restore, you need to ensure that the level range specified to the **`-r`** option of **`admalloc`** will include the levels of all files that will be backed up. If the level range is set incorrectly upon device allocation, the backup will fail. If you are backing up all files included on the distribution tape, you may need to allocate the tape device with a broad security range. If a device is allocated with a broad security level, you should ensure that the DAC settings on the device special file will prevent unauthorized users from accessing the device during any backup operations.
>
> If you set the range of a partition, that range must be within the range set for the entire disk.

The **`-w`** option of **`admalloc`** specifies the MAC security level to be set for the allocated device. The level must be within the device level range and is specified as either a level alias or a fully qualified level name. This option must be used with the **`-u`** option of **`admalloc`**, which specifies the user ID and group ID to be set for the allocated device. The argument to **`-u`** consists of a user name and a group name, separated by a comma. The user specified must have allocation permission specified in the DDB. The following example shows a terminal port being allocated to the user **`lp`** and the group **`lp`**. The port, which will be used to support a printer, is allocated at the **`Secret`** level (another level defined by the security administrator).

```
tfadmin admalloc -w Secret -u lp,lp /dev/term/tty027
```

After a device is allocated at one level for a specific user, you may want to change its level to allow another user to access the device. You can use the **`chlvl`** command to change the level of a device after it has been allocated. There may be some restrictions on changing a device level, however. For example, if a device is in **`public`** state, it can be changed only if it is not in use (that is, not open or mapped). The new level must be within the device level range allocated. This operation also requires appropriate privileges. [For a full list of restrictions governing the changing of the level of a device, see **`chlvl(1M)`**.]

The **`chlvl`** command takes two arguments. The first argument is a level, which can be either a level alias or a fully qualified level name. The second argument is either a file or a list of files, the security level of which will be set to the level given in the first argument to **chlvl**. The filename can be a block special device or character special device.

When you change the level of a device, none of its files may be opened or mapped, unless the device is in **dynamic** mode or the **private** state. The new level must also be within the device level range. You may need to change the mode of the device after allocation or a change of level.

The **-d** option of **admalloc** is used to deallocate a device. If the device still has open connections, the deallocation will fail unless the **-f** (force) option is also specified. When a device is deallocated, the device security attributes are set to allow only privileged access. The following command could be used to deallocate the terminal port allocated in the previous example. In this case, the **-f** option is not used, so the command will fail if the device has any open connections.

```
tfadmin admalloc -d /dev/term/tty027
```

If no device is specified with the **-d** option, **admalloc** attempts to deallocate all devices defined in the DDB.

CAUTION

> When the **-d** option is used with **-f** and no device argument is specified, all devices defined in the DDB can be deallocated.

The **admalloc** command is also used to allocate those devices that are configured to be allocated at system startup. The **-s** option will cause **admalloc** to allocate all devices with the startup attribute set to y in the DDB. This option cannot be combined with any other option. This option should be used only at system startup, in a script invoked by **/etc/rc2.**

Device Allocation during the Login Process

The process of logging in can be invoked only by using the Secure Attention Key (SAK).

The **login** process allocates the controlling terminal for the user in a trusted state and performs the identification and authentication checks, including the checks to ensure that the specified login level is within the device range in the DDB and that the user is authorized to use the device.

The device attributes are set as follows:

device special file level

The level of the device special file is set to the user's login level.

device special file DAC

The Discretionary Access Control settings on the device special file are set as follows: the owner of the file is set to the user, the group is set to the user's default group, and the permission bits on the file are set to allow read and write for the owner and no access for group and other (`rw-------`).

level range — The device level range used to allocate the device is set to the range defined for the device in the DDB.

state — The device state is set to **`public`**.

mode — The device mode is set according to the attribute in the DDB (which should always be **`static`**).

release flag — The release flag is set to **`lastclose`**.

When the last open connection to the login terminal is closed, the device is deallocated.

Special Cases

Some devices require special processing by the kernel or the driver for Mandatory Access Control. These devices include (but are not limited to):

- `null`
- `zero`
- `tty`
- `log devices`

Some multiplexing and cloning devices may also need special processing for support of multilevel security.

Guidelines for Multiplexing

The multiplexing device and the device being multiplexed are treated as separate devices. Applications performing the multiplexing are responsible for setting the correct labels on these devices. It is recommended that applications reserve such devices as **`private`** devices.

Guidelines for Clones

Each cloning driver must be examined to see if it supports multilevel cloning. The clone minor device created inside the kernel will have the same attributes as the cloneable file being opened. For a driver that supports multilevel cloning, you should follow these steps.

- You should create multiple cloning device special files, each servicing a range of minor devices.
- If the driver installation procedure creates device special files corresponding to minor devices, then these nodes should be created at the same level as the cloneable file that generates the minor devices with the same minor number as the clone files.
- Each cloneable file and the minor device special files corresponding to its clones should be defined in the same device entry in the DDB.

- When you allocate a cloneable file with **`admalloc`**, all other clone files are allocated at the same level.
- If unprivileged users must use cloneable files, you should allocate the entries at system startup so that unprivileged users can access the files as long as the users pass DAC and MAC access checks.
- It is recommended that administrators allocate these kinds of devices while the system is running.
- Before changing the label of a cloneable file, you should either deallocate the file with **`admalloc -d`** or use the `devstat` system call with the **`DEV_GET`** command to ensure that no one has any device open.

File Systems and Storage Devices

With the Enhanced Security Utilities installed, you need to be aware of how these utilities impact file systems and storage devices such as hard disks and tape drives. This section covers how the protection mechanisms apply to these devices.

Storage Devices

The guidelines that apply to the use of storage devices follow:

- The device must first be allocated before any data transfer can occur. This is done using the **`admalloc`** command, which is described in *"General Device File Protection"* in this chapter.
- The level range of the device must encompass the MAC label(s) of the files that are being stored or accessed. This is important because a properly set level range provides additional protection against improper data transfers. For example, a tape drive to be used for backing up user data only would be given a level range that encompasses only the MAC labels at which such data resides. If you try to include a sensitive system file, labelled at `SYS_PRIVATE`, which is a system level, not a user level, in the backup, the file will not be written because it's label is not within the current device level range of the tape drive device.

Example: Allocating a Floppy Drive to A User

For example, given a normal, unprivileged user, called *imuser*, logged in at `USER_LOGIN`, with files on a floppy drive that the user wishes to import at `USER_LOGIN`, you (if you are assigned the OP, SOP, or SSO role) can use the admalloc command to allocate the device to the user. Suppose the user is part of group *mgmt*. In this case, you would allocate the device, **`/dev/rdsk/om,`** with the following:

```
tfadmin admalloc -w USER_LOGIN -u imuser,mgmt /dev/rdsk/
f03ht
```

This assumes the user authorization list for this device, stored in the DDB, is set to grant permission to *imuser*. If not, you can override the user authorization list with the **-o** option:

```
tfadmin admalloc -o -w USER_LOGIN -u imuser,mgmt /dev/
rdsk/f03ht
```

Once the device is allocated, it should be set up for single level use at the user's level, with the owning user being *imuser* and the owning group being *mgmt*, with read and write access for *imuser* only.

Once the user has finished using the device, you can deallocate the tape drive:

```
tfadmin admalloc -d /dev/rdsk/f03ht
```

Mounting File Systems

File systems are a special case of storage device. In addition to the level range of the device on which the file system resides (for example, **/dev/rdsk/0m)**, each file system has its own *file system level range*. This is very similar to the level range of the device. When a file system is mounted using the **mount(1M)** command, a check is made to verify that the device level range encompasses the file system level range. If it does, the mounting attempt continues. If not, it fails.

NOTE

Some file system types do not support MAC. If a file system of a type that does not support MAC is mounted on a directory or mount point in a file system that does support MAC, all of the files on that file system are considered to have the level of the directory or mount point on which it was mounted. For example, suppose the root file system supports MAC, and you have a second file system that does not support MAC that you wish to mount on a directory called **/apps.** Suppose **/apps** has a level of `SYS_PUBLIC`. All of the files in this second file system will be handled as if they had a label of `SYS_PUBLIC` once the file system is mounted.

Terminal Devices

The inclusion of the Enhanced Security Utilities provides some additional security features that are useful for terminal devices:

- the device level range for the terminal device constrains the levels at which users can login to the system from the terminal. If the level at which a user is trying to log in is outside the level range of the terminal, the login attempt will fail. You can use this feature to, for example, preclude administrators from logging in at a terminal in a public area, or prevent normal users from logging in to the console. Another possible use might be to allocate groups of terminals to groups of users at different levels

- the system supports a Trusted Path mechanism, with a Secure Attention Key (SAK). This requires the user to enter the SAK to receive a login prompt, and ensures that the user is communicating with the system when entering their login name and password, rather than a “Trojan horse” program trying to acquire passwords. The rest of this section describes this feature in more detail.

The Secure Attention Key

NOTE

Note that this section describes a trusted path mechanism and the use of a Secure Attention Key to enable it. This mechanism is only applicable to terminals attached to serial ports. This mechanism is not supported for the console terminal or for windowing terminals

On many computer systems, a user's password is vulnerable whenever it is typed. Because passwords are typed over ordinary, insecure data channels, malicious users can steal passwords by using a spoofing program. A spoofing program is a program created by a malicious user of the system to trick other users into believing it is the system's login program. When the unsuspecting users enter their passwords, the spoofing program records them and passes them on to the user who wrote the program. With those stolen passwords, that user gains access to other users' accounts and files.

To protect passwords, an operating system with the Enhanced Security Utilities installed uses a secure communications channel, called a trusted path, whenever users enter their passwords. The trusted path prevents malicious users from employing spoofing programs or other devices to gain users' passwords. A user can get a login prompt only after pressing the Secure Attention Key (SAK) to establish a trusted path between the user's terminal and the main computer. The trusted path is in place only during login processing and is replaced by the normal data channel after login is complete. Any time users want to log on, they must press the SAK and obtain a new trusted path. (Note users can change their password only during a login session.)

As an administrator, you should ensure a SAK is defined for all terminals used for logins. If a SAK is not defined for a terminal, the terminal is disabled and cannot be used for logins. The SAK can be either a line condition, such as line drop or break, or a control character.

As distributed, the SAK for all terminals configured on the system is undefined. You can define a different SAK with the **`defsak`** command.

NOTE

> You should ensure all users at your site know what the SAK is for their terminals. If users guess at the SAK or rely on information from other users, their passwords are vulnerable to malicious users. You should ensure information about the SAK is distributed in a way that cannot be forged or distorted by malicious users.

As an administrator, you can also disable the trusted path for a specific terminal. The terminal can then be used for logins without the SAK being entered, and communications programs, such as **`uucp`**, can use the system.

NOTE

> Trusted path processing is not guaranteed to work for smart terminals (microcomputers used as terminals). Smart terminals might be programmed so entering the SAK sends a different signal to the host computer. This interferes with SAK recognition and makes the user of a smart terminal vulnerable to a spoofing program residing on the terminal itself.

Using defsak to Administer the SAK

The shell-level command **`defsak`** provides the preferred interface for administering the SAK. The **`defsak`** command can define the SAK for terminals, display the defined SAK, disable the trusted path for a terminal, and remove the defined SAK for a terminal. The following sections describe how to use **`defsak`** to perform these tasks.

Defining the SAK for Terminals

To define the SAK for a terminal, use the **`-d`** option of **`defsak`**.

Specifying the SAK

The **`-d`** option of **`defsak`** takes two types of arguments, depending on the type of SAK to be defined.

If the SAK is a line condition SAK, such as the break or line drop signals, the argument to **`-d`** is a string. Use the string `drop` to define the SAK as line drop, and the string `break` to define the SAK as the break key.

If the SAK is a control character, the argument to **`-d`** can be either an octal value between 000 and 015, 020 and 037, or a character preceded by a caret.

You also need to specify the full path name of the terminal for which the SAK is being defined.

Before You Begin

This task can only be performed by an administrator in the SSO role.

Procedure

To define a primary SAK, perform the following step:

1. Type:

   `tfadmin defsak` **`-d`** *SAK terminal_name*

Example

For example, to define the break key as the SAK for terminal **`/dev/term/22,`** enter the following command:

```
tfadmin defsak -d break /dev/term/22
```

CAUTION

It is possible for different pathnames to refer to the same terminal (that is, the same physical device). If this happens, SAK processing for the terminal may not occur as users expect unless each SAK definition is identical. You should avoid having different pathnames for the same device.

Defining a Secondary SAK

You can also define the line drop as a secondary SAK with the **`-x`** option of **`defsak`**. The line drop is the only recommended secondary SAK.

Before You Begin

This task can only be performed by an administrator in the SSO role.

Procedure

To define line drop as the secondary SAK, perform the following steps:

1. Include the **`-x`** option when calling **`defsak`** to define the primary SAK.

   `Enter the following command:`

   `tfadmin defsak` **`-d`** *SAK* **`-x`** *terminal_name*

Example

For example, if you want to have line drop be a secondary SAK for terminal **`/dev/term/22,`** enter the following command:

```
tfadmin defsak -d break -x /dev/term/22
```

Unless there is a good reason not to (for example, the terminal cannot generate a line drop easily), it is recommended you always define the line drop as a secondary SAK.

NOTE

> If you redefine the SAK, the new definition does not take effect until the next login session on the terminal. If a user is logged on when you define the SAK for a terminal, the SAK does not change until that user logs out. If a user wants to log out by entering the SAK, that user must enter the SAK used to get the login prompt at the start of the login session. We advise changing SAK definitions for a terminal only when users are not logged on.

Displaying the Defined SAK

If you invoke **`defsak`** without any options or arguments, it displays the defined SAKs for all terminals. If you invoke **`defsak`** with the path name of a terminal as an argument, it displays the defined SAK for only that terminal.

Before You Begin

To guarantee that the SAK can be displayed, you must be an administrative user in the SSO role.

Procedure

To display the defined SAK, perform the following step:

1. Type:

   ```
   tfadmin defsak terminal_name
   ```

   Omit *terminal_name* to print all of the defined SAK's. If line drop is defined as a secondary SAK, the string `+drop` will be displayed for the terminal next to the primary SAK.

Example

If you want to verify the SAK for **`/dev/term/22`** is defined correctly, enter the following command:

```
tfadmin defsak /dev/term/22
```

The system will respond with:

```
/dev/term/22:break +drop
```

The **`defsak`** command prints the path of the terminal (in this case, **`/dev/term/22),`** along with the defined SAK (`break`). The string `+drop` is present only if line drop is defined as a secondary SAK.

Disabling the Trusted Path

In some cases, you may need to disable trusted path processing for a terminal to allow communications utilities to function correctly.

CAUTION

> You should not disable the trusted path unless a communication utility requires it. You should ensure users cannot access any terminal that has the SAK disabled. You should never use this feature to disable the trusted path for users' terminals. Doing so seriously weakens system security, because it make the users' passwords vulnerable.

Before You Begin

To perform this action, you must be an administrative user in the SSO role.

Procedure

To disable Trusted Path for a specific terminal, perform the following step:

1. Give the argument **none** to the **`-d`** option of **`defsak`**. Type:

   ```
   tfadmin defsak -d none terminal_name
   ```

2. The system will print a warning message:

   ```
   SAK disabled for terminal terminal_name,
   terminal is no longer secure.
   ```

Example

For example, to disable the trusted path for **`/dev/term/11,`** type the following command:

```
tfadmin defsak -d none /dev/term/11
```

Removing the SAK for a Terminal

It is possible to remove the SAK definition for a terminal. When the SAK definition for a terminal is removed, the terminal is disabled and cannot be used for logins.

Before You Begin

To perform this action, you must be an administrative user in the SSO role.

Procedure

To remove the SAK for a terminal, perform the following step:

1. Use the **-r** option of the defsak command.Type:

   `tfadmin defsak` **`-r`** *terminal_name*

Example

For example, if you want to remove the SAK definition for terminal **`/dev/term/22,`** enter the following command:

```
tfadmin defsak -r /dev/term/22
```

After you enter this command, no one can log on from this terminal.

Differences Between Removing the SAK and Disabling Trusted Path

Removing the SAK definition is not the same as defining the SAK as **none**. Removing the SAK definition disables the terminal; no logins can take place on the terminal until the SAK is defined again. Defining the SAK as **none** disables trusted path processing and allows logins to take place without use of the SAK. The following table illustrates the difference.

Table 15-1. Trusted Path States

Trusted Path State	**defsak** Command	Terminal Status
Trusted Path active	**defsak -d**	Terminal available for logins; SAK must be entered.
Trusted Path disabled	**defsak -d** none	Terminal available for logins but not secure; SAK not required for login.
SAK not defined	**defsak -r**	Terminal disabled and unavailable for logins.

Guidelines for Choosing a SAK

Because the system will end a user's login session whenever it sees the SAK as input, the SAK should not be a character users will normally type. Use of the line drop as the SAK is strongly recommended.

All terminals at a site should have the same SAK, if possible. This makes it easier for users to remember the SAK and simplifies system administration.

Using a control character as the SAK is discouraged. A control character should be used only if it is not possible to use the line drop or break signals as the SAK. Using a control character as the SAK has the following problems:

- A control character SAK restricts the setting of terminal characteristics, and it may be difficult to find a character not used by application programs and commands.
- Control character SAKs may not work well in environments, such as terminal-based windowing packages, where data messages are wrapped by protocol information. Protocol information may contain the SAK, in which case the user will be logged out immediately.

If you choose a character SAK, do not use any character in the set of default settings for special characters defined in the **termio(7)** online manual page. Doing so will cause `ioctl` failures when the tty device's `termio` characteristics are being set. Also, choosing one of the following as a SAK is strongly discouraged:

Character	Octal Value
back space	010
horizontal tab	011
new line	012
vertical tab	013

Character	Octal Value
new page	014
carriage return	015
<CTRL><d>	004
<CTRL><s>	023
<CTRL><q>	021

Re-definition of the SAK is discouraged. Redefining the SAK has no security benefits, and it can lead to problems for users.

Changes to ttymon and ttyadm

The **defsak** command uses **ttymon** and **ttyadm** to control SAK information in **ttymon's _pmtab** database files. This section explains these changes to give you a better understanding of trusted path processing.

You should not need to use **ttyadm** to administer the SAK. It is recommended you use **defsak** instead of **ttyadm**, because **defsak** provides independence from the underlying database structure.

The **ttyadm** command has the following relevant options:

-k This option specifies the type of SAK representative. It takes one of four arguments:

- c The SAK is a control character.
- l The SAK is a line condition (break or drop).
- n The SAK is **none**, which disables trusted path processing on the terminal and allows users or programs to log on without entering a SAK.
- x The SAK is undefined; that is, the SAK definition is removed from the database. This disables the terminal. No one will be able to log on from that terminal until a new SAK is defined.

-K This option specifies the SAK. If the SAK is a control character, the argument to **-K** must be either an octal number in the range 000 to 015, 020 to 037, or a character preceded by a caret, such as ^A. If the SAK is a line condition, the argument to **-K** is either **drop** (line drop), or **break**.

-x This option specifies line drop as the secondary SAK, in addition to the one specified by **-K** .

The same three options are accepted by the **ttymon** command.

For more information, see the **ttyadm(1M)** and **ttymon(1M)** online manual pages.

Administering Printers

The Enhanced Security Utilities provides increased security for printers. These changes are as follows:

- a separate interface, called B2 is provided. This interface provides for the printing of the MAC label on each page of the output, providing an indication of the output's sensitivity along with the output itself. This interface also forces a banner page for each job, regardless of the forms setting for the printer. The MAC label will also be printed on the banner page.
- The level range of the printer restricts the level of the jobs that may be printed. Jobs containing files with labels outside the level range will be rejected.

See the “Advanced Print Service” chapter in volume 2 of “*System Administration*” for information on setting interfaces and forms for a printer.

Connecting New Devices

When a new device is installed on a system running with the Enhanced Security Utilities, or you need to create a new device file, you should check the security attributes of the device file to ensure that system security is preserved. Several guidelines apply:

- Verify the entry in the Device Database is correct.
- Make sure the permission bits, ACL (if present), group, and owner are appropriate. When the device is not allocated, the owner should generally be one of the special administrative logins listed in Chapter 14, “User Accounts and Group Management”.
- Make sure the MAC level assigned to the file is appropriate. Generally, this should be set to `SYS_PRIVATE`. Some pseudo-devices, similar to **`/dev/null,`** may be exceptions.
- Verify the level range for the device is appropriate. Factors include expected use of the device, and the environment. For example, the level range on a device file for a new public printer would probably only include user data levels in its level range.
- verify the state, mode and flags are appropriate for the device. Generally, the mode should be **`static`**, unless the device represents a multi-windowed terminal. A device that is intended for use by privileged users only should probably have state **`private`** and **`persistent`** or **`lastclose`** release flags set.

16
File Protection

16
File Protection

Introduction

Because the operating system is a multi-user system, you usually do not work alone in the file system. System users can follow pathnames to various directories and read and use files belonging to one another, as long as they have permission to do so.

This chapter discusses file attributes that have security relevance. Attributes specific to device files are discussed in Chapter 15, “Administering Devices, Terminals and Printers” in this book. Protection attributes common to all file types include:

- owner — the file owner
- group — users who are part of the group may have special access permissions
- discretionary access control — these are mechanisms set by the file owner to determine who may access the file. Two mechanisms supported by this release are permission bits and Access Control Lists (ACLs). ACLs are available only if the Access Control List Utilities have been installed.
- Mandatory Access Control (MAC) — an access control mechanism that does not depend on the actions of the owner. It uses comparisons of security levels set by the system to determine access. MAC is covered in detail in the Chapter 17, “Administering Mandatory Access Control and Multilevel Directories” in this book.

Other file attributes with security relevance include:

- Set-UID and Set-GID bits — these bits, when set on an executable file, give the user's process that is executing the file the identity of the owner (or group) of the executable.
- Privileges — the system recognizes a distinct set of privileges. Each privilege allows the possessor to override a specific system restriction. Privileges are described more fully in Chapter 9, “Administering Privileges” and Chapter 10, “Trusted Facility Management” of this book.

File attributes not treated elsewhere are described in this chapter, along with their security relevance.

Owner

Each file on the system has an owner. Generally, the creator of the file is also the owner. However, a file owner can generally reassign ownership rights using the **`chown(1)`** command.

If you own a file, you can decide who has the right to read it, write it (make changes to it), or, if it is a program, to execute it.You can also restrict permissions for directories. When you grant execute permission for a directory, you allow the specified users to change directory to it and list its contents with the **`ls(1)`** command. Only the owner or a privileged user can define the following:

- which users have permission to access data
- which types of permission they have (that is, how they are allowed to use the data)

Because this type of access control is mediated at the discretion of the owner of the file, it is called Discretionary Access Control (DAC). Systems running with the Enhanced Security Utilities installed may also use a second type of access control, called Mandatory Access Control (MAC). When both types are in use, access requests must pass both DAC and MAC checks to be granted. This means that the owner no longer has sole control over file access.

Group

Each file is also assigned to a particular group. A group is a collection of users. Each user may be assigned to one or more groups. The users in a file's group may have special DAC permissions set.

Discretionary Access Control: Permission Bits

In the first field of the **`ls -l`** output, the first character indicates the type of file (- for a regular file, `d` for a directory file, `b` for a block special device file, `c` for a character special device file, and `l` for a symbolic link). The next nine characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions, the next set refers to the permissions of the file's group class (this will consist only of the owning group, unless additional ACL entries are present), and the last set to all others. Within each set, the three characters show permission to read, to write, and to execute the file as a program, respectively. For a directory, “execute” permission is interpreted to mean permission to search the directory for a specified file.

One additional character may appear at the end of the permission bit characters. A plus sign (`+`) is displayed to show that additional access permissions, beyond those shown by the three sets of three bits, have been granted or denied through the Access Control List (ACL) mechanism. ACLs and their relation to permission bits are discussed in detail in the

"Protecting Your Files" section of the "Using the File System" chapter of the *User's Guide.*

The permissions are as follows:

Permissions	
Explanation	Symbol
The file is readable.	r
The file is writable.	w
The file is executable.	x
This permission is *not* granted.	—
Mandatory locking will occur during access. (The set-group-ID bit is on and the group execution bit is off.)	l
The set-user-ID or set-group-ID bit is on, and the corresponding user or group execution bit is also on.	s
The set-user-ID bit is on and the use rexecution bit is off.	S
The sticky and the execution bits for `other` are on.	t
The sticky bit is turned on, and the execution bit for `other` is off.	T

Figure 16-1. File Access Permissions

Permissions	
Explanation	Symbol
The directory is readable.	r
The directory may be altered (files may be added or removed).	w
The directory may be searched. (This permission is required to **`cd`** to the directory.)	x
File removal from a writable directory is limited to the owner of the directory or file unless the file is writable.	t

Figure 16-2. Directory Access Permissions

For more information, refer to the **`ls(1)`**, **`getacl(1)`**, **`chmod(1)`**, and **`setacl(1)`** online manual pages. For a complete description of using ACLs to control access to files, see the "Using the File System" chapter of the *User's Guide.*

Discretionary Access Control: Access Control Lists

Access Control Lists (ACLs) give you a more precise way to control access to files. The ACL contains one-line entries naming specific users and groups and indicating the access is granted to each. The presence of an ACL also changes the meaning of the “group” permission bits displayed using the **`ls-l`** command.

There are always at least four entries in an ACL, a **`user`** entry, a **`group`** entry, a **`class`** entry, and an **`other`** entry. When an ACL contains only four entries, the permissions it grants are exactly the same as the permissions represented by the permission bits.

Additional entries are used to specify permissions for users and groups other than the owning user and group for the file.You can also specify *default* ACL entries. These entries are only used in directories. By specifying default ACL entries for a directory, a file created in that directory will have those default entries incorporated into its ACL.

ACLs are set and displayed using the **`getacl(1)`**, **`setacl(1)`**, and **`chmod(1)`** commands. Setting and viewing ACLs are discussed in the *“Protecting Your Files”* section of the “Using the File System” chapter of the *User's Guide.*

Set-UID and Set-GID

The set-user identification (set-UID) and set-group identification (set-GID) bits must be used carefully. These bits are set through the **`chmod(1)`** command and can be specified for any executable file. When any user runs an executable file that has either of these bits set, the system gives the user the permissions of the owner (or group) of the executable. For this reason it's a good idea to verify, occasionally, that the protection of files has not been compromised. Procedures for doing this can be found in Chapte r19, “Security Procedures” of this book.

17
Administering Mandatory Access Control and Multi-level Directories

17
Administering Mandatory Access Control and Multilevel Directories

Mandatory Access Control

The Mandatory Access Control (MAC) mechanism enforces access restrictions that do not depend on the actions of the user. MAC is based on the comparison of security levels that are assigned to users, processes, files, and other objects. MAC supplements DAC (see Discretionary Access Control: Permission Bits in Chapter 16) to prevent accidental or malicious disclosure of sensitive information.

A user's security level limits the user's ability to read and change information. These limits are enforced by the Trusted Computing Base (TCB). A user may be assigned a single security level or several, but the user is restricted to one level for a login session.

The *"Mandatory Access Control"* section of the "Managing Files Securely" chapter of the *User's Guide* describes how the MAC mechanism restricts user access to files (and other objects), and describes classifications, categories, levels, level aliases, the dominance relationship, and level equality. You should be familiar with the MAC information in the *User's Guide* before reading this section.

This section describes the administrative duties of defining levels, disabling levels, examining current level definitions, assigning levels to users, and changing the levels of files.

CAUTION

> It is extremely important that you administer the security level definitions and user login level authorizations with care. An error in the administration of the MAC assignments could compromise the security of the system.

Classifications, Categories, Levels, and Aliases

A classification, category, or level alias must be named before it can be used. A level must have a level identifier (LID) assigned before it can be used. A LID is a number the system uses to identify a level.

The values associated with classifications and categories, and the definitions of security levels and level aliases are stored in files in the **/etc/security/mac** directory.

ltf.class	definitions of classifications
ltf.cat	definitions of categories
ltf.alias	level aliases
lid.internal	level definitions

CAUTION

> Do not directly edit these files to change the names of classifications, categories, aliases or level identifiers. Using a text editor to change the security level translation files will corrupt the format of the files and cause level validations and level comparisons to produce faulty results. Processes may be granted access to files at inappropriate levels, and access at some levels may be blocked. This could lead to serious system problems if the levels of system files are affected.

A system administration menu is available for maintaining the names of classifications, categories, and aliases, and for assigning level identifiers. To use the menu, enter `sysadm security` and select **levels.** If you do not want to use the menu interface to maintain the names, you may use the **lvlname** and **lvldelete** commands.

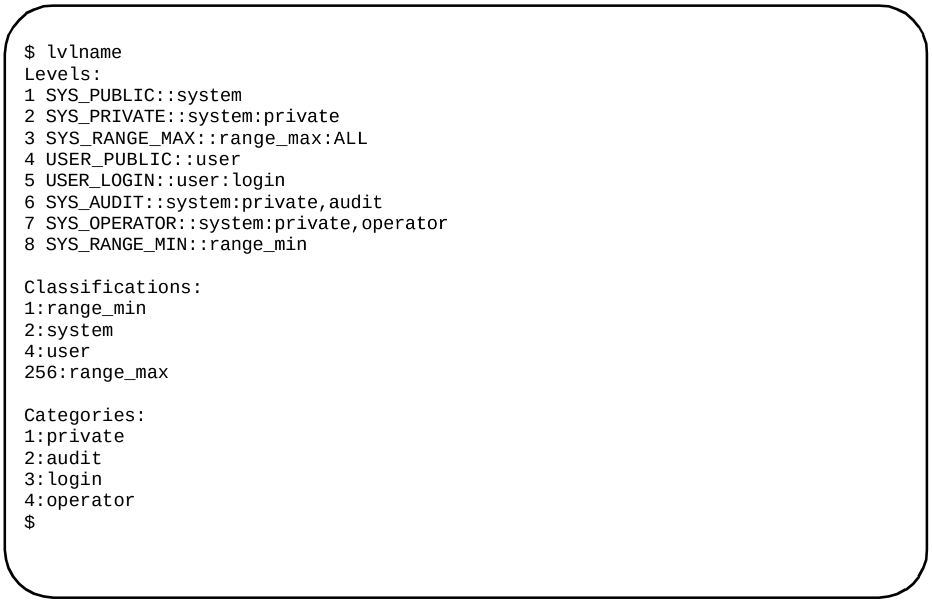
CAUTION

> The system administration menus are available only in single-user mode. In general, creation or deletion of MAC classifications, categories, levels and aliases should only be performed in single-user mode. The system is initially configured upon installation to only allow such actions in single-user mode. Changing the MAC databases while in multi-user mode will create indeterminate and possibly undesirable results.

Predefined Classifications, Categories, Levels, and Aliases

Four classifications, four categories, and eight levels (with aliases) are predefined. The **lvlname** command without arguments or options will list all the defined levels.

An asterisk after a level name as printed by **lvlname** shows that the level is inactive, but may still be in use on the system. A complete description of the **lvlname** output format can be found in the **lvlname**(1M) online manual page. Screen 17-1 shows the predefined classifications, categories, levels and aliases as printed by the **lvlname** command on a newly installed system.



```
$ lvlname
Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min

Classifications:
1:range_min
2:system
4:user
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
$
```

Screen 17-1. Predefined Classifications, Categories, Levels, and Aliases

The pseudo-category, `ALL`, can be used in level definitions. A level defined with the category `ALL` includes not only all of the categories currently defined on the system, but any additional categories which may be defined in the future.

Restrictions on Classification, Categories, Levels, and Aliases

Each classification, category, and alias name must be unique. You must not assign the same name to a classification and a category, an alias and a category, or an alias and a classification.

There are restrictions on the characters that can be used in identifier names. Also, a small set of LIDs, classification numbers, category numbers and names are reserved for system use. The name restrictions and reserved identifiers are listed in the **`lvlname(1M)`** online manual page.

The **`-r`** option to the **`lvlname`** command can be used to override the restrictions on the assignment of reserved identifiers.

CAUTION

When the system is delivered, system files are set to levels defined by some of the reserved identifiers. Other reserved identifiers may be used in future releases. Overriding the restrictions on the assignment of reserved identifiers may undermine MAC controls on the system files.

LID States

A level identifier (LID) can be in one of three states: invalid, valid-active, and valid-inactive.

A LID which has never been assigned a level name is invalid. An invalid level cannot be assigned to a file or assigned to a user as a login level, and cannot be compared to another level.

Once a level is assigned using the **lvlname** command, the LID becomes valid-active. Users can log on only at valid-active levels.

Once a LID has been defined on your system, it is always valid for level comparisons. However, you may make the level inactive with the **lvldelete** command.

When a level is inactive, the level's name can no longer be used. This prevents users from logging in at the deactivated level. For commands that take a level as an argument, inactive levels must be specified by LID. Except for **lvlname,** commands that display levels will display the LID rather than the name or alias for inactive levels. Deactivating a level with **lvldelete** also deletes the alias, if any.

The **lvldelete** command does not affect users already logged on at the specified level. If you want to be sure no user processes at that level remain on the system and no files of that level remain open, you must reboot after the security level is set to inactive.

The system still uses the level for level comparisons, so files (and other objects) at the inactive level are still readable by the appropriate users. Since users can no longer log on at the inactive level, only privileged users are able to write to objects existing at that level.

Adding Category and Classification Names

You will need to define additional levels and level aliases to make best use of the MAC mechanism. The first step in defining the levels that will be used on your system is naming the classifications and categories needed for the level definitions.

NOTE

> Although this section cannot anticipate the requirements of your site, the examples throughout this section may provide you with some idea of how the MAC mechanism can best be applied to your system.

The **lvlname** command is used to name classifications and categories. The numerical values of classifications and categories do not have to be sequential. Because classifications are hierarchical, it is good practice to leave gaps between values for classification numbers in case you need to add intermediate classifications later. Categories are not hierarchical, so there is no need to leave gaps between the numerical values, but you may do so.

You should determine what the values and names of classifications and categories are going to be before you start naming them. Laying out the names beforehand will prevent problems that may force you to repeat previous work.

NOTE

If you want to transport file systems or other data with embedded security labels to another system, the other system must have identical classification, category, and alias names. The systems must also have identical level identifiers (LIDs). Otherwise, imported levels will have different meanings than they had on the exporting system.

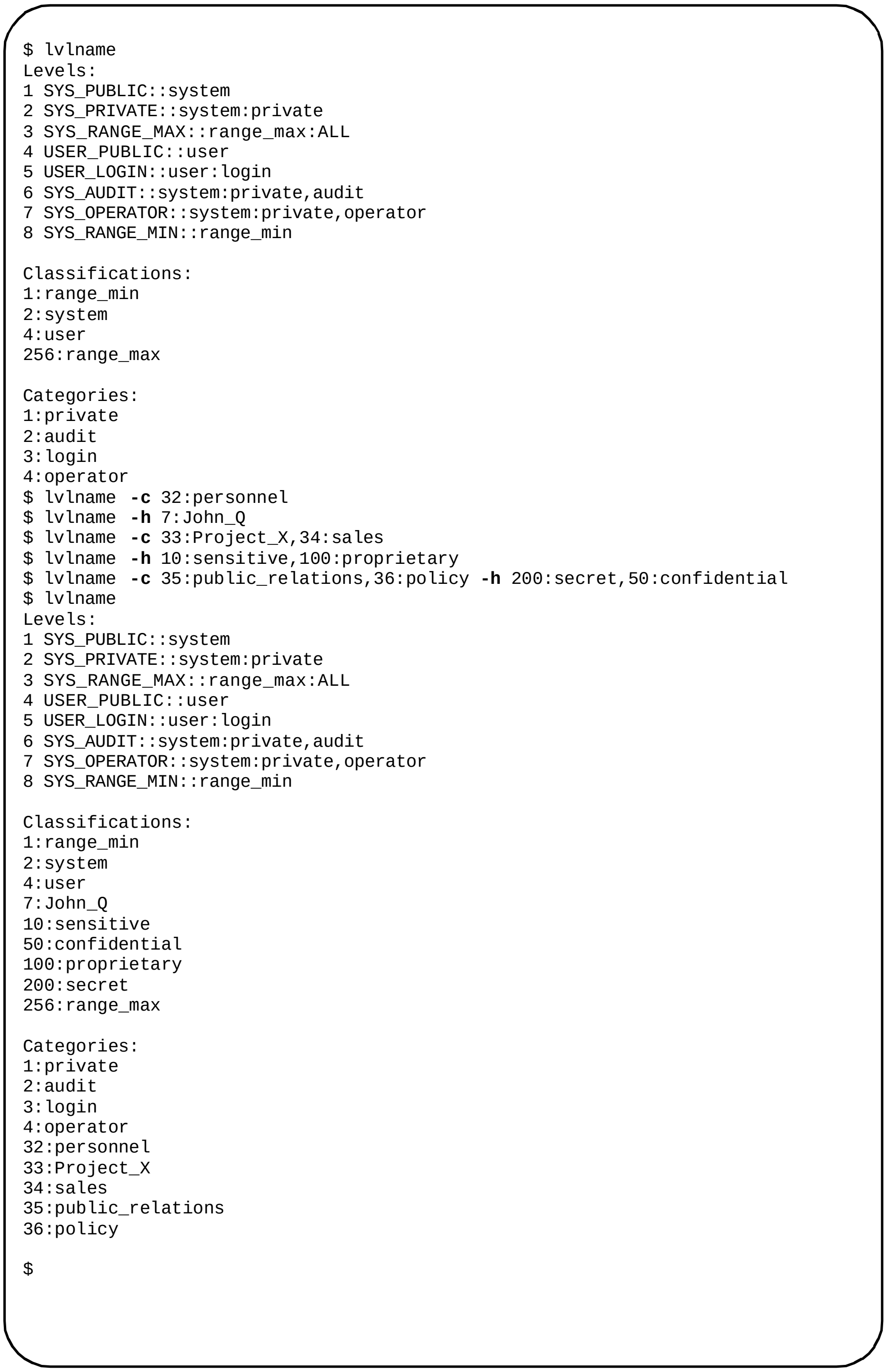
CAUTION

The names you select for classifications, categories, and aliases should be unclassified, since the **ltf.*** files are installed at SYS_PUBLIC and are publicly readable. If you must use classified names, you must set the level of the **ltf.*** files to SYS_PRIVATE to prevent unauthorized disclosure of these names. Users will still, however, be able to display the classification, category, and alias names of files dominated by the level of the current process.

The level translation and history log files are:

- **/etc/security/mac/ltf.cat**
- **/etc/security/mac/ltf.class**
- **/etc/security/mac/ltf.alias**
- **/etc/security/mac/lid.internal**
- **/etc/security/mac/hist.class.add**
- **/etc/security/mac/hist.cat.add**
- **/etc/security/mac/hist.alias.add**
- **/etc/security/mac/hist.lid.add**
- **/etc/security/mac/hist.class.del**
- **/etc/security/mac/hist.cat.del**
- **/etc/security/mac/hist.alias.del**
- **/etc/security/mac/hist.lid.del**

The **-c** option to **lvlname** is used to define a new category, and the **-h** option to **lvlname** is used to define a new classification (the h stands for hierarchical). Multiple classifications and categories can be defined by one **lvlname** command. Screen 17-2 shows an example of defining classifications and categories on a newly installed system.



```
$ lvlname
Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min

Classifications:
1:range_min
2:system
4:user
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
$ lvlname -c 32:personnel
$ lvlname -h 7:John_Q
$ lvlname -c 33:Project_X,34:sales
$ lvlname -h 10:sensitive,100:proprietary
$ lvlname -c 35:public_relations,36:policy -h 200:secret,50:confidential
$ lvlname
Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min

Classifications:
1:range_min
2:system
4:user
7:John_Q
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy

$
```

Screen 17-2. Adding Categories and Classifications

In the above example, the **lvlname** command is first used without options to display the current MAC definitions. Then the definition for a new category, **personnel**, is added. The example next defines a new classification, **John_Q**, which has a low hierarchical value. and continues with categories **Project_X** and **sales**, classifications

sensitive and **proprietary**, categories **public_relations** and **policy**, and classifications **secret** and **confidential**. Finally, the administrator reviews the MAC definitions.

Adding a New Classification

The **-h** option to **lvlname** is used to define a new classification (the h stands for hierarchical). Multiple classifications can be defined by one **lvlname** command.

Before You Begin

You must choose a name and number for the new classification that meets the naming criteria found on the **lvlname(1M)** online manual page. If not, you must use the **-r** option to override this restriction. Remember if you want to transfer data from this stem to another easily, the classifications on the two systems must be identical. You must be assigned to the SSO role to perform this task.

Procedure

To add a new MAC classification, perform the following steps:

1. Type:

   `tfadmin lvlname` **`-h`** *number*`:`*name*

   where *number* and *name* are the number and name you have chosen for the new category.

2. Verify the new classification was correctly added. Invoke the **lvlname** command and check the list of category is listed. Type:

   ```
   lvlname
   ```

Example of Adding a New Classification

Suppose we want to add a new classification. We choose a classification name of *confidential* and a classification number of 50. We invoke **lvlname-h** to add the new classification, and then invoke **lvlname** again to verify the classification was correctly added. Screen 17-2 shows an example of this process.

Adding a New Category

The **-c** option to **lvlname** is used to define a new category. Multiple categories can be defined by one **lvlname** command.

```
$ tfadmin lvlname -h 50:confidential
$ lvlname
Levels:

Classifications:
1:range_min
2:system
4:user
50:confidential

Categories:

$
```

Screen 17-3. Adding a New Classification

Before You Begin

You must choose a name and number for the new category that meets the naming criteria found on the **`lvlname(1M)`** online manual page. If not, you must use the **`-r`** option to override this restriction, although use of reserved names and numbers means that future compatibility cannot be guaranteed. Remember if you want to transfer data from this stem to another easily, the categories on the two systems must be identical. You must be assigned to the SSO role to perform this action.

Procedure

To add a new MAC category, perform the following steps:

1. Type:

   `tfadmin lvlname` **`-c`** *number*`:`*name*

   where *number* and *name* are the number and name you have chosen for the new category.

2. Verify the new category was correctly added. Invoke the **`lvlname`** command and check the list of categories to verify the new category is listed. Type:

   `lvlname`

Example of Adding a New Category

Suppose we want to add a new category. We choose a category name of *personnel* and a category number of 32. We invoke **`lvlname-c`** to add the new category, and then invoke **`lvlname`** again to verify the category was correctly added. Refer to Screen 17-4.

```
$ tfadmin lvlname -c 32:personnel
$ lvlname
Levels:

Classifications:

Categories:
1:private
2:audit
3:login
4:operator
32:personnel

$
```

Screen 17-4. Adding a New Category

Example of Adding Multiple Classifications and Categories

You can add multiple classifications and categories at one time using the **`lvlname`** command. Screen 17-5 is an example of adding classifications of proprietary and secret, and categories of Project_X and Project_Y.

```
$ tfadmin lvlname -h 100:proprietary,200:secret -c 33:Project_X,34:Project_Y
$ lvlname
Levels:

Classifications:
100:proprietary
200:secret
256:range_max

Categories:
33:Project_X
34:Project_Y

$
```

Screen 17-5. Adding Multiple Categories and Classifications

Defining a New Security Level

Once you have defined the classifications and categories, you may combine them to define the security levels that will be used on your system.

Before You Begin

The classifications and categories you wish to use in the new level must already be defined. You should be in single-user mode when performing this procedure. If you want to transfer data between this system and another system easily, you must define the level identically on the two systems. In this case, you must specify the level identifier (LID) explicitly when creating the level. LID values less than 100 are reserved. You can override this by using the **`-r`** option, but compatibility with future releases cannot be guaranteed. Otherwise, the system will assign the next available LID to the new security level; that is, the new LID will be the highest LID yet defined, plus one.

Procedure

To add a new level, perform the following steps:

1. Use the **`-l`** option to the **`lvlname`** command. If you do not need to explicitly specify the level identifier (LID), enter:

   `lvlname -l` *classification_name*`:`*category1*`,`*category2*`...`

   Otherwise enter:

   `lvlname -l` *LID::classification_name:category1,category2...*

   The *classification_name* is the name of the chosen classification, and *category1,category2* is a comma-separated list of the desired category names. The identifier `ALL` may be used in place of the list of category names. The category list may be empty.

2. Invoke **`lvlname`** again to check that the level was properly defined.

Example of Adding a Level

Building on the categories and classifications added in previous examples in this chapter, we use the **`lvlname`** command to define several new levels, and then invoke the command again to verify that the new levels were properly created.

In the example in Screen 17-6, the administrator defines five security levels. In the first three definitions, the administrator allows the **`lvlname`** command to select the next available LID value (LIDs in the range 1-99 are reserved for system use only, so the next available one is 100).

In the fourth definition, the administrator explicitly assigns the LID value 200 to the level. When an additional level is defined using the default LID in the final definition, the default LID is 201, or one more than the highest LID assigned.

CAUTION

Since gaps in the level translation file can cause performance problems, it is generally better not to use explicit assignment. You may need to assign LIDs explicitly if you transfer labeled data between machines and need to maintain matching level databases.

```
# lvlname -l secret:Project_X
# lvlname -l confidential:personnel,sales
# lvlname -l secret:ALL
# lvlname -l 200::sensitive:public_relations
# lvlname -l proprietary:
# lvlname
Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min
100 secret:Project_X
101 confidential:personnel,sales
102 secret:ALL
200 sensitive:public_relations
201 proprietary

Classifications:
1:range_min
2:system
4:user
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy
#
```

Screen 17-6. Defining Security Levels

Adding an Alias Name

Since the fully qualified level name may be tedious to type if it contains many categories, a level alias name may be defined for any defined level.

Before You Begin

The level for which you wish to define an alias must already exist. You should be in single-user mode to perform this procedure.

Procedure

To define an alias name for a level, perform the following steps:

1. Use the **`-a`** option of **`lvlname`** to assign an alias. Enter:

   `lvlname` **`-a`** *alias::level_name*

   The *alias* is the name you wish to use as an alias for the level. The *level_name* identifies the level. It must be the fully qualified level name, complete with classification and any categories, rather than the LID.

2. Use the **`lvlname`** command to verify the alias is properly defined.

Example of Adding an Alias

Screen 17-7 shows an example of assigning level aliases, based on the levels defined in the examples earlier in this chapter.

```
# lvlname -a Development::secret:Project_X
# lvlname -a Management::secret:ALL
# lvlname -a Finance::confidential:personnel,sales
# lvlname
Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min
100 Development::secret:Project_X
101 Finance::confidential:personnel,sales
102 Management::secret:ALL
200 proprietary:

Classifications:
1:range_min
2:system
4:user
7:John_Q
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy
```

Screen 17-7. Adding a Level Alias

In the example in Screen 17-7, the administrator assigns the alias **`Development`** to the level **`secret:Project_X.`** The administrator then assigns two other aliases. There is no alias assigned to **`proprietary:`**

Removing Mandatory Access Control Definitions

Occasionally, you may find it necessary to remove some of the MAC definitions. The **`lvldelete`** command is used for this operation.

NOTE

> Only the Trusted System Programmer has access to the **`lvldelete`** command, which is available in single-user mode only. See Chapter 13, “Maintaining an Enhanced Security System” for a discussion of single-user mode.

Deactivating a Security Level

You can remove an existing security level from use.

Before You Begin

You should be in single-user mode to perform this action.

Procedure

To deactivate a security level,

1. Enter:

   `lvldelete` **`-l`** *LID*

   or

   `lvldelete` **`-f`** *full_level_name*

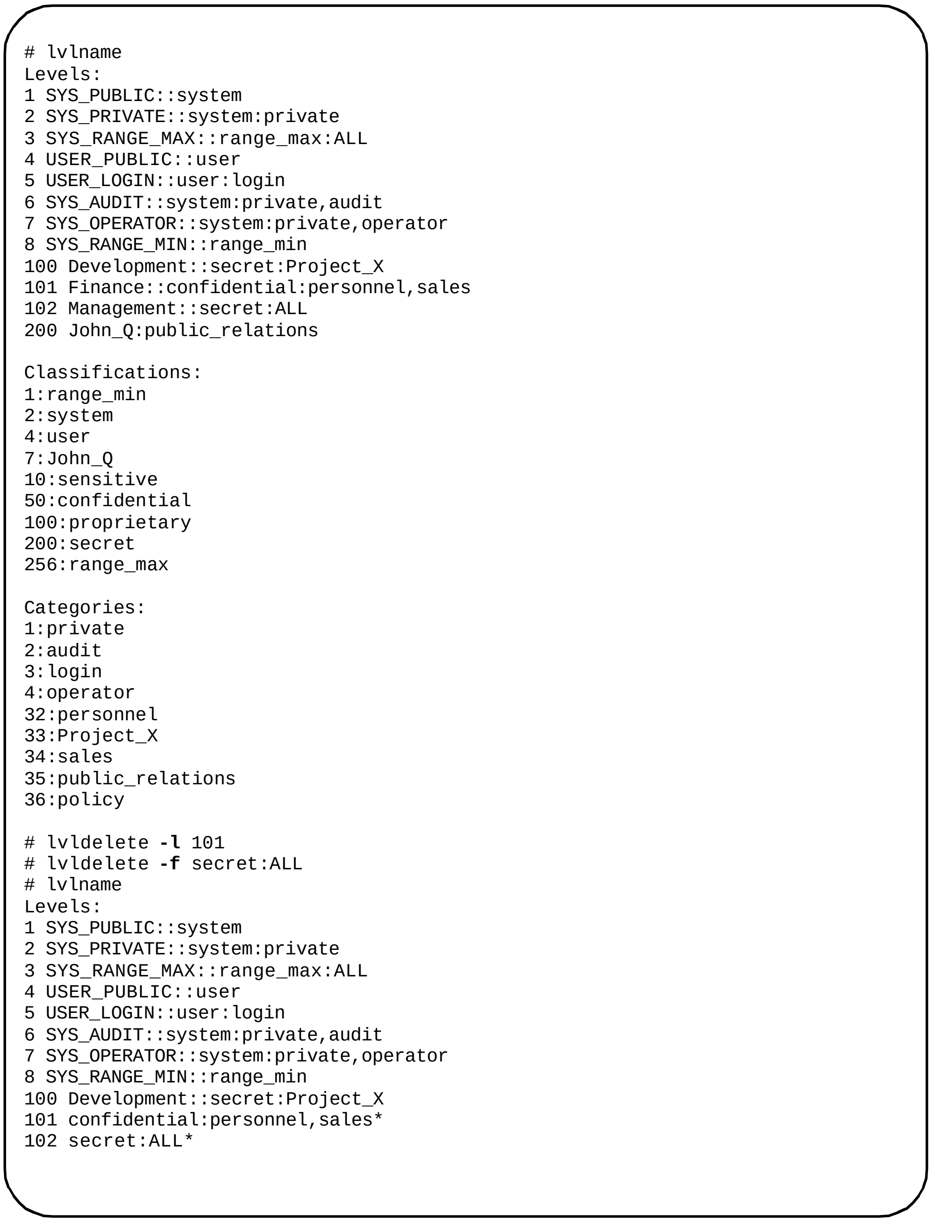
   You cannot use an alias to specify the level to be deactivated.

2. Check the database using **`lvlname`** to ensure the level has been deactivated. Enter:

   ```
   lvlname
   ```

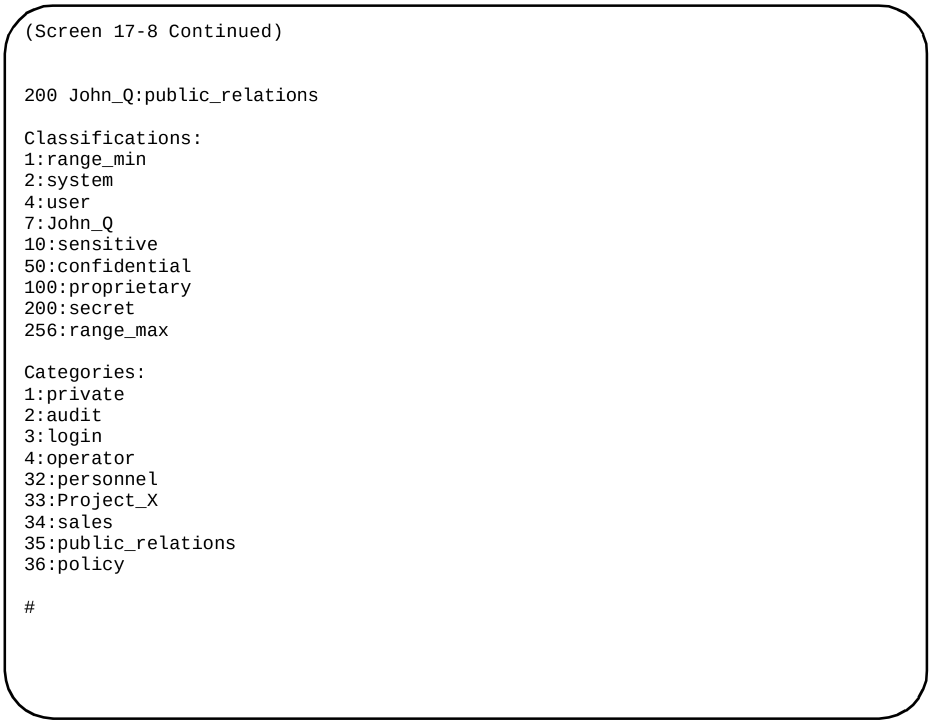
Example: Deactivating a Security Level

Screen 17-8 is an example of using the **`lvldelete`** command to change a level to inactive. It assumes the same categories, classifications, levels, and aliases as defined in previous examples.



```
# lvlname
Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min
100 Development::secret:Project_X
101 Finance::confidential:personnel,sales
102 Management::secret:ALL
200 John_Q:public_relations

Classifications:
1:range_min
2:system
4:user
7:John_Q
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy

# lvldelete -l 101
# lvldelete -f secret:ALL
# lvlname
Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min
100 Development::secret:Project_X
101 confidential:personnel,sales*
102 secret:ALL*
```

Screen 17-8. Changing a Security Level to Inactive



```
(Screen 17-8 Continued)


200 John_Q:public_relations

Classifications:
1:range_min
2:system
4:user
7:John_Q
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy

#
```

The administrator has chosen to make the levels **`Finance`** and **`Management`** inactive. The inactive levels are marked with an asterisk on the display; the aliases for these levels have been deleted and are not displayed.You can delete a level alias with the **`-a`** option of **`lvldelete`**. This will prevent any use of the alias, although the fully qualified name can still be used.

Before You Begin

You must be in single-user mode to perform this action.

Procedure

To delete a level alias,

1. Enter:

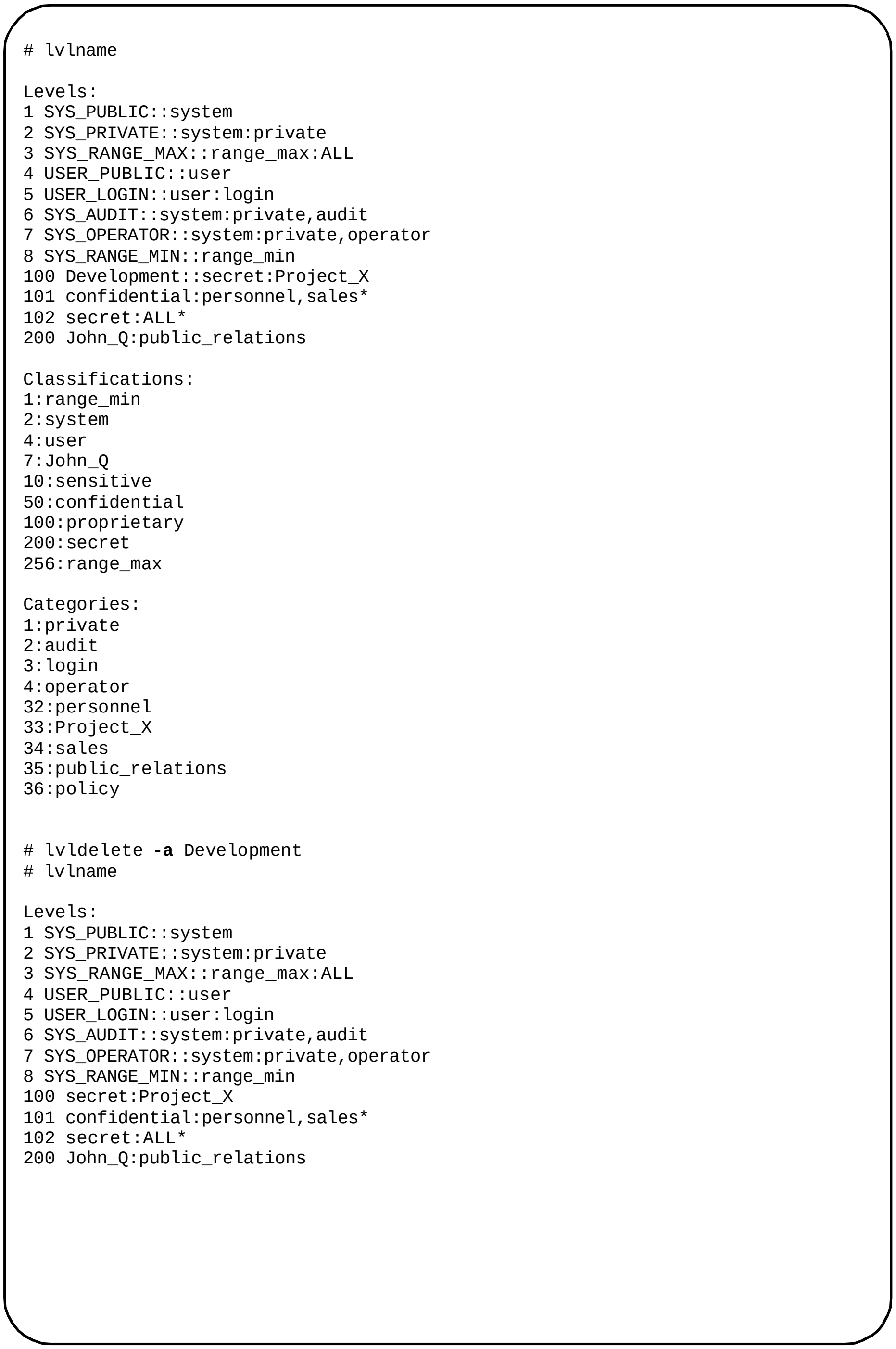
   `lvldelete` **`-a`** *alias_name*

2. Check the database using **`lvlname`** to ensure the alias has been removed. Enter:

   `lvlname`

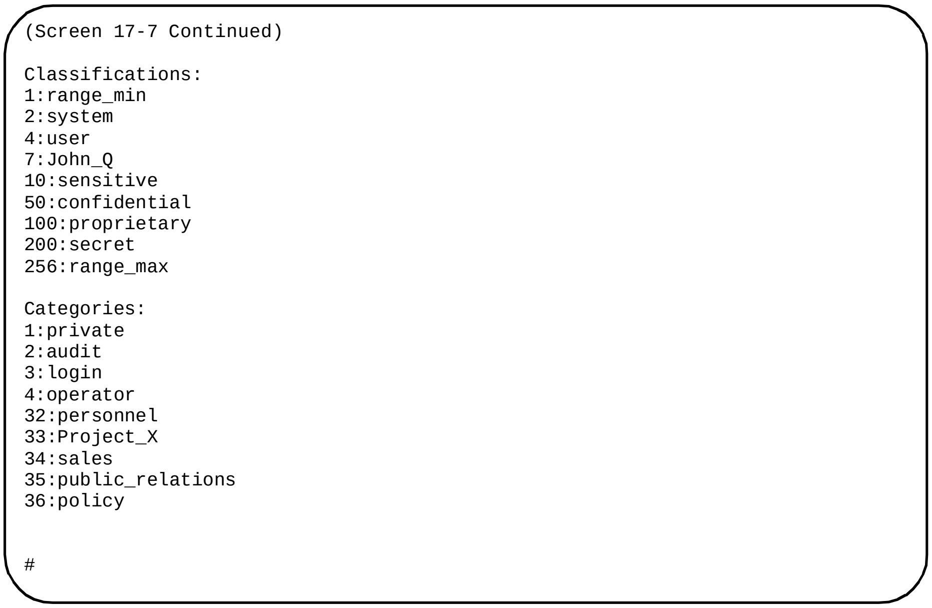
Example: Deleting an Alias

Screen 17-7 shows an example of a level alias name being deleted.



```
# lvlname

Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min
100 Development::secret:Project_X
101 confidential:personnel,sales*
102 secret:ALL*
200 John_Q:public_relations

Classifications:
1:range_min
2:system
4:user
7:John_Q
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy


# lvldelete -a Development
# lvlname

Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min
100 secret:Project_X
101 confidential:personnel,sales*
102 secret:ALL*
200 John_Q:public_relations
```

Screen 17-9. Deleting a Level Alias



```
(Screen 17-7 Continued)

Classifications:
1:range_min
2:system
4:user
7:John_Q
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy


#
```

The administrator has deleted the **`Development`** alias, but the **`secret:Project_X`** level is still active and can still be used.

Deleting a Classification or Category Name

You will probably not need to remove a classification or category definition. Once a classification or category is defined and used in a security level, you should not remove it, because the level will always be valid for level comparisons, even if it is made inactive.

You can rename a classification or category by deleting it with the **`lvldelete`** command, then assigning a new name to the classification or category identifier with the **`lvlname`** command. This will redefine all levels containing the classification or category, replacing the old name with the new one.

CAUTION

> Removing a classification or category used in a level will not change the definition of the level or make the level inactive. If you want the level to be made inactive, you must explicitly deactivate it.

If you delete a category or classification used in level definitions, the consistency checking mechanism for the level translation files will detect an undefined (deleted) classification or category the next time the level is validated by the system. The inconsistency will be reported, but no other action will be taken by the system. The dominance relationship of a level with a deleted classification or category will remain unchanged, as will access to files at that level. Since the name of the deleted classification or category is no longer valid, levels containing the deleted component can no longer be referenced by name. Such levels can still be referenced by alias or LID.

The **-h** option to **lvldelete** removes the definition of the specified classification, and the **-c** option to **lvldelete** removes the definition of the specified category. Classifications and categories can be specified either by name or by value.

Before You Begin

You should be in single-user mode to perform this action.

Procedure

To remove a classification or category definition, perform the following steps:

1. Enter:

   ```
   lvldelete -c category
   ```

   or

   ```
   lvldelete -h classification
   ```

2. Check the database using **lvlname** to ensure the desired action has been taken. Enter:

   ```
   lvlname
   ```

Classifications and categories can be specified either by name or by value.

Example Removing Classifications and Categories

Screen 17-10 shows an example of removing a classification and category.

The administrator removed the definitions for the classification **John_Q** and the category **public_relations**, then checked for level definitions using the deleted classification and category. Both are used in only one level definition, which the administrator deactivated with **lvldelete -l**.

Examining the Level Translation History Log

The **lvlname** and **lvldelete** commands and the system administration menu interface to level names maintain a log of operations that add or delete the assignment of a classification, category, level alias, or level identifier. The log is stored in the following eight files in the **/etc/security/mac** directory:

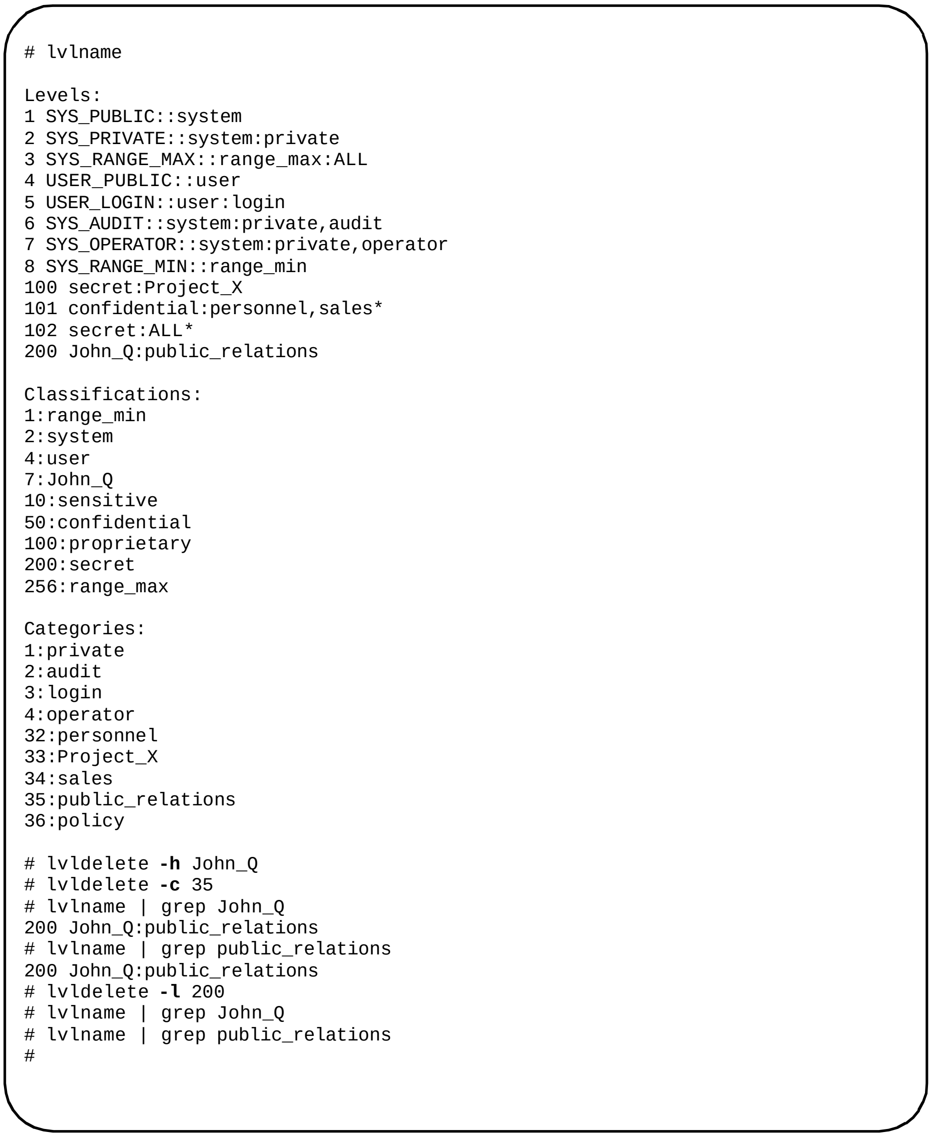
hist.class.add names and values of all classifications that have been added

hist.class.del names of all classifications that have been deleted

hist.cat.add names and values of all categories that have been added

hist.cat.del names of all categories that have been deleted

hist.lid.add definitions of all levels that have been added



```
# lvlname

Levels:
1 SYS_PUBLIC::system
2 SYS_PRIVATE::system:private
3 SYS_RANGE_MAX::range_max:ALL
4 USER_PUBLIC::user
5 USER_LOGIN::user:login
6 SYS_AUDIT::system:private,audit
7 SYS_OPERATOR::system:private,operator
8 SYS_RANGE_MIN::range_min
100 secret:Project_X
101 confidential:personnel,sales*
102 secret:ALL*
200 John_Q:public_relations

Classifications:
1:range_min
2:system
4:user
7:John_Q
10:sensitive
50:confidential
100:proprietary
200:secret
256:range_max

Categories:
1:private
2:audit
3:login
4:operator
32:personnel
33:Project_X
34:sales
35:public_relations
36:policy

# lvldelete -h John_Q
# lvldelete -c 35
# lvlname | grep John_Q
200 John_Q:public_relations
# lvlname | grep public_relations
200 John_Q:public_relations
# lvldelete -l 200
# lvlname | grep John_Q
# lvlname | grep public_relations
#
```

Screen 17-10. Removing Categories and Classifications

`hist.lid.del` definitions of all levels that have been deactivated

`hist.alias.add` definitions of all level aliases that have been added

`hist.alias.del` definitions of all level aliases that have been deleted

Each entry in these log files also contains the time and date of the change. The history log files may contain information you will need when maintaining level names. The **`-p`** option to **`lvlname`** displays all level naming actions that have occurred since the history log was last cleared.

The **`lvlname -p`** command prints:

- level identifiers

- classifications
- categories
- alias names

The level identifiers are sorted by the value of the level identifier, the classifications and categories are sorted by the associated value, and the alias names are sorted alphabetically. Log entries for the same value are sorted by date. Every name (level, classification, category, or alias) has an entry for each action (creation or deletion) taken against it. The date and time of each action is also listed. Each entry in the history log is prefixed by either `ADD` or `DEL`, denoting the type of operation.

Procedure

This procedure can be performed only by the SSO role.To look at the level translation history log, perform the following steps:

1. Enter

   ```
   tfadmin lvlname -p
   ```

Example

Screen 17-11 shows an example of the output of **`lvlname -p`**.

Saving and Clearing the Level Translation History Log

If the level name history log files become too large, you may save and then remove the log files.

CAUTION

The log files are the only record of changes made to the level names (unless you keep a system log book with this information). If you want to refer to this information in the future, you will need to save this information before clearing the log files.

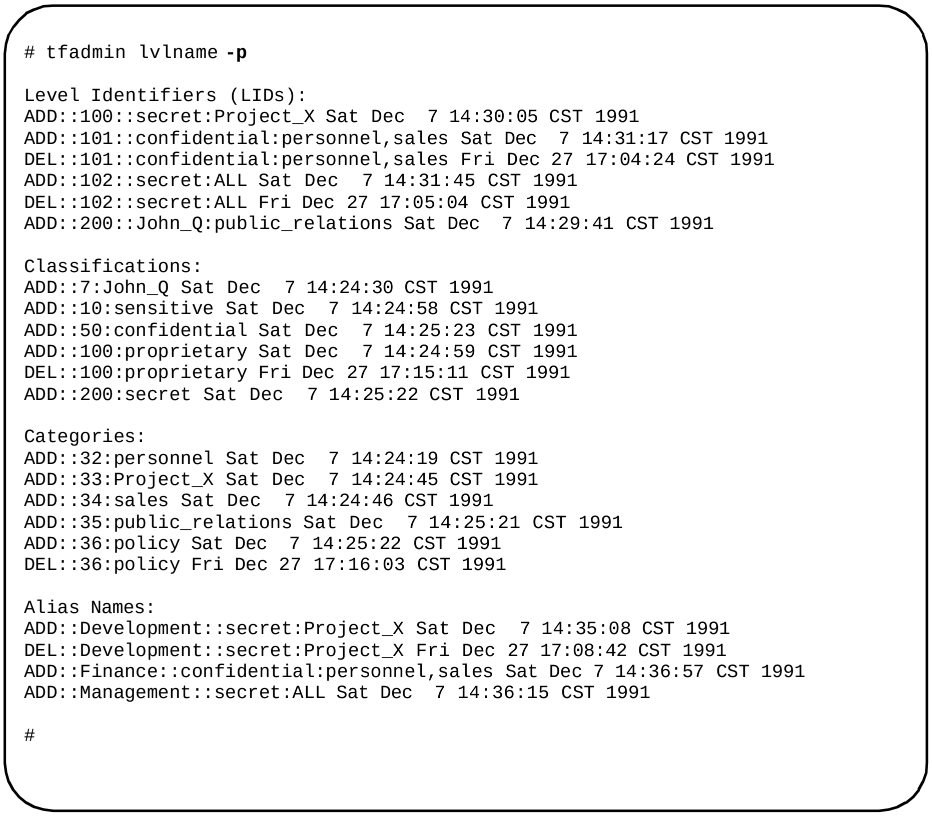
Procedure

To save and clear the level translation history log, perform the following steps:

1. Save a copy of the current log to a file. Enter:

   `lvlname` **`-p`** `\>`*file_name*

2. Archive the file from step 1 to a backup medium. See “Trusted Backup and Restore” for information on how to archive the file.



```
# tfadmin lvlname -p

Level Identifiers (LIDs):
ADD::100::secret:Project_X Sat Dec  7 14:30:05 CST 1991
ADD::101::confidential:personnel,sales Sat Dec  7 14:31:17 CST 1991
DEL::101::confidential:personnel,sales Fri Dec 27 17:04:24 CST 1991
ADD::102::secret:ALL Sat Dec  7 14:31:45 CST 1991
DEL::102::secret:ALL Fri Dec 27 17:05:04 CST 1991
ADD::200::John_Q:public_relations Sat Dec  7 14:29:41 CST 1991

Classifications:
ADD::7:John_Q Sat Dec  7 14:24:30 CST 1991
ADD::10:sensitive Sat Dec  7 14:24:58 CST 1991
ADD::50:confidential Sat Dec  7 14:25:23 CST 1991
ADD::100:proprietary Sat Dec  7 14:24:59 CST 1991
DEL::100:proprietary Fri Dec 27 17:15:11 CST 1991
ADD::200:secret Sat Dec  7 14:25:22 CST 1991

Categories:
ADD::32:personnel Sat Dec  7 14:24:19 CST 1991
ADD::33:Project_X Sat Dec  7 14:24:45 CST 1991
ADD::34:sales Sat Dec  7 14:24:46 CST 1991
ADD::35:public_relations Sat Dec  7 14:25:21 CST 1991
ADD::36:policy Sat Dec  7 14:25:22 CST 1991
DEL::36:policy Fri Dec 27 17:16:03 CST 1991

Alias Names:
ADD::Development::secret:Project_X Sat Dec  7 14:35:08 CST 1991
DEL::Development::secret:Project_X Fri Dec 27 17:08:42 CST 1991
ADD::Finance::confidential:personnel,sales Sat Dec 7 14:36:57 CST 1991
ADD::Management::secret:ALL Sat Dec  7 14:36:15 CST 1991

#
```

Screen 17-11. Displaying the lvlname History Log

3. If you wish, print the file from step 1 to keep a hard copy of the previous log available.

4. Once you have saved the files by one of these methods, use the **`rm`** command to clear them. New log files will automatically be created as needed by the **`lvlname`** and **`lvldelete`** commands. Enter:

```
rm /etc/security/mac/hist.class.add
rm /etc/security/mac/hist.cat.add
rm /etc/security/mac/hist.alias.add
rm /etc/security/mac/hist.lid.add
rm /etc/security/mac/hist.class.del
rm /etc/security/mac/hist.cat.del
rm /etc/security/mac/hist.alias.del
rm /etc/security/mac/hist.lid.del
```

Login Level Range

You can restrict the levels of new logins on the system by using the **`defadm`** command to set the **`LVLHIGH`** and **`LVLLOW`** values in the **`/etc/defaults/login`** file. When the system is delivered, both **LVLLOW** and **LVLHIGH** are undefined, so all login levels are allowed. Both must be defined to restrict the login level range.

During the login sequence, the level at which the user is attempting to log in is compared to the low and high values of the login level range. If the user's level does not dominate **LVLLOW** or is not dominated by **LVLHIGH,** the user is not allowed to log in.

CAUTION

> Be careful when setting the level range that you do not accidentally “lock yourself out” of the system. Make sure the level at which you log on to perform privileged administrative actions is within the newly defined login range.

The login level range affects only new login sessions; user processes may exist outside the login level range and may create child processes that inherit the parent's level. If you want to ensure there are no user processes outside the level range, you should reboot the system or make the change in single-user mode.

Displaying the Login Level Range

Use the **defadm** command to display these.

Procedure

To display the system login level range, perform the following steps:

1. Enter the following:

   ```
   defadm login LVLLOW LVLHIGH
   ```

   The result, if the range is undefined, will be:

   ```
   defadm: WARN: name LVLLOW does not exist
   defadm: WARN: name LVLHIGH does not exist
   ```

   If a range has been defined the result will be a single line of the form:

   `LVLLOW=`*level_name* `LVLHIGH=`*level_name*

Setting the Login Level Range

This task should be performed in single-user mode.

Procedure

To change the login level range, perform the following steps:

1. Enter the following:

   `defadm login LVLLOW=`*level_name* `LVLHIGH=`*level_name*

2. Verify the setting by displaying the login level range. Enter:

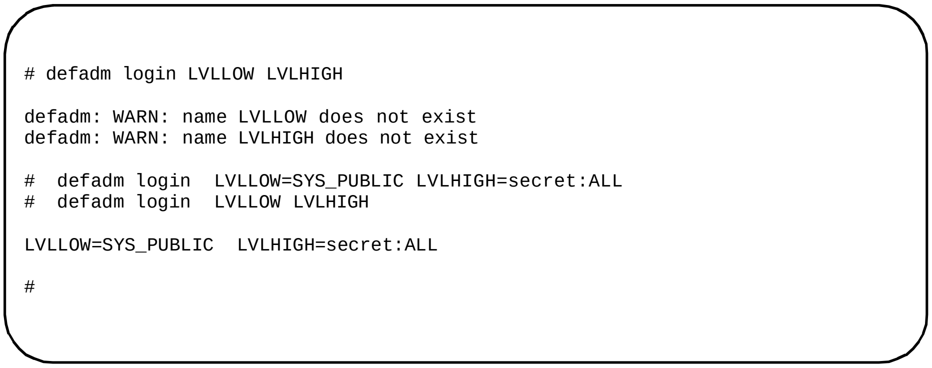
   `defadm login LVLLOW LVLHIGH`

   If a range has been defined the result will be a single line of the form:

   `LVLLOW=`*level_name* `LVLHIGH=`*level_name*

Example: Displaying and Setting the System Login Level Range

Screen 17-12 shows an example of setting and displaying the login level range.



```
# defadm login LVLLOW LVLHIGH

defadm: WARN: name LVLLOW does not exist
defadm: WARN: name LVLHIGH does not exist

#  defadm login  LVLLOW=SYS_PUBLIC LVLHIGH=secret:ALL
#  defadm login  LVLLOW LVLHIGH

LVLLOW=SYS_PUBLIC  LVLHIGH=secret:ALL

#
```

Screen 17-12. Displaying and Setting the Login Level Range

The administrator displays the current login range, then changes it, and confirms the change. This change will allow users to log in at levels between `SYS_PUBLIC` and **`secret:ALL`**.

NOTE

> In addition to setting a login level range for the system as a whole, you can set level ranges for specific terminals. A user's login level must be within both the system login level range and the terminal level range. See the *"Administering Terminals"* section of Chapter 15, "Administering Devices, Terminals, and Printers" of this book for detailed information on terminal level ranges.

User Login Levels

Each user must be assigned one or more authorized login levels. You should assign a default security level to the user and possibly one or more other valid login levels. These levels will be the only levels at which the user will be allowed to log in to the system.

The user will be logged in at the default level unless the user specifies another valid level. If you assign the user more than one login level, the user may change the default login level. The user may specify the security level as either a full level name or a valid level alias. The specified level must dominate **`LVLLOW`** and be dominated by **`LVLHIGH`** if both of these values have been defined (see the *"Login Level Range"* section in this chapter).

If you assign more than one login level to a user, set the user's home directory to a level dominated by all the user's login levels.You may also need to create a subdirectory in the user's home directory for each login level, so that the user will be able to create files at each level.

The procedure for assigning a user one or more login levels is described in Chapter 14, "User Account and Group Management" in this book. See the *"Getting Started"* section in the *User's Guide* for basic information on login procedures for all users.

Changing the Security Level of a File

The privileged command **`chlvl`** is used to change the security level of a file, directory, or a device special file (see the *"Device Security Mechanism"* section in Chapter 15, "Administering Printers, Terminals, and Devices" in this book for details on levels for device special files). You can specify the new security level either as a level alias or as the full level name. A file's level cannot be changed while the file is open or mapped.

CAUTION

> You should change the level of a file only after ensuring the change is appropriate for the data contained in the file. This is extremely important when the new level is not dominated by the old level, that is, when you are downgrading the level of the information.

It is particularly important to understand the implications of changing the levels on Trusted Computing Base (TCB) files, and changing a file's level to `SYS_PRIVATE` or `SYS_PUBLIC` (essentially making this file part of the TCB). Consider changing the level of a file:

- from `SYS_PUBLIC` to `SYS_PRIVATE`

  this is a modification of the TCB (the file is added to the TCB), but is a safe change to make since the new level is more restrictive (that is, unprivileged users will no longer be able to access the file)

- from `SYS_PRIVATE` to `SYS_PUBLIC`

this is a modification of the TCB (the file is removed from the TCB), and results in the ability of unprivileged users to access the file (creating the risk of unauthorized disclosure); the file is still, however, protected from unauthorized modification

- from `SYS_PUBLIC` or `SYS_PRIVATE` to `USER_PUBLIC` (or another non-“`SYS_`” level)

  this is a modification of the TCB, and results in the ability of unprivileged users to access the file (creating the risk of unauthorized disclosure) and creates the possibility that unprivileged users can modify the file

- from `USER_PUBLIC` (or another non-“`SYS_`” level) to `SYS_PUBLIC` or `SYS_PRIVATE`

  this is a modification of the TCB, and exposes the TCB to compromise by untrusted software

Changing the levels on files at `SYS_PRIVATE` and `SYS_PUBLIC` can have serious effects on system use because of the way TCB programs manipulate process privileges.

When some TCB programs attempt to execute or read a security sensitive file, the program removes the **`macread`** privilege from the process's working set so that the process can read or execute the file only if the process's level dominates the level of the file.

If, for example, the level on **`/etc/passwd`** is changed from `SYS_PUBLIC` to `USER_PUBLIC` TCB programs would be unable to read the **`/etc/passwd`** file, because the process's level (`SYS_PUBLIC` or `SYS_PRIVATE`) does not dominate the file's level (`USER_PUBLIC`) and the process does not have the **`macread`** privilege.

In general, it is safest to leave the levels on TCB files unchanged, and assign `SYS_PRIVATE` and `SYS_PUBLIC` levels only to trusted software.

Procedure

You must be in the SSO role to change the level of a file. To change the level of a file, perform the following step:

1. Enter:

   `tfadmin chlvl` *level_name file_name*

   The *level_name* specified may be a fully qualified level name or an alias. You can use the **`ls(1)`** command to verify the level has been changed.

Example: Changing the Level of a File

Screen 17-13 shows an example of displaying and changing the level of a file.

```
$ ls -z
report Finance
$ chlvl Management report
$ ls -z
report Management
$
```

Screen 17-13. Changing the Security Level of a File

The administrator displays the level of the file **`report`** by using the **`-z`** option of the **`ls`** command. The level of the file is reported by the alias **`Finance`**. The administrator then changes the level of **`report`** to **`Management`** and confirms the level of the file.

NOTE

Although the **`chlvl`** command can be used by administrators with the proper privilege to change the level of a mounted file system, it should not be used to change the level of the root vnode of the file system while the file system is mounted. Rather, in order to change the level of the root vnode of a file system, the file system needs to be unmounted first, then the level can be changed via **`chlvl`**, and finally the file system is remounted.

Multilevel Directories

The MAC restrictions on the creation of files prohibit non-privileged users from creating files with different security levels in the same directory. However, many existing programs depend on the ability to create temporary files in standard directories, such as**`/tmp`**. Users at different levels running these programs need to create files at more than one level in the standard directories. Such use would either violate the MAC restrictions on directories or require that all such programs be trusted. Since neither of these options is acceptable, the multilevel directory (MLD) feature is provided to allow creation of files at different levels.

The “Managing Files Securely” chapter in the *User's Guide* gives an overview of the use and structure of MLDs. As explained in that chapter, multilevel directories appear as normal directories to users in virtual mode, with minor differences. Users in virtual mode (the default MLD mode) can only “see” files at their login level. Users can optionally “see” all files at all levels to which they have MAC read access by changing to real mode.

This section describes the structure, function, use, and administration of multilevel directories. Because many of the administrative procedures require a working knowledge of the structure and function of multilevel directories, we suggest that you read this section carefully.

Structure of Multilevel Directories

A multilevel directory contains special sub-directories, called effective directories. The effective directories are individually created by the system when each one is initially accessed; the creation of an effective directory is automatic and transparent to the user.

The structure of a multilevel directory, **`/tmp,`** is shown in Figure 17-1.

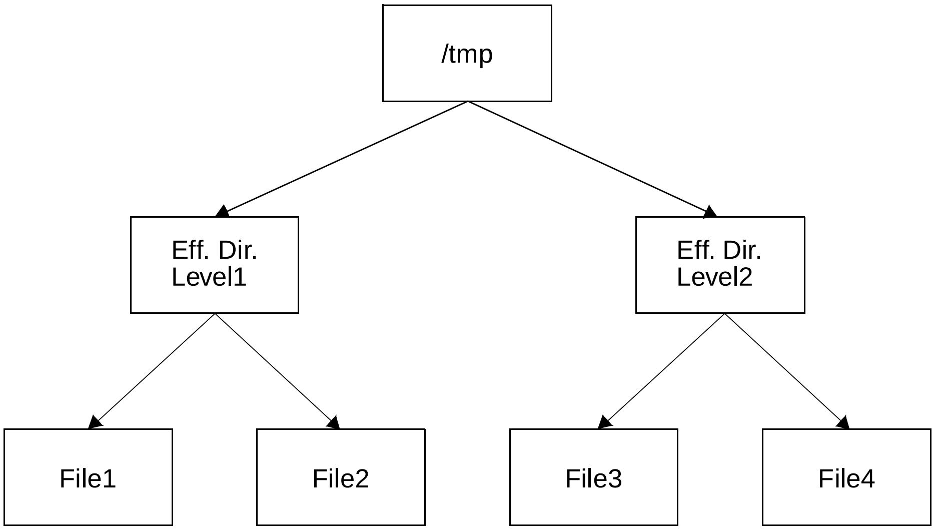


Figure 17-1. Structure of a Multilevel Directory

The multilevel directory shown has two effective directories, each with two files.

Associated with each process is a multilevel directory mode that determines the type of access to multilevel directories. This mode, which can be either **`real`** or **`virtual`**, is inherited from the calling process. The kernel uses the multilevel directory mode to determine how an access attempt to a multilevel directory should be handled.

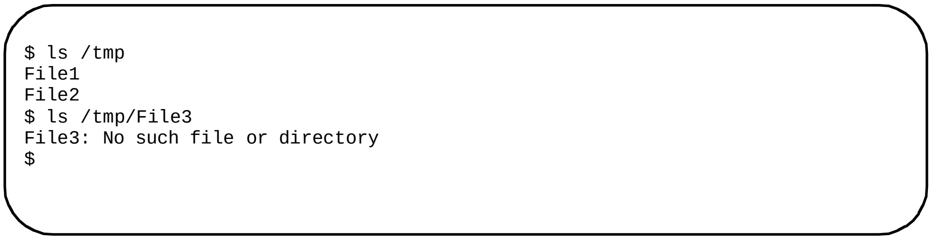
Virtual Mode Directory Access

If a process's multilevel directory mode is virtual (the default for all users), then an access to a multilevel directory by that process is modified by the kernel. The kernel changes the requested access to an access to an effective directory within the multilevel directory. The effective directory to which the kernel refers has the same level as the process accessing the multilevel directory. If an effective directory does not exist for the process's level, the kernel creates it automatically.

In Figure 17-1, a process in virtual mode at `Level1` would see the files **`File1`** and **`File2`** when accessing **`/tmp.`** A process in virtual mode at `Level2`, on the other hand, would see the files **`File3`** and **`File4.`** In either case, the process would not see the files contained in the other effective directory, nor would it see the effective directories themselves.

If a user in virtual mode accesses **`/tmp`** at a level for which there is no effective directory, the system will automatically create a new effective directory that it will appear empty.

The following example, based on the directory structure shown in Figur e17-1, demonstrates a multilevel directory access by a process in virtual mode.



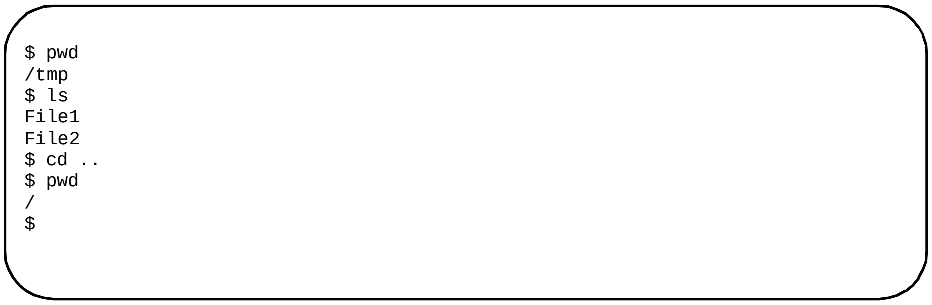
```
$ ls /tmp
File1
File2
$ ls /tmp/File3
File3: No such file or directory
$
```

Screen 17-14. Virtual Mode Access

In the above example, the user is logged in at level `Level1`. The **`ls`** command attempts to open the directory **`/tmp`**. The system determines that **`/tmp`** is a multilevel directory and "skips" through to the effective directory at `Level1`. The process "sees" only the files **`File1`** and **`File2`**. An attempt to access **`File3`** fails because that file is in a different effective directory.

In virtual mode, if the access to the multilevel directory is through an effective directory via the reverse link to the directory (**`..`**), then the access is interpreted as an access to the parent directory of the multilevel directory (in our example, the root of the file system).

Screen 17-15 shows an example of access to the reverse link in virtual mode.



```
$ pwd
/tmp
$ ls
File1
File2
$ cd ..
$ pwd
/
$
```

Screen 17-15. Reverse Link Access to a Multilevel Directory in Virtual Mode

The system detects that the access is to a multilevel directory, **`/tmp`**, through the "**`..`**" directory entry and changes the access to `/`, the parent of the multilevel directory.

Real Mode Access to Multilevel Directories

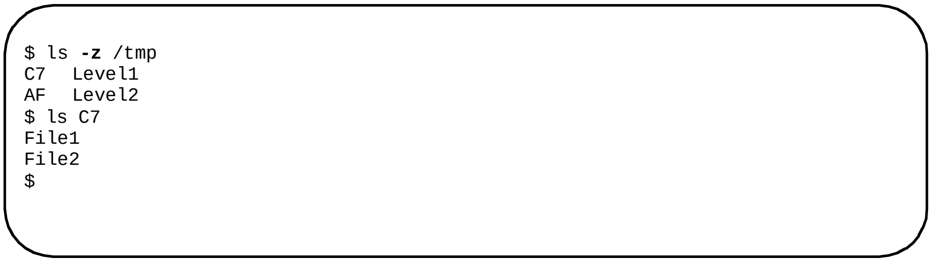
If the process's multilevel directory mode is real, an access to a multilevel directory by that process is not modified by the system. The process in real mode can see all effective directories in the multilevel directory, subject to MAC restrictions. The process can access all files in the effective directories, if the process passes the usual MAC and DAC checks on the effective directories (see the *“Mandatory Access Control”* section in this chapter and the *“Discretionary Access Control”* section of Chapter 16, “File Protection” in this book).

CAUTION

> Application programs should run in virtual mode only. If an application attempts to access a file in a multilevel directory when in real mode, the program may not be able to create the file, or other complications may occur.

Given the structure shown in Figure 17-1, a process in real mode would see the effective directories at `Level1` and `Level2` when accessing **`/tmp.`**

Screen 17-16 shows an example of processing in real mode. The system has given these effective directories names encoding the level identifiers (LIDs) of their respective levels.



```
$ ls -z /tmp
C7  Level1
AF  Level2
$ ls C7
File1
File2
$
```

Screen 17-16. Real Mode Processing

In the example in Scre e n17-16, a process in real mode displays the contents of **`/tmp`** using the **`ls -z`** command. The two effective directories are displayed with their level aliases. The process lists the contents of one of the effective directories by explicitly naming the effective directory.

Only effective directories can exist in a multilevel directory. In real mode, you can create a sub-directory within a multilevel directory, but the directory you create will be an effective directory. Processes in virtual mode will not be able to access your subdirectory by name. In real mode, you can create a regular file within an MLD, but this should be avoided since such files are inaccessible to processes in virtual mode.

CAUTION

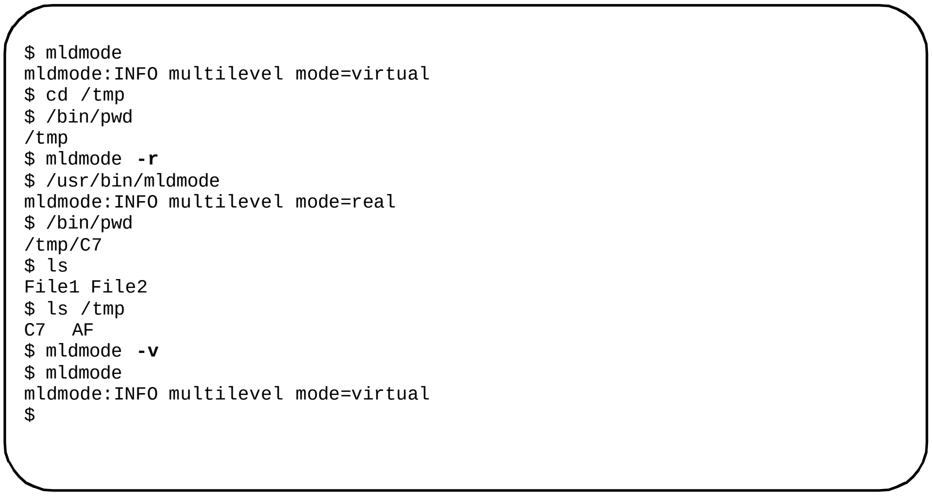
> To avoid conflicts with the naming scheme of the automatically created effective directories, you should avoid creating, moving, or renaming files and directories in a multilevel directory or an effective directory when in real mode. Such conflicts could prevent users at one or more levels from accessing the multilevel directory.

Displaying and Changing Your MLD Mode

You can display or change your MLD mode using the shell command **mldmode** or the **/usr/bin/mldmode** command.

Without any options, either command will tell you your current MLD mode. Use the **-r** option of either command to change to real mode, and the **-v** option to change to virtual mode.

When used with the **mldmode** shell command, these options change the MLD mode of your current process. Because the mode is inherited by processes created by the shell, this change will affect all commands you issue until the mode is changed back or until the shell exits. The following example, based on the directory structure shown in Screen 17-17, demonstrates the uses of the **mldmode** shell command.



```
$ mldmode
mldmode:INFO multilevel mode=virtual
$ cd /tmp
$ /bin/pwd
/tmp
$ mldmode -r
$ /usr/bin/mldmode
mldmode:INFO multilevel mode=real
$ /bin/pwd
/tmp/C7
$ ls
File1 File2
$ ls /tmp
C7   AF
$ mldmode -v
$ mldmode
mldmode:INFO multilevel mode=virtual
$
```

Screen 17-17. Example of the mldmode shell command

In Screen 15-17, the first **mldmode** shell command without options shows that the process is in virtual mode. Next, the user changes directory to **/tmp** and confirms the change. Since the process is in virtual mode, **cd** accesses the appropriate effective directory but appears to the user to be accessing **/tmp** itself.

Next, the user switches to real mode and confirms the change. In this case, the **/usr /bin/mldmode** command is used to display the current mode. Now the **pwd** command shows that the current directory is an effective directory within **/tmp.** The **ls** command shows **File1** and **File2** as the contents of the current directory, but still shows the two effective directories as the contents of **/tmp.**

The user then changes back to virtual mode, because this is the appropriate mode for normal processing.

When used with the **/usr/bin/mldmode** command, the **-r** and **-v** options specify the mode of a single command which is given as an argument. This can also be done with the **mldmode** shell command.

Screen 17-18 shows an example of the MLD Command.

```
$ /usr/bin/mldmode
mldmode:INFO multilevel mode=virtual
$ /usr/bin/mldmode -r ls /tmp
C7  AF
$ mldmode -r ls /tmp
C7  AF
$ ls /tmp
File1File2
$
```

Screen 17-18. Specifying the MLD Mode for a Single Command

In the example in Screen 17-18, the process is initially in virtual mode. The **`-r`** option of the **`/usr/bin/mldmode`** command is used to execute **`ls`** in real mode, showing the real mode view of the **`/tmp`** directory. Then **`ls`** is again executed in real mode by using the **`-r`** option of the **`mldmode`** shell command. The third **`ls`** command shows the virtual mode view of the directory, confirming that the process is still in virtual mode.

See the **`mldmode (1)`** and **`sh (1)`** online manual pages for details on both **`mldmode`** commands.

If a process is in a multilevel directory in real mode and then changes to virtual mode, the process will be able to directly access the multilevel directory as if it were in real mode. This applies only to access to the multilevel directory as the current directory. This condition will end when the process exits or changes to another directory. Note that this abnormal behavior in virtual mode is only true for direct access. Any access to the multilevel directory through any links (absolute or relative pathnames) will be in virtual mode.

Screen 17-19 shows an example of this abnormal behavior.

```
$ mldmode -r
$ cd /tmp
$ mldmode -v
$ mldmode
mldmode:INFO multilevel mode=virtual
$ ls
C7
AF
$ ls /tmp
File1
File2
$ cd /tmp
$ ls
File1
File2
$
```

Screen 17-19. Changing from Virtual to Real in a Multilevel Directory

In the example in Screen 17-19, the user changes directory to **`/tmp`** in real mode, then changes back to virtual mode. The **`ls`** command on the current directory shows the effective directories in **/tmp,** as it would in the real mode view. However, if the full path name is given, **ls** shows the contents of the effective directory, as it normally does in virtual mode. When the **cd** command is used again, the kernel skips to the effective directory and the process again shows the normal behavior for virtual mode.

The remainder of this section deals with the administration of multilevel directories.

Creating Multilevel Directories

Several public directories on the system are MLDs. The following MLDs are included in the installed system:

- **/tmp**
- **/var/tmp**
- **/var/mail**
- **/var/spool/cron/atjobs**
- **/var/preserve**
- **/var/spool/cron/crontabs**

Some optional packages also include MLDs. The Basic Networking Utilities package includes the following MLDs:

- **/var/spool/uucp**
- **/var/spool/uucppublic**

The Simple Mail Transfer Protocol Utilities package includes the following MLD:

- **/var/spool/smtpq**

The Networking Support Utilities include the following MLD for the Bilateral Identification and Authentication (Challenge-Response) scheme:

- **/var/iaf/cr1**

On installation, the above packages install a file in the **/etc/security/MLD** directory which contains the names of MLDs required for the package to function properly when the Enhanced Security Utilities are installed. A startup script in **/etc/rc2.d** reads these files and creates the MLDs listed, commenting out the lines for the MLDs created so they will be ignored the next time the script is run (that is, on the next transition to the multi-user state).

If a directory listed in the package file in **/etc/security/MLD** already exists and is not currently an MLD, the script will make the directory an MLD and restore the contents of that directory.

You may need to create additional multilevel directories for software that requires all users to access a specific directory. Because multilevel directories make it more complicated for a user to access files at other levels, MLDs are not recommended for general use.

Only a user with the appropriate privilege (the **`multidir`** privilege) can create a multilevel directory. The OP, SOP, and SSO roles can create a multilevel directory. Use the **`-M`** option of the **`mkdir`** command.

For this operation, the multilevel directory mode of the process is not important.

The effective directories are created on demand by the kernel when a user in virtual mode accesses the multilevel directory. The effective directory for a particular level is created only once, when the multilevel directory is accessed at that level for the first time. The effective directory for that level then exists until it is explicitly removed.

In virtual mode, user must pass normal DAC checks to access an MLD, and must be at a level that dominates the level of the MLD. In addition, the level and permissions of the effective directory control access to files below the MLD. In real mode, the normal MAC and DAC restrictions apply to both MLDs and effective directories.

The effective directories will inherit the access modes and ACL you assign to the multilevel directory. However, effective directories cannot have default ACLs, even if one is assigned to the multilevel directory.

CAUTION

> Creating a multilevel directory beneath another multilevel directory is useless. Any directory which is beneath a multilevel directory is either itself an effective directory, or is in a sub-tree with its root in an effective directory. A process in virtual mode will access the sub-tree rooted in the effective directory at the process's level. Therefore only processes at one level will have access to the second multilevel directory in the path.

Suggestions for Administering Multilevel Directories

Multilevel directories should be in file systems that are mounted read and write. If a multilevel directory exists in a file system that is mounted read-only, the system will not be able to create effective directories. Users attempting to access the multilevel directory (for reading) will get write-like error messages if the appropriate effective directory does not exist.

Similarly, if the file system becomes full, the automatic creation of an effective directory may fail because of the on-demand nature of the operation. You can prevent this problem from occurring by making sure that the file system is large enough for its intended use. A file system that is too small is likely to create problems for many users. See the chapter “Managing File System Types” in volume 2 of this book for more information on creating file systems.

Set the MAC and DAC attributes of an MLD to make it accessible to users who need to use it. For example, a general-purpose MLD such as **`/tmp`** should be at the SYS_PUBLIC level.

NOTE

If you need to change the level of an MLD, you must be in real mode. Otherwise, the level of the effective directory, rather than that of the MLD itself, will be changed. This will make the MLD inaccessible from your current level.

Before backing up, restoring, or copying an MLD, use the command **`mldmode -r`** to switch to real mode. (When you are done with the backup, restore, or copy operation, remember to switch back with the command **`mldmode -v`**). Only a process in real mode can recognize that a directory is an MLD. Use the **`tcpio`** command to backup and restore MLDs, and the **`-p`** option of the **`cpio`** command to copy MLDs. For details, see Chapter 18, "Trusted Backup and Restore" of this book, and the **`tcpio(1)`** and **`cpio(1)`** online manual pages.

CAUTION

Changing the levels of files (see the *"Mandatory Access Control"* section in this chapter) in an effective directory requires caution and forethought. Changing the level of the file is not enough to make the file accessible to users at the new level. The file must be moved into the appropriate effective directory for the new level. You must take care not to copy over a file with the same name.

It is strongly recommended that you do not change the level of a file in an effective directory.

Mounting a Multilevel Directory

Many systems are configured with a separate file system for the **`/tmp`** directory. Since **`/tmp`** must be a multilevel directory, the root directory of the file system to be mounted as **`/tmp`** must be a multilevel directory.To create this file system with a multilevel directory at the root requires the **`-o`** M option to **`mkfs`**. (Please note that the **`mkfs`** command is only available in single-user mode.)

The mount point should be a regular directory. It can be a multilevel directory, but if the root of the file system to be mounted is not an MLD, it will not inherit the MLD attribute of the mount point.

If you mount a file system on a multilevel directory (whether or not the root of the file system to be mounted is an MLD), you should be in real mode. If you are not in real mode, then the file system will be mounted on an effective directory.

Screen 17-20 shows an example of mounting a multilevel directory on a multilevel directory.

```
$ mldmode -r
$ mkfs -Fsfs -o M file_system
$ mount -Fsfs file_system /tmp
$ mldmode -v
$
```

Screen 17-20. Mounting a Multilevel Directory on a Multilevel Directory

In the example in Screen 17-20, the administrator changes to real mode, creates the file system, and mounts it on the **/tmp** directory. The administrator then changes back to virtual mode, since this is the appropriate mode for normal processing.

More information on mounting file systems can be found in the “Managing File System Types” chapter in volume 2 of this book.

Clearing a Multilevel Directory

Some multilevel directories, such as **/tmp**, are cleaned out automatically every time the system boots. At times it may become necessary to clean out a multilevel directory manually. If you need to clean out an entire multilevel directory, you must be in real mode. The suggested method is to remove all effective directories recursively.

Screen 17-21 shows an example of clearing a multilevel directory. Only the SSO role may clear a multilevel directory.

```
$ mldmode -r
$ cd /tmp
$ tfadmin ls
C7
AF
$ tfadmin rm -rf *
$ tfadmin ls

$ mldmode -v
$
```

Screen 17-21. Clearing a Multilevel Directory

In the example in Screen 17-21, the administrator changes to real mode and changes directory to **/tmp.** The **ls** command shows two effective directories. The administrator recursively removes the effective directories and their contents, confirms that **/tmp** is now empty, and switches back to virtual mode.

Removing Multilevel Directories

As in normal directories, a multilevel directory or an effective directory must be completely empty before it can be removed. This requires all effective directories below the multilevel directory be removed. In real mode, you can use the **`rm -r`** command to empty an MLD, deleting the effective directories and the files and subdirectories in them. The **`macwrite`** privilege is required to remove files and directories at multiple levels. The **`rmdir`** command will remove an effective directory or a multilevel directory.

Handling MLDs When Mandatory Access Controls Are Inactive

Multilevel Directories are part of the MAC feature of the Enhanced Security Package. If you remove the Enhanced Security package, or if you run a version of the kernel without MAC (for example, in single-user mode), access to both MLDs and effective directories will be the same as for regular directories.To access a file within an effective directory when MAC is not running, you will need to include the name of the effective directory in the path name.

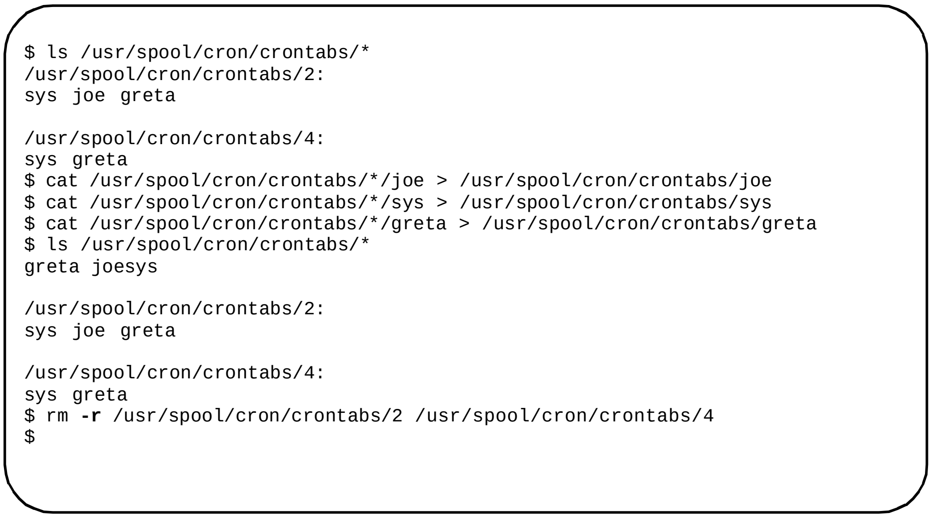
Some programs depend on files which are located in MLDs. When the MAC is removed or inactive, you will need to move these files from the effective directories to the parent directory so the programs can find them; then you may remove the effective directories. Failure to do so may result in the program aborting or being unable to execute.

There may exist files with the same name in two or more effective directories within the MLD. The appropriate handling of these files depends on the application.

In the case of files in **`/var/spool/cron/crontabs`** and **`/var/spool/cron/atjobs`**, files in different effective directories belonging to the same user should be combined in a single file in the MLD. For other applications, you might need to rename files when moving them to avoid name collisions.

If you expect to reinstall or reactivate MAC later, you should save the entire MLD before moving any files. Before deactivating or removing MAC, you can backup the contents of the MLD to tape with the **`tcpio`** command or copy them to another directory using **`cpio -p`**. See Chapter 18, “Trusted Backup and Restore” in this book for information on these commands.

Screen 17-22 shows an example of the reorganization of the **`/usr/spool/cron/crontabs`** directory after the Enhanced Security Package is removed:



```
$ ls /usr/spool/cron/crontabs/*
/usr/spool/cron/crontabs/2:
sys  joe  greta

/usr/spool/cron/crontabs/4:
sys  greta
$ cat /usr/spool/cron/crontabs/*/joe > /usr/spool/cron/crontabs/joe
$ cat /usr/spool/cron/crontabs/*/sys > /usr/spool/cron/crontabs/sys
$ cat /usr/spool/cron/crontabs/*/greta > /usr/spool/cron/crontabs/greta
$ ls /usr/spool/cron/crontabs/*
greta joesys

/usr/spool/cron/crontabs/2:
sys  joe  greta

/usr/spool/cron/crontabs/4:
sys  greta
$ rm -r /usr/spool/cron/crontabs/2 /usr/spool/cron/crontabs/4
$
```

Screen 17-22. Converting an MLD to a Regular Directory

18
Trusted Backup and Restore

18
Trusted Backup and Restore

Introduction

This chapter tells you how to perform trusted backups and how to restore backed-up data. Performing backups allows you to recover information if it is lost as a result of human or mechanical error.

The trusted backup commands copy both the data and the file security attributes to removable media such as tapes. The same commands, when used with appropriate privileges to restore backed-up data, restore files with their original security attributes (except for file privileges). File security attributes consist of Discretionary Access Controls (DAC), including owner, owning group, permission bits, Access Control Lists (ACLs), file privileges (for some executable files), and security levels, which are used for Mandatory Access Controls (MAC). See Chapter 16, “File Protection” chapter of this book for a discussion of file security attributes and privileges.

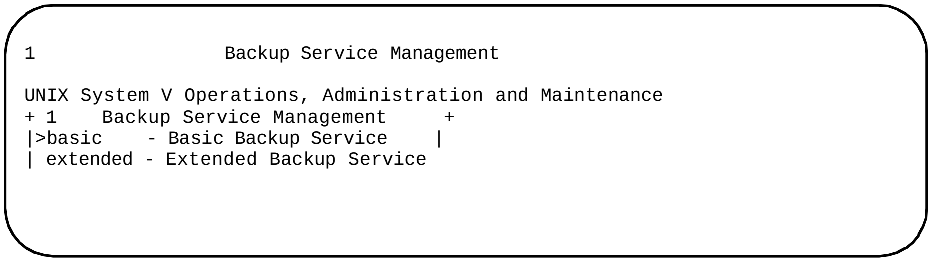
The Enhanced Security Utilities, or the Access Control List Utilities, or both must be installed for you to use these tools.

To help you execute trusted backups and restores, the system provides menus that guide you through the necessary steps in each process. If you prefer not to use the menus, you can use the shell-level commands explained in detail in this chapter.

To access the trusted backup menu, log in at level `SYS_PRIVATE` or higher and type

```
sysadm backup_service
```

The following menu will appear on your screen.

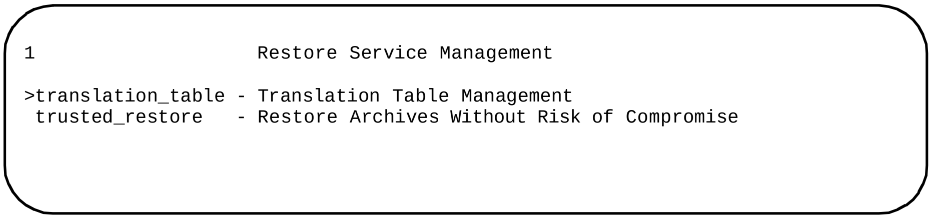


```
1                 Backup Service Management

UNIX System V Operations, Administration and Maintenance
+ 1    Backup Service Management     +
|>basic    - Basic Backup Service    |
| extended - Extended Backup Service
```

Select the one option offered on the menu; that is, the **`trusted_backup`** task. To access the system administration menu for restore, log in at level `SYS_PRIVATE` or higher and type

```
sysadm restore_service
```

The following menu will appear on your screen:



```
1                    Restore Service Management

>translation_table - Translation Table Management
 trusted_restore   - Restore Archives Without Risk of Compromise
```

Select the **`trusted_restore`** task from the menu.

The **`translation_table`** option on the menu allows you to create, modify, or delete a translation table file. This file is the argument to the **`-T`** option of the **`tcpio`**(1m) command and can be used when restoring an archive. Refer to the *"Restoring Files and Directories with tcpio"* section in this chapter for more information about translation table files.

If you prefer not to use the menus, you can perform the same tasks by executing shell-level commands instead. The following table shows the shell commands that correspond to the **`trusted_backup`** and **`trusted_restore`** tasks.

Task to Be Performed	***sysadm*** Task	Shell Command

Back up files without risk of compromise	trusted_backup	**`volcopy(1M)`**
		`find(1M) tcpio(1M)`

Restore files without risk of compromise	trusted_restore	**`volcopy(1M)`**
		`tcpio(1M)`

NOTE

Use only the **`trusted_backup`** and **`trusted_restore`** tasks from the system menus or the **`tcpio`** and **`volcopy`** commands described in this chapter to back up and restore files and retain the enhanced security attributes of the files. You may also use the **`translation_table`** task to create, display, modify, or remove translation table files for use with the **`trusted_restore`** task.

Each of these tasks is explained in this chapter. In addition, the online manual pages provide details about each shell command and its options.

Overview of the Commands

The system provides three commands, **`tcpio`**, **`rtcpio`**, and **`volcopy`**, for performing trusted backups and restores. A fourth command, **`cpio`**, is provided to transfer files and directories within your system.

Use the **`tcpio`** or **`rtcpio`** command to back up a list of files and directories. These commands are identical except that **`rtcpio`** allows you to back up or restore only files within a restricted range of security levels. After saving the files, you can restore all the backed-up files or selected files from the archive. Whether the archive was created using **`tcpio`** or **`rtcpio`**, you can use either of the two commands to restore files from the archive.

Use **`volcopy`** to back up an entire file system. The **`volcopy`** command makes a block-by-block copy of the file system without processing each file and directory individually, so it works more quickly than **`tcpio`** or **`rtcpio`** for large numbers of files. When you restore from a **`volcopy`** archive, the entire file system is restored, and any new or changed files on that file system are overwritten; you can't restore only selected files.

Use the **`cpio`** command to transfer files only. The **`-i`** and **`-o`** options of this command allow you to back up and restore file data without storing ACLs and security levels. They should not be used on a system on which the Enhanced Security Utilities are installed except for the instances cited in the *"Uses of the cpio Command"* section in this chapter.

You can use the **`-p`** option of **`cpio`** to copy a list of files and directories to another location on the system without creating an archive. This option preserves security attributes when copying files and directories.

NOTE

You must use the same command for restoring that you use to create the archive (except that **`tcpio`** and **`rtcpio`** archives are interchangeable). When selecting a backup method, keep in mind whether you will want to restore individual files or a full file system quickly.

Allocating an Archive Device

Before a device can be used for backup or restore, it must be allocated with the **`admalloc`** command. For information on **`admalloc`**, see Chapter 15, "Administering Printers, Terminals, and Devices" of this book.

CAUTION

> If a device is allocated with a broad security level, you should ensure that the Discretionary Access Control (DAC) settings on the device special file will prevent unauthorized users from accessing the device during any backup operations. See Chapter 16, “File Protection” chapter of this book for information about DAC.

When you allocate a device for trusted backup and restore, you need to ensure that the correct level range is specified to the **`-r`** option of **`admalloc`**. If you are backing up or restoring a complete file system with **`volcopy`**, the range of the storage device should match that of the disk device containing the file system. For backing up or restoring files using **`tcpio`** or **`rtcpio`** or transferring files to or from another system using **`cpio`**, the range of the backup device should encompass the level of your current process. If the device level range is not correct, the backup or restore may fail.

CAUTION

> To ensure the security of your backed-up data, the archive media themselves must be physically protected from unauthorized access. Removable media such as tapes and floppies are protected only by the DAC and MAC settings of the device on which they are mounted.

The remainder of this chapter describes in detail how to use the **`tcpio, rtcpio, volcopy,`** and **`cpio`** commands.

The tcpio Command

As the Trusted Import/Export backup and restore tool, **`tcpio`** is used to create an archive from a specified list of files. It can also selectively extract files from an archive that was created by a previous **`tcpio`** or **`rtcpio`** command.

Although **`tcpio`** is mainly intended for the use of system administrators as an incremental backup/restore tool, it may also be used by other users. When executed by a user with appropriate privileges, **`tcpio`** can save or restore files of any level to or from single- and multi-level devices, preserving file security attributes.

For privileged executable files, file privileges are saved and may be displayed using **`tcpio -t`** or **`tcpio -v`**. However, **`tcpio`** does not restore file privileges. A privileged user can assign appropriate privileges to the restored file using the **`filepriv`** command. [See the **`filepriv(1M)`** online manual page.]

When **`tcpio`** is used without appropriate privileges, all the restrictions that apply to the user also apply to **`tcpio`**. Only files to which the user has read access can be saved. Only archives to which the user has access can be restored. All files restored by an unprivileged user will be created at the user's current security level. ACLs will be preserved. Privilege is required to correctly restore a multilevel directory.

CAUTION

> To back up the contents of multilevel directories, you must be in real mode. [See **`mldmode(1)`** online manual page. Otherwise, only the subtree of the MLD at your current level will be saved. You must also be in real mode to correctly restore an MLD.

The **`-o`** option of **`tcpio`** creates an archive containing the files you specify. Whether invoked with or without privilege, **`tcpio -o`** preserves security information (level, owner, group, permissions, ACL, and file privileges) for each backed-up file, as well as some information on system databases used to validate the file security information on restore. If invoked without privileges, **`tcpio -o`** only backs up files and directories to which you have read access.

The **`-i`** option of **`tcpio`** restores specified files and directories from the archive. If you invoke **`tcpio -i`** with appropriate privileges, files will be restored with their original levels, owners, and groups unless you specify otherwise, and with theirACLs and original permissions. File privileges (if any) will not be restored, but may be displayed using the **`-t`** or **`-v`** option. Privilege is required to correctly restore a multilevel directory (MLD).

Without privileges, you can restore only archives to which you have access. **`tcpio -i`** sets the level of each file you restore to your current level. All restored files will be owned by you and belong to your current group. ACLs are preserved. The permissions of each restored file will be the same as the backed-up file except the bits set in your **`umask`** will be cleared. Also, the set-UID and set-GID mode bits on the restored files will be cleared.

CAUTION

> Invoking **`tcpio`** with a partial set of privileges may have undesired results. The roles included in the **`tfadmin`** database provide the full set of privileges used by **`tcpio`**.

Backing Up Files with tcpio

Use the **`-o`** option of **`tcpio`** to back up files and directories. **`tcpio`** reads the list of files to be backed up from standard input.

Before You Begin

You must be in the SOP or SSO roles to use **`tcpio`**.

Procedure

To back up files using **`tcpio`**, perform the following steps:

1. Create a list of the files you wish to archive and store this list in a file.
2. Use the **`cat`** command to list the file, and pipe the output to the following command line:

   `tfadmin tcpio` **`-o -O`** *backup_device* **`-v`**

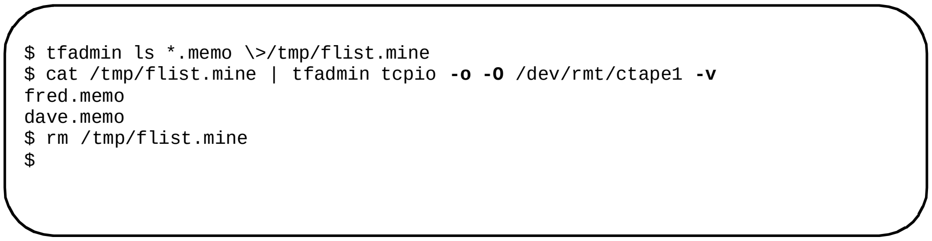
   The entire command line would appear:

   `cat` *file* **`|`** `tfadmin tcpio` **`-o -O`** *backup_device* **`-v`**

   Note that if the command to generate the file list is sufficiently simple, you can simply pipe the output of that command to **`tcpio`**.

Example

For example, if you want to back up all the files with names ending in **`.memo`** in your current directory, preserving security information, insert the backup medium and invoke **`tcpio`** as shown in the following figure:



```
$ tfadmin ls *.memo \>/tmp/flist.mine
$ cat /tmp/flist.mine | tfadmin tcpio -o -O /dev/rmt/ctape1 -v
fred.memo
dave.memo
$ rm /tmp/flist.mine
$
```

Screen 18-1. Backing Up Files with tcpio

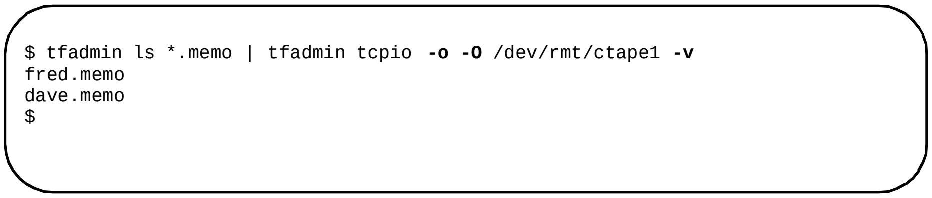
In this example, the user created a list of files to back up with the **`/bin/ls`** command, collected them into a file, and then and piped the list of filenames into the **`tcpio`** command. The options in this example are:

`-o` (copy out) Indicates that this is a backup.

`-O` Specifies the output device (in this case **`/dev/rmt/ctape1).`** The **`-O`** option is required when **`-o`** is used.

`-v` (verbose) Displays the name of each file or directory on the terminal as it is backed up.

Since it only took a simple command to generate the desired list, the above could be simplified:



```
$ tfadmin ls *.memo | tfadmin tcpio -o -O /dev/rmt/ctape1 -v
fred.memo
dave.memo
$
```

Screen 18-2.   More Backing Up Files with tcpio

Restricting Level Range When Backing Up Files

In some cases, you may want to limit an archive to files within a specified range of levels.

Before You Begin

You must be in the SOP or SSO roles to use **`tcpio`**.

Procedure

To back up files within a specified range of levels, perform the following steps:

1. Use the **`-X`** option with the **`-o`** option of **`tcpio`** to back up files selectively within a level range. Specify the range with the low level, a comma, and the high level. If you want to select files at a single level, specify that level for both low and high. Enter:

   `tfadmin tcpio` **`-o -O`** *backup device* **`-X`** *low_level*`,`*high_level*

Example

For example, to back up all files and subdirectories in your current directory with levels that dominate **`Secret:Sales`** and are dominated by **`TopSecret:Sales,`** type

```
ls|tfadmin tcpio -o -O /dev/rmt/0m -X
Secret:Sales,TopSecret:Sales
```

In the above example, the administrator runs **`tcpio`** with privileges, by accessing it via the **`tfadmin`** command. This allows **`tcpio`** to read and back up files and directories regardless of their MAC and DAC attributes. In this example, the administrator has access to the current directory, so no privileges are required by the **`ls`** command.

Only files and directories in the specified range are backed up when the **`-X`** option is used. In addition, the specified range is recorded in the archive header.

NOTE

> When you specify a directory to be backed up using tcpio, only the directory's attributes, not its contents, are backed up. If you want to back up the files in a directory, you must list them separately.

See the **tcpio(1)** online manual page for a complete description of available options.

Generating a File List with find

There are several ways to create a list of filenames to be backed up. You can use the **ls** command as in the above examples.You can use an editor to create a file containing the list of names, and then use the **cat** command to pipe the list into **tcpio**. One flexible way of generating a list of files is with the **find** command.

You can use **find** to print a list of files and directories that meet specified criteria in a specified sub-tree. Options to **find** allow you to select files by name, level, modification date, or other criteria. You can pipe the list of files into **tcpio**, or save it to a file which you can edit and use with **tcpio**.

The following command line shows an example of the use of **find** in conjunction with **tcpio**.

```
tfadmin find /home -print|tfadmin tcpio -o -O /dev/rmt/0m
```

This command line backs up all files and sub-directories under the **/home** directory. The **find** command generates a list of files to be backed up by **tcpio**. Since the administrator does not have read access to all of the files and directories to be backed up, both the **find** and **tcpio** commands must be called via the **tfadmin** command to allow privileged use.

See the **find (1)** online manual page for complete information on options to **find**.

NOTE

> When you give **tcpio** a list of pathnames of files to back up, each directory included in any of the pathnames is saved, even if you do not specifically list those directories.

Restoring Files and Directories with tcpio

Use the **-i** option of **tcpio** to restore files and directories from an archive created by **tcpio**.

You can use patterns in the notation specified on the **sh(1)** online manual page to specify names of files to restore, or you can list specific filenames. Separate the patterns or filenames with spaces. Each pattern should be in double quotes; otherwise it may be expanded by the shell. If you don't specify filenames or patterns, all files on the archive are processed; those which meet other criteria you specify are restored.

For example, to restore the file **/home/jackie/schedule** from the archive created in the previous example, enter

```
tfadmin tcpio -i -I /dev/rmt/0m /home/jackie/schedule
```

In this example, the **-i** option specifies that this is a restore ("import") operation. The **-I** option is used to specify the input device, in this case **/dev/rmt/0m.** The name of the file to be restored is the final argument. The **tcpio** command is given privileges via **tfadmin** so that the file's original level, owner, group, permissions, and ACL are restored.

With the **-X** option, you can selectively restore files in a specified level range, as in the following example:

```
tfadmin tcpio -i -I /dev/rmt/0m -X SYS_PUBLIC,Level5 "*.memo"
```

The above command line restores those files on the archive mounted on **/dev/rmt/0m** with names ending in **.memo**, and with security levels that dominate SYS_PUBLIC and are dominated by Level5. Only files that match both criteria (name and level) are restored. The use of the **tfadmin** command to run **tcpio** with appropriate privileges is required in order for the files to be restored at their original security levels.

If **-X** was used when creating the archive, the **-P** option displays the archive level range. Options are also provided to change the levels or owners of files during a restore. See the **tcpio (1)** online manual page for complete information on available options.

Handling Deleted Levels, Owners, or Group IDs

When restoring files, **tcpio** checks for any changes in the user ID, group ID and level ID databases. If a file's level has been deleted from the Level Database since the archive was created, **tcpio** displays a warning message and skips the file. If the file's owner or group ID has been deleted from the database or the name associated with the ID has changed, **tcpio** displays a warning and skips the file. If any user or group in the file's ACL has been deleted from the database, or if any user and group name associated with an ACL entry has been changed, a warning message is displayed but the file is restored. (Warnings are not issued and files are not skipped for IDs that were already invalid when the archive was made.)

To restore these files, you need to assign valid owners, groups, or levels. This assignment can be done with either of two methods.

The first is to provide a list of files for **tcpio** to restore, using the **-R** option to specify a new owner or the **-N** option to specify a new level. All the restored files will be assigned the specified owner (for **-R**) or level (for **-N**) regardless of their current owners or levels. The **-R** option also changes the group of each restored file or directory to the group associated with the specified owner.

For example, if the owner of the file **/sales/records/fy1989** is no longer a valid user on the system, and the user with UID 58 needs the file restored from a **tcpio** archive, enter:

```
tfadmin tcpio -i -I /dev/rmt/0m -R 58 /sales/records/fy1989
```

The file will be restored with UID 58 as owner and with the group associated with this user ID in the password file. The file's level is not changed.

The **-T** option of **tcpio** provides another method for changing file owners, groups, or levels during a restore. This option will also change IDs in ACLs.

To use the **-T** option you need to create a file that maps each deleted user, group, or level to a valid user, group, or level for the files you want to restore. The following figure shows an example of a map file:

```
UI  jeanie  greta
GI  sales   marketing
LI  33      111
```

The first column of the map file specifies the type of identifier: UI for user ID, GI for group ID, and LI for level ID. The second column is the identifier associated with files on the archive, and the third, the new identifier to be assigned when restoring files.

Users and groups can be specified by name or numeric ID. Levels may be specified by fully qualified name, alias, or LID. (In the case of deleted levels, the LID must be used.)

The fields must be separated by spaces or tabs, and each entry must be on a separate line.

You can create the translation table using an editor such as **ed**, or you can select the translation_table task on the sysadm restore_service menu.

After creating the map file, use the **-T** option to tell **tcpio** to use the information in the file. If the map file shown above is in the file **/misc/map.tcpio,** enter

```
tfadmin tcpio -i -I /dev/rmt/0m -T /misc/map.tcpio
```

to restore files on the archive, changing IDs. Only those files that reference the user **jeanie** and the group **sales** and the level ID 33 will be restored. Each archived file belonging to user **jeanie** and group **sales** and with level identifier 33 will be restored with **greta** as the new owner, with **marketing** as the new group, and with level identifier 111. Files with an ACL entry for **jeanie** will be restored with greta replacing **jeanie** in the ACL, and those with an ACL entry for **sales** will be restored with **marketing** replacing **sales** in the ACL.

The rtcpio Command

The **rtcpio** command is a restricted version of **tcpio**. It is intended for use by operators who need to back up and restore user files, without having access to system files. It restricts the level range of files you can back up and restore. This command is intended for use by users assigned to the OP role.

You can use **rtcpio** just as you would **tcpio**. The only difference is in the use of the **-X** option. If you specify the **-X** option with **rtcpio**, the level range you specify must be within the range USER_PUBLIC,USER_LOGIN. If you do not specify the **-X** option, **rtcpio** will limit the range of files to be backed up or restored to USER_PUBLIC,USER_LOGIN.

Because privileged use of **tcpio** is available only to administrators, operators must use **rtcpio** to back up and restore files. The **rtcpio** command can be used to restore files from archives created by either **rtcpio** or **tcpio**.

Full File System Backup and Restore

Use the **volcopy** command to copy an entire file system quickly. This command, available only to administrators, duplicates the file system block by block, without processing individual files. Because files are not processed individually, you cannot selectively restore files from a **volcopy** archive.

You can use **volcopy** to copy a file system from a disk to tape (backup); from tape to disk (restore); or from one disk to another. You must be a user assigned to the SOP or SSO roles to use this command. If the file system supports MAC security levels and ACLs volcopy preserves these file attributes. Whether a backup or a restore is done depends on the device you specify as source and destination.

For example, the following command line might be used to back up the **/usr** directory:

```
tfadmin volcopy -F sfs usr /dev/rdsk/1s0 usr /dev/rmt/0m usr411
```

In the above example, **volcopy** is given the following options and arguments:

- The **-F** option specifies the type of file system to be backed up. The argument to this option, **sfs**, stands for Secure File System (SFS).
- The name of the file system to be backed up, **usr**. This is the name stored in the file system superblock. By convention, it matches the mount point for the file system.
- The name of the raw disk segment containing that file system, **/dev/rdsk/1s0.** (This is only an example; device names on your system may be different.)
- The physical volume name of the source, **usr**. In this case, the physical volume name matches the file system name.
- The destination device, **/dev/rmt/0m.**
- The volume name to be assigned to the destination media, `usr411`. This field is limited to six characters and should match the label written on the tape. If the tape already has a physical label, you can use a hyphen to keep the existing name.

The following command line could be used to restore **/usr** from the **volcopy** archive created above:

```
volcopy -F sfs usr /dev/rmt/0m usr411 /dev/rdsk/1s0 usr
```

This is almost the same as the line used for backup, except the source and destination names are reversed. This makes sense because the tape drive that was the output device during the save is the input device for the restore, and the disk that was the input device for the save is the output device for the restore. The volume name assigned to the archive during backup, `usr411`, is now the name of the source volume.

volcopy Security Checks and Warning Messages

When you copy an SFS type file system with **`volcopy`**, security information such as file levels, and ACLs is copied along with the data. When copying from disk to tape, **`volcopy`** also stores the system name and information about the level database. This information is used to confirm the validity of the security information when you restore the file system by copying from tape to disk.

If you try to restore the archive to a different operating system, or if the level database has changed since the archive was created, **`volcopy`** will display a warning and you will need to decide whether to continue with the restore.To make this decision, you need to evaluate the extent to which security data on the archive may be invalid and what will need to be done to fix the invalid security data.

If the archive was created on a different system, the security information (levels, owners, ACLs and groups) will probably be invalid. Use of **`volcopy`** between systems is not recommended. In this situation **`volcopy`** is guaranteed to work correctly, only if you maintain identical level, user, and group databases on the two systems.

If the level database has changed since the archive was made, some of the archived files may have levels that have been deleted from the system. If you decide to continue with the **`volcopy`**, these files will be restored at their original (no longer valid) levels, so only privileged users can access them. You can check the level history files using **`lvlname -p`** to find out which levels have been deleted. Then use the **`-level`** option of the **`find`** command to identify files at those levels on the restored file system.You can then use the **`chlvl`** command to assign each file and directory an appropriate level. [For detailed information on these commands, see the **`lvlname(1M)`**, **`chlvl(1M)`**, and the **`find(1)`** online manual pages.

Uses of the cpio Command

You should always use either **`tcpio`** or **`volcopy`** to back up and restore files or file systems on a system on which the Enhanced Security Utilities are installed and running. Only these commands retain all file security information while backups and restores are being done.

However, the ability to create and read archives with **`cpio`** is provided for compatibility with earlier versions of the operating system. The only legitimate use of **`cpio -i`** when the Enhanced Security Utilities are installed is to restore files from archives created before this package was installed. Likewise, **`cpio -o`** can be used if it is necessary to transfer files to a system without the Enhanced Security Utilities.

CAUTION

> Archives created with **`cpio`** do not preserve file or ACLs levels. When you restore files using **`cpio`**, they will inherit your current login level. Sensitive data should not be backed up using **`cpio`**.

Unlike the **`-i`** and **`-o`** options, the **`-p`** option of **`cpio`** preserves file security attributes. This option preserves ACLs. If it is used with appropriate privileges, it also preserves file levels. You can use the **`-p`** option to copy directories and files from one part of the directory structure to another. If this is done with appropriate privileges, (the owner, group, and ACL) the owner and group of each file are preserved. If the **`-p`** option is used in conjunction with the **`-m`** option, and the destination is a multilevel file system, the level of each file is also preserved. A privileged user in real mode can also copy multilevel directories using **`cpio -pmd`**.

Without privileges, you can copy only files and directories to which you have read access. The copies are owned by you and belong to your current group, and they inherit your current level. ACLs are preserved.

See the **`cpio(1)`** online manual page for detailed information on the **`cpio`** command.

Quick Reference to Security Procedures

- Back up all files and directories under a specified path.

  `find` *path* **`-print | `**`tcpio` **`-oO`** *device*

- Back up files under the current directory in a specified level range.

  `find.` **`-print |`** `tcpio` **`-oO`** *device* **`-X`***lowlevel,highlevel*

- Back up a file system image.

  `volcopy` **`-F`** `sfs` *fsname srcdev srcvol destdev destvol*

- Restore a specific file from a **`tcpio`** archive.

  `tcpio` **`-iI`** *device filename*

- Restore files in a specified level range.

  `tcpio` **`-iI`** *device* **`-X`** *lowlevel,highlevel*

- Restore a file, changing the user and group IDs.

  `tcpio` **`-iI`** *device* **`-R`** *newuser filename*

- Restore files, changing the security levels.

  `tcpio` **`-iI`** *device* **`-N`** *newlevel file1 file2* `. . .` *filen*

- Restore a file system from a **`volcopy`** archive.

  `volcopy` **`-F`** `sfs` *fsname srcdev srcvol destdev destvol*

19
Security Procedures

19 Security Procedures

Introduction

This chapter contains procedures that you should use periodically to help keep your system secure.

Suggestions for Making Your System Secure

The security of any system is ultimately the responsibility of all who have access to it. As the administrator of your system, you need to do the following:

- Restrict physical access to your computer.
- Set security levels and access permissions to directories and files so they can be accessed only as needed by the owner, group, or others. Publicly writable directories are a security hazard. Allow them only for a good reason.
- Assign passwords to all logins and change them regularly. You can force them to be changed by implementing password aging. Pick non-obvious passwords: six-to-eight character nonsense strings of letters and numbers are recommended over real words. Remove or lock unused logins.
- Do not keep sensitive information on a system with dial-up ports; the security of any system with dial-up ports is difficult to guarantee. For more information on network security and dialup passwords, see *"Dial-up Passwords"* in Chapter 14, "User Account and Group Management".
- Record all use of the **su(1)** command. See *"Recording su Use"* in this chapter for two methods of doing this.
- Keep in mind that login directories, user profile files, and files in **/sbin**, **/usr/sbin**, and **/etc** that are writable by others are security give-aways.
- Encrypt sensitive data files. The **crypt(1)** command together with the encryption capabilities of the editors **ed** and **vi** protect sensitive information. The Encryption Utilities package (domestic customers only) must be installed before you can run **crypt(1)**.
- Do not leave a logged-in terminal unattended, especially if you are able to run privileged programs.

- Place an appropriate **umask** command in the system profile (**/etc /profile**) to set a default security level for file creation. For B2 system operation, the umask should be set to 0077. For B1 system operation, the umask should be set to 0077.
- Use full pathnames for critical commands (for example **/usr/bin/su** instead of **su**).
- Do not mount a removable medium (such as a floppy disk or cartridge tape), or add packages or programs, unless the contents are trusted. These file systems may contain set-user-ID or Trojan horse programs. These are the most common ways of spreading computer viruses.

Login Logging

Unsuccessful attempts to access your system can be logged using a logging mechanism supplied with the system. After a person makes five consecutive unsuccessful attempts to log on, all five attempts are logged in the file **/var/adm/loginlog**.

loginlog

To turn on the mechanism that logs unsuccessful attempts to access the system, the administrator must create the file **/var/adm/loginlog**. If this file exists and five (by default- to change this, see *“Limiting Attempts to Log In”* in Chapter 14, “User Account and Group Management”*)* consecutive unsuccessful login attempts occur, all are logged in **loginlog** and then **login** sleeps for 20 seconds before dropping the line. If a person makes fewer than five unsuccessful attempts, none of them are logged.

If **loginlog** does not exist, five (by default) failed login attempts will still cause the system to sleep for 20 seconds and drop the line, but nothing will be logged.

Enabling Login Logging

By default, this text file does not exist and logging is off. To enable logging, create the log file with read and write permission for **root** only.

Before You Begin

You must be in the SSO role to perform this procedure.

Procedure

To enable login logging, perform the following steps:

1. Begin execution of a subshell. Type:

```
/usr/bin/sh
```

The system should respond with a shell prompt.

2. Reset the default file creation privileges. Type:

   ```
   umask 077
   ```

3. Create the **loginlog** file. Type:

   ```
   > /var/adm/loginlog
   ```
   Set the group to **sys**.
   ```
   tfadmin chgrpsys/var/adm/loginlog
   ```
   Change the ownership of the file to **root**.
   ```
   tfadmin chownroot/var/adm/loginlog
   ```

4. Set the security level to SYS_PRIVATE.

   ```
   tfadmin chlvlSYS_PRIVA T E/var/adm/loginlog
   ```

   (See Chapter 15, "Mandatory Access Control".)

5. Return from the newly created shell. Type:

   ```
   exit
   ```

It is important to check and to clear the contents of the **loginlog** file occasionally, because this file may grow in size quickly. A large number of lines in a short amount of time in this file may suggest an attempt to break into the system. For more information about this file, see **loginlog(4)** online manual page.

Last Login Time

When a user logs on the system, the time the login was last used is displayed. Advise your users to check this time to ensure it corresponds to the actual last time they logged on. If it does not, an unauthorized person may have used the login.

Recording su Use

Users who run **su(1)** to become another user can compromise security by accessing other users' files without their knowledge. For this reason, a usage log is kept for **su**. Check the file **/var/adm/sulog** to monitor use of **su**. The format of **/var/adm /sulog** is described in the chapter "Directories and Files" in this book.

Another way to record all use of the **su** command is to print a message on the system console each time the command is run. To do this, add the line

```
CONSOLE=/dev/console
```

to **/etc/default/su.**

Checking File Characteristics

You can check the attributes of any file installed during installation using **`pkgadd`**.

Before You Begin

You must be in single-user mode to perform this action.

Procedure

1. Execute

   ```
   pkginfo
   ```

   to obtain a list of all the packages installed on the system.

2. Execute

   ```
   pkgchk -l packagename 2>pkg.prob.list
   ```

   to check the current characteristics of the files in a specific package against the package information that was stored during installation.

   A detailed description of any problems found is reported by **`pkgchk`** on standard error output, so you should probably redirect standard error output to a file (as shown in the above command line).

Once you obtain problem descriptions, you can use the **`chmod, chown, chgrp, setacl`**, **`chlvl,`** and **`filepriv`** commands to reset any file characteristics found to be in error. You can also use the **`pkgchk`** command in single-user mode to reset file characteristics. See **`pkgchk(1M)`** online manual page for more information.

Please note that the **`filepriv`** and **`pkgchk`** commands can be accessed in single-user mode only.

Creating Reference Files

The procedures in the following sections involve getting file attributes and checking those attributes for any suspicious settings. The checking portion of the work can be considerably simplified if you execute these procedures and save the results in a reference file. Then, at a later time, when you execute one of these procedures again, you can compare the current output with the output of the reference file. The **`diff(1M)`** command is a good way to compare two files. This will allow you to pinpoint any differences quickly and let you investigate them. Obviously, as you install or remove files, you need to update your reference files.

Check Set-UIDs

One procedure that is worthwhile is to check for suspicious programs that have the set-UID bit set, particularly to IDs such as **`root,`** **`sys`**, **`bin`**, and **`mail`**.

Before You Begin

This procedure is best executed in single-user mode. Make sure all file systems are mounted before performing this procedure.

Procedure

To get a list of all set-UID programs owned by a particular user, perform the following steps:

1. Type:

   ```
   find / -user user -perm -4000 -exec ls -ldb
   {} \; > file
   ```

   The *file* is the name of a temporary file to store the information.

2. Examine the contents of *file* from step 1, and look for suspicious programs. You can compare the file with a reference file if you have created one. Any differences should be investigated.

Example

The following example lists all set-UID programs owned by **`root`**. The output is saved in a file in **`/tmp`**. All mounted paths are checked by this command starting at `/`.

This program can be run for **`sys`**, **`bin`**, and **`mail`**, as well.

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

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

```
# find / -user root -perm -4000 -exec ls -ldb {} \; > /tmp/ckprm
# ca t/tmp/ckprm
-r-sr-xr-x    1 root     sys        65988 Nov  1 11:22 /sbin/su
---s--x---    2 root     lp         38780 Nov  1 11:23 /usr/bin/enable
-r-sr-sr-x    1 root     sys        29960 Nov  1 11:23 /usr/bin/passwd
-r-sr-xr-x    1 root     root       14480 Nov  1 11:23 /usr/bin/priocntl
---s--x---    2 root     lp         38780 Nov  1 11:23 /usr/bin/disable
-r-sr-xr-x    1 root     root       65988 Nov  1 11:51 /usr/bin/su
-rwsr-xr-x    1 root     sys          162 Nov  1 10:33 /usr/bin/disable_glogin
---s--x---    1 root     rar        45376 Oct 30 15:11 /usr/rar/bin/sh
-rwsr-xr-x    1 root     sys          197 Nov  1 11:28 /usr/bin/enable_glogin
-rwsr-xr-x    1 root     sys          174 Nov  1 10:33 /usr/bin/start_glogin
---s--s--x    1 root     uucp       30964 Nov  1 11:26 /usr/bin/ct
-r-sr-xr-x    1 root     bin        51804 Nov  1 11:26 /usr/bin/listusers
-r-sr-x---    1 root     bin        67408 Nov  1 10:28 /usr/lib/iaf/in.login/
scheme
.
.
.
#
```

Check Set-UIDs by File System

You can also check all set-UID programs on a file system, not just those owned by a particular user.

Before You Begin

You should be in single-user mode to perform this action. For each file system you wish to check, you need to know the mount point, file system type, and device name. These can be obtained from the **`/etc/vfstab`** file. The file systems should be mounted.

Procedure

To obtain a list of all of the set-UID files on a file system, perform the following steps:

1. Change your working directory to the mount point for the file system using the **`cd`** command. Type:

   `cd` *path_name*

2. Execute the following command:

   `ncheck` **`-F`** *fstype* **`-s`** *device_name* `| cut -f2 | xargs ls` **`-l`** `>`*file_name*

   The *fstype* is the file system type, and *file_name* is the name of a temporary file.

3. Check *file_name* for suspicious files. You can compare it with a reference file if you have one available.

4. Remove *file_name* when you are finished.

Example

The example below shows the use of **ncheck** to examine the **/usr** file system (assuming **/dev/dsk/1s0** is the special file), for files with a set-UID. to find the appropriate name for your system. The normal output of the **ncheck -s** command includes special files. If you are using another file system type, see the "Managing File System Types" chapter in volume 2 of *System Administration*. The output of the modified **ncheck** is used as an argument to the **ls** command. The use of the **ls** command is possible only if the file system is mounted.

```
# ncheck -F sfs -s /dev/dsk/1s0 | cut -f2 | xargs ls -l >/tmp/cksuid
# cat /tmp/cksuid
-r-sr-xr-x    1 root     sys        65988 Nov  1 11:22 /sbin/su
-rwxr-sr-x    1 bin      sys        43544 Nov  1 11:24 /sbin/swap
-r-xr-sr-x    1 bin      sys        14448 Nov  1 11:23 /usr/bin/crontab
---x--s--x    1 uucp     uucp       42376 Nov  1 11:23 /usr/bin/cu
---s--x---    2 root     lp         38780 Nov  1 11:23 /usr/bin/disable
---s--x---    2 root     lp         38780 Nov  1 11:23 /usr/bin/enable
-r-xr-sr-x    1 bin      sys        23392 Nov  1 11:23 /usr/bin/ipcs
-r-xr-sr-x    2 bin      mail      232240 Nov  1 11:22 /usr/bin/mail
-r-xr-sr-x    1 bin      mail      211356 Nov  1 11:22 /usr/bin/mailx
-r-sr-sr-x    1 root     sys        29960 Nov  1 11:23 /usr/bin/passwd
-r-sr-xr-x    1 root     root       14480 Nov  1 11:23 /usr/bin/priocntl
-r-xr-sr-x    2 bin      mail      232240 Nov  1 11:22 /usr/bin/rmail
---s--s--x    1 uucp     uucp       65244 Nov  1 11:23 /usr/bin/uucp
---x--s--x    1 uucp     uucp       15300 Nov  1 11:23 /usr/bin/uuname
---x--s--x    1 uucp     uucp       58732 Nov  1 11:23 /usr/bin/uustat
---x--s--x    1 uucp     uucp       48904 Nov  1 11:23 /usr/bin/uux
-r-sr-x--x    1 root     mail      106440 Nov  1 11:26 /usr/ucblib/sendmail
-r-sr-x--x    1 root     mail      109688 Nov  1 11:26 /usr/ucblib/sendmail.mx
-r-x--s--x    1 bin      dos        13920 Nov  1 11:20 /usr/bin/doscat
.
.
.
-r-x--s--x    1 bin      dos        30436 Nov  1 11:20 /usr/bin/doscp
-r-xr-sr-x    1 bin      sys        42988 Nov  1 10:28 /usr/bin/netstat
-r-sr-xr-x    1 root     root       65988 Nov  1 11:51 /usr/bin/su
-r-xr-s--x    1 sys      sys        19640 Nov  1 11:29 /usr/bin/uidadmin
---s--x---    1 root     lp        246156 Nov  1 10:28 /usr/lib/lp/lpsched
-r-sr-xr-x    1 root     sys        23824 Nov  1 01:27 /usr/rar/bin/su
-r-xr-sr-x    1 bin      sys        11274 Oct 20 09:25 /usr/sbin/whodo
#
```

In this example, the **/usr/rar/bin/su** should be investigated.

Checking File Privileges

Another possible avenue of attack on your system is through the placing of privileges on a program. You should check the privileges on your system periodically. If you create a reference file the first time you run this procedure, it will help you quickly discover any future changes.

Before You Begin

You should perform this task in single-user mode.

Procedure

To obtain a list of privileges on files on your system, perform the following steps:

1. Execute:

   ```
   find / -type f -perm -111 -print -exec filepriv {} \   ;
   > file_name
   ```

   The *file_name* is the name of a temporary file. This can be archived for later use as a reference file.

2. Check the *file_name* file for any suspicious programs; programs that have more privileges than they should. Take corrective action to remove the privileges using the **`filepriv`** command. See Chapter 9, “Administering Privilege” for more information.

Example

The following is an example of a program that finds all the executables with fixed privileges:

In this case, user `rar` has managed to place the entire set of privileges as fixed privileges upon **`/usr/rar/bin/xsh.`**

```
# find / -type f -perm -111 -print -exec filepriv {} \\; > sec.audit
# cat sec.audit
.
.
.
/usr/bin/message
/usr/bin/mimencode
/usr/bin/mkdir
fixedmacupgrade
/usr/bin/newgrp
fixedsetuid
/usr/bin/news
/usr/bin/newvt
.
.
.
/usr/rar/bin/xsh
fixedallprivs
.
.
.
#
```

A
Interrupts

Overview

This appendix contains information on the interrupts for the Night Hawk, Power Hawk and PowerMAXION systems. Information in this appendix should be used in conjunction with the applicable platform architecture manual.

Night HawK and PowerMAXION Interrupts

The information in this section is used to determine which interrupt controller pin number is associated with a particular device interrupt and interrupt priority level. An administrator using the information provided in Table A-1 through A-4, can assign interrupt handling to a specific processor. On Night Hawk and PowerMAXION systems, an external interrupt connects to one of multiple pins on the interrupt controller. Associated with each interrupt pin (i.e. the pin at which the interrupt occurs) are the following attributes:

processor — The processor attribute defines which processor processes the interrupt occurring at a pin. This information is defined in the configuration file and is discussed briefly later under,“Assigning Processors to Interrupt Pin Numbers.”

priority — The priority attribute defines the interrupt priority level (IPL) number.

vector — The vector attribute defines which vector in the ivt, interrupt vector table, is associated with the pin.

Assigning Processors to Interrupt Pin Numbers

The attributes for each pin are configured into the interrupt controller by the operating system at system initialization time. The priority and vector attributes are managed by the operating system together with the hardware. Tables A-2 through A-4, have a column called “Processor Number.” This column indicates the processor that can be assigned to handle this interrupt. Interrupts which have a value in the “TUNABLE NAME” column of these tables can be assigned to a specific processor by the administrator using the **`config`** utility (see Chapter 8, Volume 2) or **`idtune`**.

The tunable file for interrupt pins can be found in **`/etc/conf/mtune.d/pin`**. Some of the interrupt levels are assigned to a fixed processor and cannot be changed. Others may be assigned to a processor.

The assignment is as shown below:

0-3	specific processor number for PowerMAXION systems
0-7	specific processor number for HN6200 and HN6800 systems (Note: Limited to two processors (0-1) on HN6200 systems.)
-1	round-robin selection of processor number
-2	boot processor
-3	interrupt is disabled on all processors

The command **`idtune -g name`** reports the current setting for tunable parameter "**`name`**". A line is output to the screen with four separated fields:

current_value	original_default_value	minimum_value	maximum_value

An example using the HN6800 system is shown below:

To print the current assignment status of PICPU_PRIM_HVME-4 (pin 78) - (H)VME Level 4, enter the command: **`/etc/conf/bin/idtune -g PICPU_PRIM_HVME-4.`**

The display "`-1 -1 -2 7`" is explained as follows (reading left to right):

Field one, `-1`, is the current processor assignment for this interrupt pin. The value `-1` implies that the OS will assign a processor in a round-robin fashion during system initialization.

Field two, `-1`, is the original default processor assignment. If the operator does not modify this PIN assignment then the OS will default to a round-robin scheme for processor assignment.

Field three, `-2`, is the minimum value which can be assigned for this PIN.

Field four, `-7`, is the maximum value which can be assigned for this PIN.

To reassign to processor 1, enter: **`/etc/conf/bin/idtune -f PICPU_PRIM_HVME-4 1`** to force the re-assignment.

To confirm, enter: **`/etc/conf/bin/idtune -g PICPU_PRIM_HVME-4`**

which should display "`1 -1 -2 7` ".

The kernel must be rebuilt using **`config`** (or **`idbuild(1M)`**, and the system rebooted for any changes to tunable parameters to take effect. Refer to manual pages **`config(1M), Mtune(4)`** and **`idbuild(1M)`** for more information.

Determining Interrupt Priority Level and Interrupt Tunables For (H)VME Devices

Tables A-1 through A-4 are used to look up the applicable interrupt tunable and priority level for different devices. Table A-1 lists all the Concurrent supported (H)VME devices and the respective interrupt priority levels. The interrupt level number from this table is used to find the interrupt tunable in Table A-2 (HN6200), Table A-3 (PowerMAXION) or Table A-4 (HN6800). Tables A-2 through A-4 list both VME and non-VME devices and the associated interrupt tunables.

To determine the interrupt priority level and interrupt tunables for non-VME devices perform these steps:

1. Locate the device name in Table A-2 (A-3 or A-4).
2. Read the appropriate tunable and priority level from that line in the table.

To determine the interrupt priority level and interrupt tunable for (H)VME devices follow these steps:

1. Refer to Table A-1 and locate the row corresponding to the I/O device you are working with. Record the (H)VME "interrupt level number" from this row. You will need this number for the next step.
2. Now refer to Table A-2 (A-3 or A-4) and find the row that has the (H)VME "interrupt level number" that you obtained from step 1.On HN6800 systems be sure to select the line appropriate to either "Primary" or "Secondary," according to which I/O bus the device will be connected.

From this row, you can now read the interrupt tunable and the interrupt priority level.

Example;

Assume you need to know the interrupt level and tunable for the following device:

A Motorola IEEE 488 GBIP VME device that will operate on the Primary I/O bus.

1. Locate the Motorola IEEE 488 GBIP device in Table A-1 and note the "Interrupt Level" number. For this device, the number is "5".
2. Locate the row in Table A-2 (or A-4) that corresponds to interrupt level number 5 for a "Primary" I/O bus.
3. You will see a Priority Level of "HEX 35" and the Tunable is "PICPU_PRIM_HVME-5"

.

Table A-1. (H)VME Devices Interrupt Priority Levels

Board Name	(H)VME Interrupt Level
1553 V2 ABI	7
HSA	6
VIA	6
Interphase 4220 Cougar (SCSI-2)	6
Interphase Condor	5
Interphase 4207 Eagle (Ethernet)	5
Interphase 4211 Peregrine (FDDI)	5
Motorola IEEE 488 GBIP	5
DR11W	5
Multiplexer VMEbus Controller(MVC)	4
SYSTECH HPS MUX	4
HSDE OR VME-BBC	4

Model HN6200 Interrupt Pins

The interrupt pins for the Model 6200 are listed in Table A-2. Each interrupt is assigned a unique priority level and vector. When two or more interrupt requests occur simultaneously, the interrupt logic requests processing per the priority schedule shown in Table A-2. The vector for each interrupt is also shown. (Refer to the *HN6200 Architecture* manual for additional information.)

Table A-2. HN6200 Computer System Interrupt Levels

LEVEL			PIN NO.	TUNABLE NAME	IPL	PROC NO	HN6200 INTERRUPT DEFAULT
BINARY	DEC	HEX					
1111111	127	7f	96		---	0 - 1	Int-On-No-Int
1111101	125	7d	00		---	0	Powerfail
1111011	123	7b	01		---	0 - 1	Console Wakeup
1110111 1110110	119 118	77 76	02 03		--- ---	0 1	Inter–Processor Inter–Processor
1101111	111	6f	10		spl8	0 – 1	System Fault

Table A-2. HN6200 Computer System Interrupt Levels (Cont.)

LEVEL			PIN NO.	TUNABLE NAME	IPL	PROC NO	HN6200 INTERRUPT DEFAULT
BINARY	DEC	HEX					
1100111	103	67	15	PICPU_EDGE0-0	---	0 - 1	Edge 0 (Pin 0)
1100100	100	64	18	PICPU_EDGE0-1	---	0 - 1	Edge 1 (Pin 1)
1011111 1011110 1011101 1011100 1011011	95 94 93 92 91	5f 5e 5d 5c 5b	23 24 25 26 27	PICPU_RTC-0C0 PICPU_RTC-0C2 PICPU_RTC-0C1 PICPU_RTC-0C3 PICPU_RTC-0C4	--- --- --- --- ---	0 - 1 0 - 1 0 - 1 0 - 1 0 - 1	RTC 0 (Timer 0) RTC 2 (Timer 2) RTC 1 (Timer 1) RTC 3 (Timer 3) RTC 4 (Timer 4)
1010011	83	53	35	PICPU_EDGE0-2	---	0 - 1	Edge 2 (Pin 2)
1010001	81	51	37	PICPU_EDGE0-3	---	0 - 1	Edge 3 (Pin 3)
1001001 1001000	73 72	49 48	44 45		--- ---	0 1	Hard Clock (60Hz) Hard Clock (60Hz)
0111111	63	3f	52	PICPU_PRIM_HVME-7	spl7	0 - 1	VME Level 7
0111011	59	3b	54	PICPU_PRIM_HVME-6	spl6	0 - 1	(H)VME Level 6
0111001	57	39	56	PICPU_EXT_PORT-0A6	---	0 – 1	Port 0A6 (SCSI)
0110101	53	35	60	PICPU_PRIM_HVME-5	spl5	0 – 1	(H)VME Level 5
0110011	51	33	62	PICPU_UART-0	---	0 – 1	Console UART
0101111	47	2f	66	PICPU_EXT_PORT-0B5	---	0 - 1	Port 0B5
0101011	43	2b	70	PICPU_EXT_PORT-0C5	---	0 – 1	Port 0C5 (Ethernet)
0100111	39	27	74	PICPU_EXT_PORT-0D5	---	0 – 1	Port 0D5
0100011	35	23	78	PICPU_PRIM_HVME-4	spl4	0 – 1	(H)VME Level 4
0011111	31	1f	80	PICPU_PRIM_HVME-3	spl3	0 – 1	(H)VME Level 3
0011011	27	1b	82	PICPU_PRIM_HVME-2	spl2	0 – 1	(H)VME Level 2
0010111	23	17	84	PICPU_PRIM_HVME-1	spl1	0 – 1	(H)VME Level 1
0010101	21	15	86	PICPU_SOFTCLOCK	---	0 – 1	UNUSED
0010011	19	13	86		---	0 - 1	Soft Clock
000100100 01000	09 08	09 08	88 89		--- ---	0 1	Context Switch Context Switch
0000000	00	00	97		---	0 - 1	Spurious

PowerMAXION Interrupt Pins

The PowerMAXION consists of up to four processor cards with one interrupt controller and one PPC604 processor residing on each card. The interrupt pins for each interrupt controller on the PowerMAXION are listed in Table A-3. When two or more interrupt requests occur simultaneously on the same processor card, the interrupt controller processes the requests per the priority schedule shown in Table A-3. In the PowerMAXION architecture multiple devices may share one vector. (Refer to the *PowerMAXION Architecture* manual for additional information.)

Table A-3. PowerMAXION Computer System Interrupt Level s

LEVEL			PIN NO.	TUNABLE NAME	PROC NO.	IPL SYMBOL	PowerMAXION INTERRUPT DEFAULT
BINARY	DEC	HEX					
1000000	---	40	---		---	---	Int-On-No-Int
0111111	63	3f	---		---	---	Reserved
0111110	62	3e	---		---	---	Reserved
0111101	61	3d	---		---	PLXCALL	Inter-processor
0111100	60	3c	---		---	---	Reserved
0111011	59	3b	---		---	PL8	System Fault
0111010	58	3a	---		---	PLPROBE	Reserved
0111001	57	39	---	PICPU_EDGE0-0	---	---	Edge 0
0111000	56	38	---	PICPU_EDGE0-1	---	---	Edge 1
0110111	55	37	---		---	---	Reserved
0110110	54	36	---		---	---	Reserved
0110101	53	35	---	PICPU_RTC-0C0	---	---	Real-Time Clock 0 (RTC 0)
0110100	52	34	---	PICPU_RTC-0C1	---	---	Real-Time Clock 1 (RTC 1)
0110011	51	33	---	PICPU_RTC-0C2	---	---	Real-Time Clock 2 (RTC 2)
0110010	50	32	---	PICPU_RTC-0C3	---	---	Real-Time Clock 3 (RTC 3)
0110001	49	31	---	PICPU_RTC-0C4	---	---	Real-Time Clock 4 (RTC 4)
0110000	48	30	---		---	---	Reserved
0101111	47	2f	---		---	---	Reserved
0101110	46	2e	---	PICPU_EDGE0-2	---	---	Edge 2
0101101	45	2d	---	PICPU_EDGE0-3	---	---	Edge 3
0101100	44	2c	---		---	---	Reserved
0101011	43	2b	---		---	---	Reserved
0101010	42	2a	---		---	---	60Hz Hard Clock
0101001	41	29	---		---	---	Reserved
0101000	40	28	---		---	---	Reserved
0100111	39	27	39	PICPU_PRIM_VME-7	0 - 3	PL7	Primary VME I/O Level 7 / PCI A
0100110	38	26	38	PICPU_EXP_VME-7	0 - 3	---	Global Expansion VME I/O Level 7 / PCI A

Table A-3. PowerMAXION Computer System Interrupt Levels (Cont.)

LEVEL			PIN NO.	TUNABLE NAME	PROC NO.	IPL SYMBOL	PowerMAXION INTERRUPT DEFAULT
BINARY	DEC	HEX					
0100101	37	25	---		---	---	Local Expansion PCI A0
0100100	36	24	---		---	---	Reserved
0100011	35	23	35	PICPU_PRIM_VME-6	0 - 3	PL6	Primary VME I/O Level 6
0100010	34	22	34	PICPU_EXP_VME-6	0 - 3	---	Global Expansion VME I/O Level 6 / PCI A
0100001	33	21	---		---	---	Local Expansion PCI A1
0100000	32	20	32	PICPU_EMBEDDED_ SCSI-X	0 - 3	---	Global Embedded SCSI
0011111	31	1f	---		---	---	Local Embedded SCSI
0011110	30	1e	---		---	---	Reserved
0011101	29	1d	29	PICPU_PRIM_VME-5	0 - 3	PL5	Primary VME I/O Level 5
0011100	28	1c	28	PICPU_EXP_VME-5	0 - 3	---	Global Expansion VME I/O Level 5 / PCI A
0011011	27	1b	---		---	---	Local Expansion PCI A2
0011010	26	1a	26	PICPU_EMBEDDED_ ENET-X	0 - 3	---	Global Embedded Ethernet
0011001	25	19	---		---	---	Local Embedded Ethernet
0011000	24	18	---		---	---	Reserved
0010111	23	17	23	PICPU_EMBEDDED_ UART-X	0 - 3	---	Global Embedded UART
0010110	22	16	---		---	---	Local Embedded UART
0010101	21	15	---		---	---	Reserved
0010100	20	14	20	PICPU_PRIM_VME-4	0 - 3	PL4	Primary VME I/O Level 4
0010011	19	13	19	PICPU_EXP_VME-4	0 - 3	---	Global Expansion VME I/O Level 4 / PCI A
0010010	18	12	---		---	---	Reserved
0010001	17	11	---		---	---	Reserved
0010000	16	10	16	PICPU_PRIM_VME-3	0 - 3	PL3	Primary VME I/O Level 3
0001111	15	0f	15	PICPU_EXP_VME-3	0 - 3	---	Global Expansion VME I/O Level 3 / PCI B
0001110	14	0e	---		---	---	Reserved
0001101	13	0d	---		---	---	Reserved
0001100	12	0c	12	PICPU_PRIM_VME-2	0 - 3	PL2	Primary VME I/O Level 2

Table A-3. PowerMAXION Computer System Interrupt Levels (Cont.)

LEVEL			PIN NO.	TUNABLE NAME	PROC NO.	IPL SYMBOL	PowerMAXION INTERRUPT DEFAULT
BINARY	DEC	HEX					
0001011	11	0b	11	PICPU_EXP_VME-2	0 - 3	---	Global Expansion VME I/O Level 2 / PCI C
0001010	10	0a	---		---	---	Reserved
0001001	09	09	---		---	---	Reserved
0001000	08	08	8	PICPU_PRIM_VME-1	0 - 3	PL1	Primary VME I/O Level 1
0000111	07	07	7	PICPU_EXP_VME-1	0 - 3	---	Global Expansion VME I/O Level 1 / PCI D
0000110	06	06	---		---	---	Reserved
0000101	05	05	---		---	---	Reserved
0000100	04	04	4		0 - 3	---	Soft Clock
0000011	03	03	---		---	---	Reserved
0000010	02	02	---		---	PLSWTCH	Context Switch
0000001	01	01	---		---	---	Reserved
0000000	00	00	---		---	PL0 (PLBASE)	Reserved

Model HN6800 Interrupt Pins

The interrupt pins for the Model HN6800 are listed in Table A-4. Each interrupt is assigned a unique priority level and vector. When two or more interrupt requests occur simultaneously, the interrupt logic requests processing per the priority schedule shown inTable A-4. The vector for each interrupt is also shown. (Refer to the *HN6800 Architecture* manual for additional information.)

Table A-4. HN6800 Computer System Interrupt Levels

LEVEL			PIN NO.	TUNABLE NAME	IPL	PROC NO	BOARD NO	HN6800 INTERRUPT DEFAULT
BINARY	DEC	HEX						
1111111	127	7f	96		---	0 - 7	0 - 3	Int-On-No-Int
1111110	126	7e	---		---	-------	----------	UNUSED
1111101	125	7d	00		---	0	0	Powerfail
1111100	124	7c	---		---	-------	-------	UNUSED
1111011	123	7b	01		---	0 - 7	0 - 3	Console Wakeup
1111010	122	7a	---		---	-------	---------	UNUSED

Table A-4. HN6800 Computer System Interrupt Levels (Cont.)

LEVEL			PIN NO.	TUNABLE NAME	IPL	PROC NO	BOARD NO	HN6800 INTERRUPT DEFAULT
BINARY	DEC	HEX						
1111001	121	79	---		spipi	-------	---------	UNUSED
1111000	120	78	---		---	-------	---------	UNUSED
1110111	119	77	02		---	0	0	Inter–Processor
1110110	118	76	03		---	1	0	Inter–Processor
1110101	117	75	04		---	2	1	Inter–Processor
1110100	116	74	05		---	3	1	Inter–Processor
1110011	115	73	06		---	4	2	Inter–Processor
1110010	114	72	07		---	5	2	Inter–Processor
1110001	113	71	08		---	6	3	Inter–Processor
1110000	112	70	09		---	7	3	Inter–Processor
1101111	111	6f 6e	10		---	0 – 1	0	System Fault
1101110	110	6d	11		---	2 – 3	1	System Fault
1101101	109	6c	12		---	4 – 5	2	System Fault
1101100	108		13		---	6 – 7	3	System Fault
1101011	107	6b	---		---	-------	---------	UNUSED
1101010	106	6a	---		---	-------	---------	UNUSED
1101001	105	69	---		---	-------	---------	UNUSED
1101000	104	68	14	PICPU_EDGE1-0	spl8	2 - 3	1	Edge 4 (Pin 0)
1100111	103	67	15	PICPU_EDGE0-1	---	0 - 1	0	Edge 0 (Pin 0)
1100110	102	66	16	PICPU_EDGE3-0	---	6 - 7	3	Edge 12 (Pin 0)
1100101	101	65	17	PICPU_EDGE2-0	---	4 - 5	2	Edge 8 (Pin 0)
1100100	100	64	18	PICPU_EDGE0-1	---	0 - 1	0	Edge 1 (Pin 1)
1100011	99	63	19	PICPU_EDGE1-1	---	2 - 3	1	Edge 5 (Pin 1)
1100010	98	62	20	PICPU_EDGE2-1	---	4 - 5	2	Edge 9 (Pin 1)
1100001	97	61	21	PICPU_EDGE3-1	---	6 - 7	3	Edge 13 (Pin 1)
1100000	96	60	22	PICPU_RTC_1C0	---	2 - 3	1	RTC 5 (Timer 0)
1011111	95	5f 5e	23	PICPU_RTC_0C0	---	0 - 1	0	RTC 0 (Timer 0)
1011110	94	5d	24	PICPU_RTC_0C2	---	0 - 1	0	RTC 2 (Timer 2)
1011101	93	5c	25	PICPU_RTC_0C1	---	0 - 1	0	RTC 1 (Timer 1)
1011100	92	5b	26	PICPU_RTC_0C3	---	0 - 1	0	RTC 3 (Timer 3)
1011011	91	5a	27	PICPU_RTC_0C4	---	0 - 1	0	RTC 4 (Timer 4)
1011010	90	59	28	PICPU_RTC_1C2	---	2 - 3	1	RTC 7 (Timer 2)
1011001	89	58	29	PICPU_RTC_1C1	---	2 - 3	1	RTC 6 (Timer 1)
1011000	88	57	30	PICPU_RTC_2C0	---	4 - 5	2	RTC 8 (Timer 0)
1010111	87	56	31	PICPU_RTC_2C2	---	4 - 5	2	RTC 10 (Timer 2)
1010110	86	55	32	PICPU_RTC_2C1	---	4 - 5	2	RTC 9 (Timer 1)
1010101	85	54	33	PICPU_RTC_3C0	---	6 - 7	3	RTC 11 (Timer 0)
1010100	84		34	PICPU_RTC_3C1	---	6 - 7	3	RTC 12 (Timer 2
1010011	83	53	35	PICPU_EDGE0-2	---	0 - 1	0	Edge 2 (Pin 2)
1010010	82	52	36	PICPU_RTC-3C2	---	6 - 7	3	RTC 13 (Timer 1)

Table A-4. HN6800 Computer System Interrupt Levels (Cont.)

LEVEL			PIN NO.	TUNABLE NAME	IPL	PROC NO	BOARD NO	HN6800 INTERRUPT DEFAULT
BINARY	DEC	HEX						
1010001	81	51	37	PICPU_EDGE0-3	---	0 - 1	0	Edge 3 (Pin 3)
1010000	80	50	38	PICPU_EDGE1-2	---	2 - 3	1	Edge 6 (Pin 2)
1001111	79	4f	39	PICPU_EDGE2-2	---	4 - 5	2	Edge 10 (Pin 2)
1001110	78	4e	40	PICPU_EDGE3-2	---	6 - 7	3	Edge 14 (Pin 2
1001101	77	4d	41	PICPU_EDGE1-3	---	2 - 3	1	Edge 7 (Pin 3)
1001100	76	4c	42	PICPU_EDGE2-3	---	4 - 5	2	Edge 11 (Pin 3)
1001011	75	4b	43	PICPU_EDGE3-3	---	6 - 7	3	Edge 15 (Pin 3)
1001010	74	4a	---		---	-------	---------	UNUSED
1001001	73	49	44		---	0	0	Hard Clock (60Hz)
1001000	72	48	45		---	1	0	Hard Clock (60Hz)
1000111	71	47	46		---	2	1	Hard Clock (60Hz)
1000110	70	46	47		---	3	1	Hard Clock (60Hz)
1000101	69	45	48		---	4	2	Hard Clock (60Hz)
1000100	68	44	49		---	5	2	Hard Clock (60Hz)
1000011	67	43	50		---	6	3	Hard Clock (60Hz)
1000010	66	42	51		---	7	3	Hard Clock (60Hz)
1000001	65	41	---		---	-------	----------	UNUSED
1000000	64	40	---		---	-------	----------	UNUSED
0111111	63	3f	52	PICPU_PRIM_HVME-7	spl7	0 - 7	0 - 3	(H)VME Level 7 (Primary)
0111110	62	3e	53	PICPU_PRIM_VME-7	---	0 - 7	0 - 3	VME Level 7 (Secondary)
0111101	61	3d	---		---	-------	---------	UNUSED
0111100	60	3d	---		---	-------	---------	UNUSED
0111011	59	3b	54	PICPU_PRIM_HVME-6	spl6	0 - 7	0 - 3	(H)VME Level 6 (Primary)
0111010	58	3a	55	PICPU_PRIM_VME-6	---	0 - 7	0 - 3	VME Level 6 (Secondary)
0111001	57	39	56	PICPU_EXT_PORT-0A6	---	0 – 1	0	Port 0A6 (SCSI)
0111000	56	38	57	PICPU_EXT_PORT-1A6	---	2 – 3	1	Port 1A6 (SCSI)
0110111	55	37	58	PICPU_EXT_PORT-2A6	---	4 – 5	2	Port 2A6 (SCSI)
0110110	54	36	59	PICPU_EXT_PORT-3A6	spltty	6 – 7	3	Port 3A6 (SCSI)
0110101	53	35	60	PICPU_PRIM_HVME-5	spl5	0 – 7	0 - 3	(H)VME Level 5 (Primary)
0110100	52	34	61	PICPU_PRIM_VME-5	---	0 – 7	0 - 3	VME Level 5 (Secondary)
0110011	51	33	62	PICPU_UART-0	---	0 – 1	0	Console UART
0110010	50	32	63	PICPU_UART-1	---	2 – 3	1	Console UART
0110001	49	31	64	PICPU_UART-2	---	4 – 5	2	Console UART
0110000	48	30	65	PICPU_UART-3	---	6 – 7	3	Console UART

Table A-4. HN6800 Computer System Interrupt Levels (Cont.)

LEVEL			PIN NO.	TUNABLE NAME	IPL	PROC NO	BOARD NO	HN6800 INTERRUPT DEFAULT
BINARY	DEC	HEX						
0101111	47	2f	66	PICPU_EXT_PORT-0B5	---	0 – 1	0	Port 0B5
0101110	46	2e	67	PICPU_EXT_PORT-1B5	---	2 – 3	1	Port 1B5
0101101	45	2d	68	PICPU_EXT_PORT-2B5	---	4 – 5	2	Port 2B5
0101100	44	2c	69	PICPU_EXT_PORT-3B5	---	6 – 7	3	Port 3B5
0101011	43	2b	70	PICPU_EXT_PORT-0C5	---	0 – 1	0	Port 0C5 (Ethernet)
0101010	42	2a	71	PICPU_EXT_PORT-1C5	---	2 – 3	1	Port 1C5 (Ethernet)
0101001	41	29	72	PICPU_EXT_PORT-2C5	---	4 – 5	2	Port 2C5 (Ethernet)
0101000	40	28	73	PICPU_EXT_PORT-3C5	---	6 – 7	3	Port 3C5 (Ethernet)
0100111	39	27	74	PICPU_EXT_PORT-0D5	---	0 – 1	0	Port 0D5
0100110	38	26	75	PICPU_EXT_PORT-1D5	---	2 – 3	1	Port 1D5
0100101	37	25	76	PICPU_EXT_PORT-2D5	---	4 – 5	2	Port 2D5
0100100	36	24	77	PICPU_EXT_PORT-3D5	---	6 – 7	3	Port 3D5
0100011	35	23	78	PICPU_PRIM_HVME-4	spl4	0 – 7	0 - 3	(H)VME Level 4 (Primary)
0100010	34	22	79	PICPU_PRIM_VME-4	---	0 – 7	0 - 3	VME Level 4 (Secondary)
0100001	33	21	---		---	-------	---------	UNUSED
0100000	32	20	---		---	-------	---------	UNUSED
0011111	31	1f	80	PICPU_PRIM_HVME-3	spl3	0 – 7	0 - 3	(H)VME Level 3 (Primary)
0011110	30	1e	81	PICPU_PRIM_VME-3	---	0 – 7	0 - 3	VME Level 3 (Secondary)
0011101	29	1d	---		---	-------	---------	UNUSED
0011100	28	1c	---		---	-------	---------	UNUSED
0011011	27	1b	82	PICPU_PRIM_HVME-2	spl2	0 – 7	0 - 3	(H)VME Level 2 (Primary)
0011010	26	1a	83	PICPU_PRIM_VME-2	---	0 – 7	0 - 3	VME Level 2 (Secondary)
0011001	25	19	---		---	-------	----------	UNUSED
0011000	24	18	---		---	-------	----------	UNUSED
0010111	23	17	84	PICPU_PRIM_HVME-1	spl1	0 – 7	0 - 3	(H)VME Level 1 (Primary)
0010110	22	16	85	PICPU_PRIM_VME-1	---	0 – 7	0 - 3	VME Level 1 (Secondary)
0010101	21	15	86		splnet	0 – 7	0 - 3	UNUSED
0010100	20	14	---		---	-------	----------	UNUSED
0010011	19	13	87	PICPU_SOFTCLOCK	splsoft	0 - 7	0 - 3	Soft Clock

Table A-4. HN6800 Computer System Interrupt Levels (Cont.)

LEVEL			PIN NO.	TUNABLE NAME	IPL	PROC NO	BOARD NO	HN6800 INTERRUPT DEFAULT
BINARY	DEC	HEX						
0010010	18	12	---		---	-------	----------	UNUSED
0010001	17	11	---		---	-------	----------	UNUSED
0010000	16	10	---		---	-------	----------	UNUSED
0001111	15	0f 0e	---		---	-------	----------	UNUSED
0001110	14	0d	---		---	-------	----------	UNUSED
0001101	13	0c	---		---	-------	----------	UNUSED
0001100	12	0b	---		---	-------	----------	UNUSED
0001011	11	0a	---		---	-------	----------	UNUSED
0001010	10		---		---	-------	----------	UNUSED
0001001	09	09	88		splswtch	0	0	Context Switch
0001000	08	08	89		---	1	0	Context Switch
0000111	07	07	90		---	2	1	Context Switch
0000110	06	06	91		---	3	1	Context Switch
0000101	05	05	92		---	4	2	Context Switch
0000100	04	04	93		---	5	2	Context Switch
0000011	03	03	94		---	6	3	Context Switch
0000010	02	02	95			7	3	Context Switch
0000001	01	01	---		---	-------	----------	UNUSED
0000000	00	00	97		---	0 - 7	0 - 3	Spurious

Power Hawk 610 Interrupts

Table A-5 lists all the Power Hawk supported VME devices and the respective interrupt priority levels. Table A-6 lists the Motorola MVME1604 interrupt priority levels. Note that unlike the Night Hawk and PowerMAXION interrupts, the Power Hawk interrupts are not configurable.

Table A-5. Power Hawk VME Devices Interrupt Priority Levels

Board Name	Interrupt Level
1553 V2 ABI	7
VIA	6
Interphase Condor	5
Interphase 4211 Peregrine (FDDI)	5
Motorola IEEE 488 GBIP	5

Table A-5. Power Hawk VME Devices Interrupt Priority Levels (Cont.)

Board Name	Interrupt Level
DR11W	5
SYSTECH HPS MUX	4
HSDE OR VME-BBC	4

Table A-6. MVME1604 Interrupt Priority Levels

LEVEL			INTERRUPT NAME
BINARY	DEC	HEX	
111111	63	3f	Powerfail
111101	61	3d	Console Wakeup (Abort Switch)
111010	58	3a	Sysfault
110110	54	36	RTC 1 (Tick Timer 1)
110101	53	35	RTC 0 (Tick Timer 0)
110010	50	32	Hard Clock
101111	47	2f	VME Level 7
101100	44	2c	Internal NCR SCSI
101010	42	2a	VME Level 6
100110	38	26	Floppy Controller
100011	35	23	Internal DEC Ethernet
100001	33	21	VME Level 5
011110	30	1e	UNUSED
011101	29	1d	UNUSED
011011	27	1b	Serial port 1
011010	26	1a	Serial port 2
011001	25	19	RTC 2 (Z8536 Timer 0), RTC 3 (Z8536 Timer 1), RTC 4 (Z8536 Timer 2), and Serial ports 3 and 4
010110	22	16	VME Level 4
010100	20	14	UNUSED
010011	19	13	UNUSED

Table A-6. MVME1604 Interrupt Priority Levels (Cont.)

LEVEL			INTERRUPT NAME
BINARY	DEC	HEX	
010000	16	10	VME Level 3
001110	14	0e	VME Level 2
001100	12	0c	VME Level 1
001011	11	0b	Softclock
000010	2	02	Reschedule

Glossary

This glossary defines terms used in the documentation. If a term has multiple definitions, each definition is numbered. Some terms may describe software not loaded on your system.

Terms in *italics* are defined in the glossary. If a term with multiple definitions is used in a definition, the applicable definition number appears in brackets (for example, [1]).

access
: the ability of any *subject* to communicate with any *object* or any other subject.

Access Control List (ACL)
: a component of *Discretionary Access Control* (DAC), consisting of one or more *user* entries, one or more *group* entries, and one other entry. ACLs permit a finer grain of discretionary *access* than the nine *access permission* bits available without the Enhanced Security Utilities package.

access permission
: components of *Discretionary Access Control*. These bits define the *access permissions* and can be changed by the file's owner via the **`chmod`** command. Running **`ls -l`** lists the permission bits before the file name(s).

address
: a number, label, or name that shows the location of information in the computer's *memory*.

administrative role
: any one of the roles defined in the Trusted Facilities Management database: *Auditor* (AUD), *Operator* (OP), *Security Operator* (SOP), *System Security Officer* (SSO), also known as the *Site Security Officer*, and the *Network Administrator* (NET).

a.out
: the default name of a compiled *object file*, pronounced “a-dot-out”. **`a.out`** is the default name produced by the **`cc`** command.

allocation unit
: a group of consecutive blocks on a file system that contain resource summaries, free resource maps, inodes, and data blocks. The “allocation unit” is equivalent to the **`ufs`** “cylinder group.”

archive
: 1. a collection of data gathered from several *files* into one file. 2. especially, collection gathered by **`ar`** for use as a library.

argument
: The element of a command line that specifies data on which a command is to operate. Arguments follow the command name and can include numbers, letters, or text strings. For instance, in the command **`lp -m`** myfile, **`lp`** is the command and **`myfile`** is the argument.

audit
: the act of recording all potentially *sensitive* and *security*-related transactions on a *trusted computer system*. The *System Administra-*

tors decide whether to install the auditing programs and which transactions to audit.

audit trail

the written record that reports on all potentially *sensitive* and *security*-related transactions on a *trusted computer system*. Only the *Trusted Computing Base* can write audit trail data.

Auditor (AUD)

an authorized individual entrusted with *secure audit* administrative duties on a *trusted computer system*. Auditor duties may include selecting events to be audited enabling the recording of events, analyzing the *audit trail*, and modifying or deleting auditing information.

authentication

1. verification of the *client* machine and login_name of an incoming request. 2. the mechanism by which the *Trusted Computing Base* verifies the identity of a *user*.

authorization

allowing or disallowing *user access* to a service.

automatic calling unit

a hardware *device* used to dial stored telephone numbers. This unit enables a system to contact another system over phone lines without manual intervention.

automatic data

data that is persistent only during the invocation of a procedure. It describes data belonging to a process. Automatic data occupies the stack segment. See *static data*.

bad block

a *sector* of a storage medium which cannot store data reliably.

bandwidth

a measurement of the amount of information that can be passed through a communication *channel* in a given amount of time. Bandwidth is usually in units of bits per second.

block

the basic unit of *buffering* in the *kernel*, 1024 bytes; see *indirect*, *logical*, and *physical blocks*.

block device

a *device* upon which a *file system* [1] can be *mounted*, typically a permanent storage device such as a disk drive, so called because data transfers to the device occur by *blocks*; see *character device*.

boot

the process by which the operating system is started. The *kernel* must bootstrap itself from secondary storage into an empty machine. No *login* [3] or *process* persists across a boot.

boot program

loads the *operating system* into *core*.

buffer

1. a staging area for input/output where arbitrary-length transactions are collected into convenient units. The *file system* [3] uses buffers, as does *stdio*. 2. to use buffers.

buffer pool

a region of storage available to the *file system* [3] for holding *blocks*. To make read and write operations independent of *device* blocks, all but *raw* input/output for *block devices* goes through the buffer pool.

cartridge tape

a storage medium that consists of a magnetic tape wound on spools housed in a plastic container.

category

a non-hierarchical, restrictive grouping of *objects* to which a name is applied (for example, “Project Alpha,” “Project Sigma,” “Project Phi”). The system supports up to 1024 categories. The non-hierarchical category and the hierarchical *classification* together constitute the *security level* to which a *sensitivity label* is applied; see *security level* and *sensitivity label*.

certification

the technical evaluation of a *trusted computer system*, by the *National Computer Security Center* (NCSC). This evaluation is part of the accreditation process establishing the extent a computing system's design and implementation meet the security requirements of the NCSC. From least secure to most secure, the levels of accreditation are D (untrusted), C1, C2, B1, B2, B3, and A1.

certify

the action which produces *certification*.

channel

a path or mechanism by which information is transferred within a computer system. On a *trusted computer system*, a channel offers a potential avenue of *compromise* to the *Trusted Computing Base* and to *sensitive information*, and it must be *trusted* or *audited*.

character device

a *device* on which a *file system* [1] cannot be *mounted*, such as a terminal or the *null device*.

child process

See *fork*.

classification

one of 256 levels of a hierarchy for grouping *objects* of like *sensitivity*, where 0 is the lowest level and 255 is the highest. The levels are usually known by the names associated with them (“Classified,” “Secret,” “Top Secret”). The hierarchical classification and the non-hierarchical *category* together constitute a *security level*; see *security level*.

client

a *host* that has *mounted* an *advertised resource* from another host in a *Network File Sharing* environment.

command

1. an instruction to the *shell*, usually to run a *program* [1] as a *child process*. 2. by extension, any *executable file*, especially a *utility program*.

command file

See *shell script.*

compromise

a violation of the *security policy* that causes potential or actual un*authorized* disclosure, modification, or destruction of *sensitive information*.

configuration

the arrangement of the software or hardware of a system, peripheral, or network as defined by the nature, number, and chief characteristics of its functional units.

configuration management

the identification of a system's hardware, software, firmware, documentation, test fixtures, and test documentation and changes made to them throughout the development and operational life of the system. The current configuration and revision level of all components should be known so these components can be effectively and productively integrated, managed, and used.

console terminal
the directly connected terminal used for communication between the operator and the computer.

controller
a *device* that directs the transmission of data over the data links of a *network*.

core
Core is a name commonly associated with primary memory, although very little memory is still “core”.A better term might be “primary memory”.

core file
a *core image* of a terminated *process* saved for debugging. A core file is created under the name **`core`** in the current directory of the process.

core image
a copy of all the *segments* of a running or terminated program. The copy may exist in main storage, the *swap area*, or a *core file*.

covert channel
a communications *channel* that allows a *process* to transfer or deduce *sensitive information* in violation of the *security policy* of a *trusted computer system*.

covert storage channel
a *covert channel* that involves the direct or indirect writing of a storage location by one *process* and the direct or indirect reading of that storage location by another process. Covert storage channels typically involve a finite resource that is shared by two *subjects* at different *security levels*.

covert timing channel
a *covert channel* in which one *process* signals information to another by modulating its own use of system resources in a way that affects the real response time observed by the second process.

crash
occurs when a hardware or software *error* condition occurs that causes the system to take itself out of service. For example, such conditions may occur when the system cannot allocate resources, manage *processes*, or respond to requests for system functions, or when the electrical power is unstable.

cron
a command that creates a `daemon` that invokes *commands* at specified dates and times; see *daemon*.

current directory
The directory in which you are presently working. You have direct access to all files and subdirectories contained in your current directory. The shorthand notation for the current directory is a dot (.).

cylinder
the set of all *tracks* on a *disk* that are the same distance from the axis about which the disk rotates.

daemon
a background process, often perpetual, that performs a system-wide public function. **`cron`** is an example.

data integrity
the consistency between stored data that has not been exposed to alteration or destruction (either accidental or malicious) and the source data.

destination
the remote system that will ultimately receive a *file* transferred over a *network*.

device
1. a *file* [2] that is not a plain *file* or a *directory,* such as a tape drive, or the *null device*; a *special file*. 2. a physical input/output unit.

device activation
the means by which a device is *accessed*. A device can be accessed only by an activated *Device Special File* that maps to it.

device allocation
device access by any one of several *kernel* mechanisms that support access to devices by *trusted* processes. The permission, usage, and protection strategies for accessing a device depend on the specific device, its intended use, and the programs that do the allocation.

Device Control Information (DCI)
information associated with a *device* when it is not in the *device disabled state*. DCI includes the *device level range*, the *multilevel* or *single-level* mode, and the activated *device special file* list.

Device Database (DDB)
the database that contains critical security attributes for *devices*, and information defining which *users* can use the device. (Administered via the **`admalloc`** command.)

device disabled state
the state in which a *device* is not accessible. The only operations allowed on a device in this state are the transitions to *device setup state* or *device enabled state* using the `devalloc` system call.

device driver
a module of the operating system that controls a specific input/output *device* in response to a request by a *subject* for input or output.

device enabled state
the state in which a *device* is fully operational at a specific *security level*, which cannot be changed even by a *privileged* process. From the enabled state, a device can be taken to the *device setup state* or the *device disabled state* by the `devstat` system call.

device level range
the range of *security levels* that a device can store or process.

device readiness state
one of three states, *device disabled*, *device setup*, or *device enabled*, that each *configured device* must be in. These states are used to assure proper allocation by *device allocation* mechanisms. A device can make the transition from any state to either of the others, but some restrictions apply.

device setup state
the state used by a *device allocation* program to prepare a *device* for use.

Device Special File (DSF)

the mechanism through which *processes* address *devices*. The DSF has the structure of a regular file but it lacks a data area and has only an inode; data in the inode indicates a path through various kernel tables that eventually maps to a single, logical device; although this logical device is conceptually a single data object, more than one DSF can map onto a single device; see *device allocation*.

diagnostic

a message printed at your terminal that identifies and isolates *program* errors.

directory

a *file* that comprises a catalog of *filenames* [2]. The organizing principle of the *file system* [2], a directory consists of *directory entries* which specify further *files* [2] (including directories), and constitutes a node of the *directory tree*.

directory entry

1. an association of a name with an *inode number* appearing as an element of a *directory*. 2. the name part of such an association.

directory hierarchy

the tree of all *directories*, in which each is reachable from the *root* via a chain of subdirectories.

directory tree

See *directory hierarchy*.

Discretionary Access Control (DAC)

one of the mechanisms controlling sharing *objects* among *subjects*. The DAC mechanism uses the object *owner*, the object *group*, the nine *access permission* bits. It also uses the *Access Control List* of an object to determine the discretionary access to the object; contrasts with *Mandatory Access Control*. see *access permissions*.

disk

A magnetic data storage device consisting of several round plates similar to phonograph records. Disks store large amounts of data and allow quick access to any piece of data.

diskette

a magnetic storage medium which is smaller and more flexible than a hard *disk*.

domain

a logical grouping of *hosts* in a *Remote File Sharing* environment. Each host in a domain relies on the same *domain name server*(s) for certain *resource* sharing and security services. Each domain has one *primary* and zero or more *secondary domain name servers*.

domain name server

a computer that creates and maintains the following information for *hosts* in a *Network File Sharing* domain: *advertised resources*, *host* names and passwords, names and addresses for name servers of other domains (optional), *host* user and group information used for *ID mapping* (optional).

dominate

a relationship between *security levels*; security level S1 is said to dominate security level S2 if the hierarchical *classification* of S1 is greater than or equal to that of S2, and if the non-hierarchical *categories* of S1 include those of S2 as a subset.

drive
the hardware device that holds magnetic disks, diskettes, and tapes while they are in use.

dump
a copy of the *core image* of the operating system.

encryption
the process of transforming data or text, usually by means of a cipher or substitution code, from its ordinary readable state to a superficially nonsense state. Changing the state of the information makes it much harder to *access* understandably and thus increases the confidentiality of the information.

environment
1. a set of strings, distinct from arguments, made available to a *process* when it *executes* [2] a *file*; the environment is usually inherited across **`exec(2)`** operations; see *exec*. 2. a specific environment maintained by the *shell*.

error
occurs when a hardware or software condition prevents the successful *execution* of a system or a user *process*.

error message
a message sent from the system when an *error* occurs.

exclusive access
a feature used mainly by *device allocation* programs allowing a *process*, or multiple processes, to exclude *access* to a *device* by other processes.

`exec`
a system call which allows the user to request the execution of another program.

executable file
1. an *object file* that is ready to be copied into the *address* space of a *process* to run as the code of that process. 2. a file that has execute *permission*, either an *executable file* [1] or a *shell script.*

execute
1. informally, to run a *program*. 2. to replace the text *segment* and data *segments* of a *process* with a given *program* [1].

expired password
a *password* that is invalid because it is older than the mandatory retirement age allotted it by the *system administrator.*

exploitable channel
a *channel* that is usable or detectable by *subjects* external to the *Trusted Computing Base*.

exportation
the extraction of information from one computer system for transportation to another computer system; a potential source of *trojan horses* and *viruses*; see *importation*.

FIFO
a named permanent *pipe* which allows two unrelated *processes* to exchange information using a pipe connection.

file
1. in general, a potential source of input or destination for output. 2. an *inode* and/or its associated contents, that is, a plain or ordinary file, *special file*, or *directory.* 3. a *directory entry*; several directory entries may name the same file [2]. 4. a plain file.

file descriptor
a conventional integer quantity that designates an *open file* [1].

filename
1. a *pathname*. 2. the last component name in a *pathname.*

file system

1. a collection of *files* that can be *mounted* on a block *special file*. Each file of a file system appears exactly once in the *i-list* of the file system and is accessible via some *path* from the *root* directory of the file system. 2. the collection of all *files* on a computer. 3. the part of the kernel that deals with file systems [1].

filter

a *program* [1] that reads from the *standard input* and writes on the *standard output*, so called because it can be used as a data-transformer in a *pipeline*.

fixed privilege

a set of *privileges* that are always given to *processes* created from the executable file, independent of the *process privileges* from the creating, parent process. See *inheritable privilege*.

flaw

an error of commission, omission, or oversight in a *trusted computer system* that allows protection mechanisms to be bypassed.

floppy key

a copy of the default *firmware password* for an AT&T 3B2 Computer on a *diskette*. It may be used to reset the password to its original value.

flush

to empty a *buffer*, for example to throw away unwanted input/output on *interrupt* or to release output from *stdio*.

`fork`

to split one *process* into two, the *parent process* and *child process*, with separate, but initially identical, *text*, data, and *stack segments*.

formatting

the process of imposing an addressing scheme on a *disk*. This includes the establishment of a *VTOC*, and the mapping of both sides of the disk into *tracks* and *sectors*.

free list

in a *file system* [1], the list of *blocks* that are not occupied by data.

functional channel

a *channel* whose existence can be determined from the system call interface, independent of the internal implementation. Such *channels* include inode information, file system directories containing elements whose level is different from that of the directory, signals, FIFO pipes, IPC mechanisms, I/O device control mechanisms, file and record locking, and text locking.

`getty`

one of a series of *processes* that connect the user to the UNIX system. `getty` is invoked by **`init`**, and in turn invokes **`login`**. See *init* and *login*.

group

1. a set of *permissions* for access to a *file*; see *owner*. 2. a set of *user ID*s that may assume the privileges of a *group* [1]. 3. the *group ID* of a file. 4. a set of *users* who share a common project or purpose, for example, all users performing basic systems operations may belong to group `operator` and all users working on the “Alpha Project” may belong to group **`alpha`**. The *login names* of group members are listed together in **`/etc/group;`** the members of a group share the same group *access permissions.*

group ID (GID)

an integer value, usually associated with one or more *login names*; as the *user ID* of a process becomes the *owner* of files created by a process, so the group ID of a process becomes the *group* [3] of such files. The *Trusted Computing Base* uses the group ID along with the *user ID* and *Access Control Lists* to enforce *Discretionary Access Control*.

hole

a gap in a plain file caused by seeking while writing [see **lseek(2)**]; **read(2)** takes data in holes to be zero; a *block* in a hole occupies no space in its *file system*.

host

a computer that is configured to share *resources* in a *Remote File Sharing* environment.

ID mapping

a means of setting the permissions each remote user and group will have for a *host*'s *advertised resources* in a *Remote File Sharing* environment.

identification

via the *login name*, the mechanism by which the *Trusted Computing Base* recognizes a *user* as legitimate.

i-list

the index to a *file system* [1] listing all the *inodes* of the file system; see *inode number*.

importation

the inclusion of information on one computer system after transportation from another computer system; a potential source of *trojan horses* and *viruses*; see *exportation*.

indirect blocks

data blocks that are not directly referenced by a *inode*. The inode has three *addresses* that indirectly reference data blocks.

inheritable privilege

privileges which a *parent process* can pass to a process it creates.

init

a general *process* spawner which is invoked as the last step in the *boot* procedure. It regularly checks a table that defines which processes should run at what *run level*.

inode

an element of a *file system* [1]. An inode specifies all properties of a particular *file* [2] and locates the file's contents, if any.

inode number, i-number

the position of an *inode* in the *i-list* of a *file system* [1].

instruction

See *address*.

integrity

in a *file system*, the quality of being without errors due to *bad blocks*. 1. in a *file system*, the quality of being without errors due to *bad blocks*. 2. the logical completeness of the hardware and software that implement the *security* mechanisms; the consistency of the data structures and the accuracy of the stored data.

interface programs

shell scripts furnished with spooler software (for the LP print service) which processes data sent by a user to a printer.

interrupt

1. a *signal* that normally terminates a *process*, caused by a break or an interrupt character. 2. a signal generated by a hardware condition or a peripheral *device*. 3. any *signal*.

IPC
an acronym for interprocess communication.

kernel
resident code that implements the *system calls*.

kernel address space
a portion of memory used for data and code addressable only by the *kernel*.

label
See *sensitivity label*.

least privilege
the principle in the *security policy* that restricts a *process* to the minimum *privileges* necessary to perform a given function at a given time; see *privilege*, *process privileges*, *maximum set of process privileges*, and *working set of process privileges*.

level
See *security level*.

level alias
See *security level alias*.

line discipline
a module to handle protocol or data conversion for a *stream*. A line discipline, unlike a *filter*, is part of the *kernel*.

link
1. to add an entry for an existing *file* to a *directory*; converse of unlink. 2. by extension, a *directory entry*. 3. any link[2] or *symbolic link* for a given *inode*.

link count
the number of *directory entries* that pertain to an *inode*. A *file* ceases to exist when its link count becomes zero and it is not *open*.

load device
designates the physical *device* from which a program will be loaded into main *memory.*

log files
contain records of transactions that occur on the system. For example, software that *spools* generates various log files.

logical block
a unit of data as it is handled by the software.

login
1. the act of *logging on*: a *user* identifies himself or herself to the operating system by supplying a *login name* (and the correct *password*) pin response to prompts; the system then creates a `process` that runs on the user's behalf. 2. an abbreviated form of *login name*. 3. the *program* that controls logging on. 4. by extension, the computing session that follows a login [1].

login name
a unique character string that identifies a *user* to the operating system.

machine alias
an abbreviated notation for a collection of remote machines; machine aliases can be used on command lines to simplify the specification of destination machines.

maintenance mode
the state of a *secure system* when only a *trusted administrator* has access to it, usually for the purpose of running diagnostics or performing maintenance on the system or the software.

Mandatory Access Control (MAC)
the mechanism controlling *access* of *objects* among *subjects*. Unlike *Discretionary Access Control*, MAC is controlled by the

Trusted Computing Base and is independent of the object owner's ability to grant discretionary access. The TCB enforces MAC by the assignment of a *security level* (recorded in or on a *sensitivity label*) to the information contained in the object.

maximum set of process privileges

the set of all *process privileges* a process is allowed to use. A process may delete privileges from its maximum set, but it may not add them. If the process does not have a process privilege, it will not be allowed to perform the privileged function; see *process privileges* and *working set of process privileges.*

memory

1. See *core image*. 2. physical memory representing the available space in main memory; *programs* are either *swapped* or *paged* into physical memory for *execution*. 3. virtual memory management techniques permitting *programs* to treat *disk* storage as an extension of main memory.

mode, file mode

the *permissions* of a *file*; referred to by a 3-digit octal number, for example “a 755 file”; see **`chmod(1)`**.

mount

extends the *directory hierarchy* by associating the *root* of a *file system* [1] with a *directory entry* in an already mounted file system; the converse is to unmount using the `umount` system call.

mounted file system range

the range of the *security levels* of the information that can be stored on a file system.

multilevel device

a device which separately and simultaneously processes, displays, or stores data of different *security levels* without risk of *compromise*. To accomplish this separation of data, *sensitivity labels* are normally stored on the same physical medium and in the same form (that is, machine-readable or human-readable) as the data. These devices include disks that contain *multilevel directories*, archival and storage media, and administrative devices such as **`/dev/kmem`** and **`/dev/mem.`**

Multilevel Directory (MLD)

a directory, such as **`/tmp,`** which allows *untrusted* processes to create files at different *security levels* within that same directory.

namelist

See *symbol table*.

National Computer Security Center (NCSC)

the agency of the government of the United States of America which *certifies* computing systems as being *trusted.*

network

the hardware and software that connect computer systems, permitting electronic communication between systems and associated peripherals.

Network Administrator (NET)

the administrator responsible for *networking* when the Enhanced Security Utilities are installed.

networking

for computer systems, means sending data from one system to another over some communications medium (coaxial cable, phone lines, and so on). Common networking services include *file* transfer, remote *login*, and remote *execution*.

node name

a character string that identifies a single computer in a *network*. The node name may contain up to eight characters; it resides in the `NODE` parameter.

normal operations

The usual, multi-user state of a *trusted computer system*, when *users* can *login*.

null device

a *device* [1] that always yields end-of-file on reading and discards all data on writing.

object

anything that contains or receives information. *access* to an object implies access to the information it contains. Examples of objects are: *processes*, *blocks*, *files*, *directories*, *pipes*, *programs*, bits, bytes, processors, keyboards, clocks, and printers.

object file

a *file* of machine language code and data. Object files are produced from source programs by compilers and from other object files and libraries by the link editor. An object file that is ready to run is an *executable file* [1].

Object Reuse (OR)

the reassignment, from one process to another, of a storage *object* (such as a segment of memory, disk block, or magnetic tape) that contains data owned by the original “owner process.” In order to be securely reassigned, such media must contain no residual data from the previous owner.

open file

1. the destination for input or output obtained by *opening* a *file* or creating a *pipe*. Open files are shared across *forks* and persist across *executes* [2]; see *file descriptor*. 2. a file that has been opened; however, an *open file* [1] need not exist in a *file system* [1], and a *file* [2] may be the destination of several *open files* simultaneously.

Operator (OP)

An *untrusted* administrator. The operator performs regular, non-*secure* activities, such as starting and stopping the system, and generating raw accounting data, but not changing *passwords*, *login levels*, or other *security*-related login parameters.

operating system

the *program* for managing the resources of the computer. It provides tools for handling basic operations such as input/output procedures and process scheduling, thus rendering unnecessary *user programs* for maintaining these functions.

other

1. a set of *permissions* regulating *access* to a *file* by *processes* with a *user ID* different from that of the *owner*, and a *group ID* different from that of the *group* for that file. 2. the customary name of the default *group* [2] assigned upon *login*.

output

information that has been *exported* by the *Trusted Computing Base*.

overt channel

a communications path within a network that is designed for the authorized transfer of data.

owner

the *user ID* of the *process* that created a *file*.

page

a fixed length, 4096-byte block that has a virtual *address*, and that can be transferred between main and secondary storage.

paging

a memory-management technique in which *programs* are organized into *pages* that can be transferred between main and secondary storage by the virtual handler (or paging *daemon*).

parent process

See `fork`.

partitions

units of storage space on disk.

password

a character string typed by a *user* during the **`login`** process. The user is prompted to type a password after entering his or her *login name*. By correctly supplying the password, a user proves he or she has both authorization to *access* the account and the *privileges* associated with the *login name*. By keeping the password a secret, users help protect a system from unauthorized access.

path, pathname

a chain of *directory* names ending in a *filename*, that shows the location of a file in the *file system*. The location is expressed as a list of the *directories* that must be traversed to access the specified file. There are two types of paths: relative and full. Relative paths show how to reach the location of a file from the directory in which you're currently working; full paths, from the **`/`** (root) directory.

Thus, for example, to designate a file called **`file_x`** while you're working in a directory called **`B`**, use the following relative path: **`B/C/file_x.`** Because the first element of a full path is the **`/`** (root) directory, the full path for this example is **`/A/B/C/file_x.`**

permission

1. a right to access a *file* for one, two, or three purposes: to *read*, *write*, and/or *execute* the file. Because permission for each function is granted separately to *owner*, *group*, and *other*, nine permissions are associated with every file. 2. an abbreviated form of *permission bit*.

permission bit

a permission, so called because each permission is encoded into one bit in an *inode*.

physical block

a unit of data as actually stored and manipulated.

physical memory

See *memory.*

`pipe`

a direct stream connection between *processes*, whereby data written on an *open file* in one process becomes available for reading in another.

pipeline

a sequence of *programs* [1] connected by *pipes.*

plain file

A *file*, containing text or data, that is not executable. See *executable file*.

polling

the interrogation of *devices* by the *operating system* to avoid contention, determine operation status, or ascertain readiness to send or receive data.

ports

the point of physical connection between a peripheral *device* (such as a terminal or a printer) and the device *controller* (ports board), which is part of the computer hardware.

primary name server

the computer on which the *Remote File Sharing Utilities* are installed and maintained.

privilege

the ability to override system restrictions and thus *access* or use a system call, function, resource, or program; see *least privilege* and *process privileges*.

privileged process

a *process* that has at least one of the *process privileges*.

process

a sequence of computations characterized by a *core image* with instruction location counter, current directory, *open files*, control terminal, *user ID*, and *group ID*; a program in execution. A process is characterized by a single current execution point and address space. A process is a *subject* when it requests some action; an *object*, when it receives data or a signal.

process ID

an integer that identifies a *process*.

process number

See *process ID*.

process privilege

authorization for a process to perform sensitive operations.

profile

1. an optional *shell script*, **`.profile`**, used by the *shell* on *logging in* to establish an *environment* [2] and other working conditions for a particular *user*. 2. to collect an instruction location counter value histogram of a *process*.

program

1. an *executable file*. 2. a *process*.

programmer

a *user* who writes code. A programmer needs to know which system commands, system calls, and libraries are affected by the *Trusted Computing Base* and how they are affected.

pseudo-device

a special *object* in the system used to perform special-purpose system functions; examples of pseudo-devices are memory pseudo-devices and pseudo-terminals. Pseudo-devices have associated *Device Special Files* and the same access semantics as physical devices, but they may not have associated device hardware.

queue

a line (or list) of items waiting for service in a system.

RAID

Redundant Arrays of Independent Disks.

raw device
a *block device* for which read and write operations are synchronized to natural records of the physical *device* (not *buffered*).

read
a fundamental operation that results only in the flow of information from an *object* to a *subject*.

read access
authorization for a *subject* to *read* the information in an *object*.

reboot
See *boot*.

reference monitor concept
at levels B3 and above (see *certification*), equivalent to the *Trusted Computing Base*; the smallest section of code that needs to be *trusted* and that mediates all security-relevant decisions.

region
a group of machine *addresses* that refer to a base address.

release
one of multiple, sequentially produced versions of a software product, each of which contains improvements on the last. a distribution of fixes or new functions for an existing software product.

Remote File Sharing
a software utilities package that enables the users of multiple computers to share *resources* across a *network*.

resource
a directory that is *advertised* in a *Remote File Sharing* environment. When a *resource* is *mounted* on a *client*, its contents (files, devices, named pipes, and subdirectories) are potentially available to users on the *client*.

resource channels
are *channels* inherent in the design of the operating system kernel, consisting mainly of internal data structures shared by *processes* of different *security levels*; resource channels include the system file table, the system lock table, disk blocks, STREAMS buffers, **`proc`** and u structures, file system inodes, and IPC structures.

re-tension
the process of rewinding a tape in a *cartridge tape device* to get the amount of tautness necessary for accurate recording of data.

role
a named description of a given administrative function regardless of the individual identity of the administrators to which the function is assigned. It is a convenient way of assigning privileges to a group of users who will be doing the identical type of work (also see “administrative role”).

root
1. a distinguished directory that constitutes the origin of the *directory hierarchy* in a *file system* [1]. 2. specifically, the origin for the *file system* [2], with the conventional *pathname* “**`/`**”. 3. the origin of the directory hierarchy in a *file system* [1].

rotational gap
the gap between the actual *disk* locations of blocks of data belonging to the same *file*. The rotational gap compensates for the continuous, high-speed rotation of the disk so that when the controller is ready to reference the next physical block the read-write head is positioned correctly at the beginning of that block.

run level
See *system* state.

schedule
to assign resources (main store and CPU time) to *processes*.

scheduler
a permanent *process* (with *process number* 0 and associated *kernel* facilities) that determines the order in which various processes are executed.

search path
in the *shell*, a list of *pathnames* that determine the directories in which a desired file will be sought and the order in which they will be investigated. The command name is prefixed with members of the search path in turn until a pathname of an *executable file* [2] results; the search path is given by the shell variable `PATH`.

secondary name server
a *host* that is configured to take over *domain name server* responsibilities temporarily in case the *primary name server* goes down.

sector
A 512-byte portion of a *track* that can be accessed by magnetic disk heads in the course of a predetermined *rotational* displacement of the storage device.

secure
describes the assurance that the hardware and software comprising a computer system are *trusted*, and that the mechanisms, policies, and procedures governing the use of that computer system are enforced to protect against unauthorized *access* to or modification or destruction of the *Trusted Computing Base* or *sensitive information*.

Secure Attention Key (SAK)
a character or asynchronous line condition (for example, break, line drop) that a *user* must enter to invoke the *Trusted Path*; note that several keys can be required to enter one character, as with a control character. The *system administrators* define the default SAK for each *terminal* on the system; the active SAK is the one in effect for a given terminal.

security
1. the mechanisms and techniques that control *access* to a *trusted computer system*. 2. the assurance that these mechanisms and techniques are functioning correctly. 3. the state of being *secure*.

Security Administrator
an authorized administrator responsible for the security of computer system; see *System Security Officer*.

security level
the combination of a hierarchical *classification* and a set of non-hierarchical *categories* that represents the *sensitivity* of information.

security level alias
a name applied to a *security level* for easy reference.

Security Operator (SOP)
a *trusted* administrator. The Security Operator performs routine, daily activities similar to those performed by the *Operator*; however, certain of these activities are *security*-related and thus restricted to the Security Operator. Because these activities are routine and would require the efforts of more than one person,

there can be several Security Operators. The Security Operator can perform all of the activities of the Operator.

security policy
the set of rules, concepts, and practices that regulate the organization, management, protection, and distribution of *sensitive information*. With the Enhanced Security Utilities, the security policy states that a *subject* can read only those *objects* that are *dominated* by the subject's *level*, and can write only to those objects at the same level.

segment
a contiguous address space range of a *process* with consistent *read*, *write*, and *execute* capabilities. For example, three common segments are (i) the text segment, containing read-only instructions and data; (I) the data segment, containing static data that is explicitly initialized; (iii) the bss segment, containing static data that is initialized to zero.

semaphore
an IPC facility which allows two or more processes to be synchronized.

sensitive
describes any information, any action, or any part of a *trusted computer system* which involves *security* and to which *access* is restricted.

sensitive information
information that must be protected because its unauthorized disclosure, alteration, loss, or destruction will cause harm or damage.

sensitivity
the degree to which information is restricted and requires special authorization in order to be *accessed*.

sensitivity label
data that represents the *security level* of an *object*, that describes the *sensitivity* of the data in the object, and that can be changed only by a *privileged process* via `chlvl`. The *Trusted Computing Base* enforces *Mandatory Access Control* by comparing the sensitivity label of an object with the security level of a *subject* trying to *access* the object. Often mistakenly referred to as “security label.”

server
a *host* that actively shares one of its *advertised resources* with another *host* in a *Remote File Sharing* environment.

set user ID
a special *permission* for an *executable file* [1] that causes a *process* executing it to have the access rights of the *owner* of the file. The owner's *user ID* becomes the effective user ID of the process, distinguished from the real user ID under which the process began.

set user ID bit
the associated *permission bit*.

shared memory
an IPC facility that allows two or more processes to share the same data space.

shared text
Shared text is a text segment, one copy of which may be used simultaneously by more than one process.

shell
1. the program **`sh(1)`**, which causes other programs to be executed on command; the shell is usually started on a user's behalf when the user *logs in*. 2. by analogy, any program started upon logging in.

shell script
an executable *file* of *commands* taken as input to the *shell*.

signal
an exceptional occurrence that causes a *process* to terminate or divert from the normal flow of control; see *interrupt*, *trap*.

single-level device
a device that processes, display, or stores data of a single *security level* at any one time. Because the device need not separate data of different security levels, *sensitivity labels* do not need to be stored with the data. These devices include *terminals*, storage devices such as magnetic tape, and administrative devices such as **`/dev/console`**.

single-user
a *system state* in which only one user is supported.

Site Security Officer
See *System Security Officer.*

source file
1. the uncompiled version of a *program*. 2. generally, the unprocessed version of a *file*.

special file
an *inode* that designates a *device*, further categorized as either (i) a block special file describing a *block device*, or (ii) a character special file describing a *character device*.

spoofing
the act of creating a hoax on a computer system, usually with malicious intent; generally, a program that masquerades as part of the *Trusted Computing Base* in order to trick a *user* or other *subject* into revealing *sensitive information*. A typical instance of spoofing is a program that appears to be the *login* program and tricks the user into supplying a *password*.

spool
to collect and serialize output from multiple *processes* competing for a single output service.

spool area
a *directory* in which a spooler collects work.

spooler
a *daemon* that spools.

stack
a *segment* of the *address* space into which data and subroutine linkage information is allocated in last-in-first-out fashion.

standard error
one of three files described below under *standard output*.

standard input
the second of three files described below under *standard output.*

standard output
open files with *file descriptors* 0, 1, and 2, and *stdio* names `stdin`, `stdout`, and `stderr`, respectively. Where possible, utilities read from the standard input, write on the standard output, and place error comments on the standard error file.

static data
static represents a condition persistent throughout a process. Said of data. Static data occupies the data segment and the bss segment.

startup

See *boot*

`stdio`

(standard I/O) a library of efficient and portable I/O routines; the header file **`/usr/include/stdio.h`** contains definitions and declarations; see **`stdio(3S)`**.

sticky bit

a *permission* flag that identifies a file as a *sticky file*.

sticky file

a special *permission* for a shared text file that causes a copy of the text *segment* to be retained in the *swap area* to improve system response.

storage object

an *object* that supports both *read* and *write accesses*.

stream

A kernel aggregate created by connecting STREAMS components, resulting from an application of the STREAMS mechanism. The primary components are the Stream head, the driver, and zero or more pushable modules between the Stream head and driver.

striping

A technique that spreads data across several physical disks or using stripes. The data is allocated alternately to the stripes within the subdisks of each mirror.

striped VP

Striped VP (virtual partition) divides the contiguous data of the VP into slices.

subdirectory

A *directory* pointed to by a directory one level above it in the file system organization; also called a child directory.

subject

a *process*; anything that causes information to flow among *objects* or that changes the system state. The initial process for a *user* is the shell invoked by *login*. A process is assigned a *security level* when it is created. The shell is established with the user's *security level*, as determined by the Identification & Authentication mechanism. Additional processes inherit the security level of the invoking process.

super-block

the second *block* in a *file system* [1], which describes the allocation of space in the file system; see *boot block*.

super user

user ID 0, which can access any *file* regardless of *permissions* and can perform certain privileged *system calls*, for example, setting the clock.

Super User Module

Super User Module (SUM). This privilege mechanism functions exactly the same as the superuser, a UID-based mechanism common to earlier releases of the OS. It also provides additional flexibility by allowing the association of fixed privileges with executable files. This mechanism works by using: 1) a list of system privileges, 2) a working and maximum set of privileges for each process on the system, and 3) a fixed set of privileges for files.

swap

to move the *core image* of an executing program between main and secondary storage to make room for other *processes*.

swap area

the part of secondary store to which *core images* are *swapped*; the swap area is disjointed from the *file system*.

symbolic link

an *inode* that contains the *pathname* of another *inode*. References to the symbolic link become references to the named inode.

symbol table

information in an *object file* about the names of data and functions in that file; the symbol table and *address* relocation information are used by the link editor to compile *object files* and by debuggers.

System Administration

when capitalized, refers to the menu interface for administrative tasks that's invoked through the **`sysadm(1)`** command.

System Administrator

a person who supervises the running of a computer system (or part of one). With Enhanced Security Utilities, the secure system administrators include the *Site Security Officer*, the *Auditor*, the *Security Operator*, and the *Network Administrator.*

system calls

1. the set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated. 2. the system primitives invoked by user *processes* for system-dependent functions, such as I/O, process creation, and so on.

system console

the directly connected terminal used for communication between the operator and the computer.

system high

1. the highest *security level* supported by a *trusted computer system* at a given time; 2. a system where every *object* is assigned the highest possible security level.

system low

the lowest *security level* supported by a *trusted computer system* at a given time.

system name

an up-to-eight character name for the system; resides in the `SYS` parameter.

System Security Officer (SSO)

an *authorized* and *trusted system administrator* responsible for the security of a *trusted computer system*. The System Security Officer is also known as the *Security Administrator.* With the exception of security *audit*, The SSO performs the most critical *security*-related functions, excluding security *audit*.

system state

one of five configurations of the operating system for which a predetermined set of processes can be executed. The operating system is running in one of these states at any given time. (Also known as “run level” and “init state.”)

table

an array of data in which each item may be uniquely identified by one or more arguments.

terminal

an I/O device connected to a computer system, usually consisting of a keyboard with a video display or printer, allowing a *user* to give the computer instructions and to receive information in response. On *trusted computer systems*, we assume that only directly connected “dumb” terminals are used, because “smart” terminals may *compromise security*.

text file

(or ASCII file) a *file* which contains ASCII code.

text segment

part of a contiguous range of the address space of a process, a text segment is occupied by executable code. A text segment is a contiguous range of the address space of a process with consistent store access capabilities.

track

an addressable ring of *sectors* on a *disk* or *diskette*; each disk or diskette has a predefined number of concentric tracks, which allows the disk head to properly access *sectors* of data.

trap

a method of detecting and interpreting certain hardware and software conditions via software. A trap is set to catch a *signal* (or *interrupt*), and determine what course of action to take.

trap door

a hidden software or hardware mechanism that permits the protection mechanisms of the *Trusted Computing Base* to be circumvented.

trojan horse

software with apparent or actual useful functions that contains additional, often hidden functions that surreptitiously violate the protection mechanisms of the *Trusted Computing Base* by exploiting the legitimate *authorizations* and *privileges* of invoking *processes*.

trusted

describes a component of the computer system that can be relied upon to enforce the system's *security policy*.

trusted computer system

a computer system on which a *security policy* is enforced by a *Trusted Computing Base*; also known as a *trusted system*.

Trusted Computing Base (TCB)

the Trusted Computing Base is the totality of the software, firmware, and hardware that enforces a *security policy*; the ability of the TCB to enforce a security policy correctly depends solely on the mechanisms within the TCB and on the correct input by the *system administrators* of parameters related to *security policy*.

Trusted Path (TP)

the link between the *user* and the *trusted computer system* during *login* processing; the Trusted Path is *trusted* because every step in establishing the link between the user and the TCB is trusted; the Trusted Path is intended to help prevent *spoofing*; see *Secure Attention Key*, *identification*, and *authentication*.

trusted process channels

channels of low *bandwidth* that occur when *trusted processes*, usually those associated with *privileged*, non-administrative commands, are unwittingly used as *covert channels*.

trusted software

the software portion of the *Trusted Computing Base*.

trusted system
See *trusted computer system.*

tunable parameters
variables used to set the sizes and thresholds of the various control structures of the *operating system.*

tuning
1. modifying the *tunable parameters* to improve system performance. 2. reconfiguring an *operating system* so that modifications are incorporated into an *executable* version of the system.

untrusted
not *trusted*; a component of the computer system whose incorrect or malicious execution cannot affect the *security* of the system, because it has no control over security-related actions.

user
any person who interacts directly with a computer system. With Enhanced Security Utilities, a user needs to know what system commands are affected by the *Trusted Computing Base* and how they are affected. The user's access to the TCB is determined by the *System Security Officer.*

user ID (UID)
an integer value, usually associated with a *login name*. The user ID of a *process* becomes the *owner* of files created by the process and descendent (*forked*) processes. The *Trusted Computing Base* uses the user ID along with the *group ID* and *Access Control Lists* to enforce *Discretionary Access Control*.

utility, utility program
a standard, generally useful, permanently available *program*.

verification
the process by which an authority, such as the *National Computer Security Center*, compares two levels of system specification for proper correspondence. For example, the security policy model is compared with the design documentation, the design documentation with the source code, and the source code with the object code.

version
a separate *program* product, based on an existing one, but containing significant new code or new functions.

virus
a particularly malicious form of *trojan horse* that reproduces itself or other executable code.

virtual memory
See *memory.*

VTOC
the Volume Table Of Contents is a table that shows how the *partitions* on the *disk* are allocated.

working set of process privileges
a subset of the *maximum set of process privileges*, the working set consists of those *privileges* that a *process* has while performing a given function at a given time; the process can freely alter its working set by using the `procpriv` system call, but each working set must be a subset of the *maximum set*; see *maximum set of process privileges*.

write
a fundamental operation that results only in the flow of information from a *subject* to an *object*.

write access authorization for a *subject* to *write* information to an *object*.

Volume 1 Index

Volume 2 Index

Spine for 2” Binder

Product Name: 0.5" from top of spine, Helvetica, 36 pt, Bold

Volume Number (if any): Helvetica, 24 pt, Bold

Volume Name (if any): Helvetica, 18 pt, Bold

Manual Title(s): Helvetica, 10 pt, Bold, centered vertically within space above bar, double space between each title

Bar: 1" x 1/8" beginning 1/4" in from either side

Part Number: Helvetica, 6 pt, centered, 1/8" up

PowerMAX OS

User/Administrator

System Administration

0890429

Volume 1

Copy the contents of this chapter into the Table of Contents.

Illustrations

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Screens

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Tables

Do not include this document in the final book.

Copy the contents of this chapter into the Table of Contents.

Do not include this document in the final book.