AIX 5L Version 5.1

IBM

System Management Concepts: Operating System and Devices

AIX 5L Version 5.1

System Management Concepts: Operating System and Devices

Third Edition (April 2001)

Before using the information in this book, read the general information in "Appendix B. Notices" on page 217.

This edition applies to AIX 5L Version 5.1 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to Publications Department, Internal Zip 9561, 11400 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

Contents

About This Book

This book contains information for understanding the concepts of the tasks that you perform as a system administrator, as well as the tools provided for system management. Use this book together with the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

Who Should Use This Book

This book is for persons performing system management on the computer and operating system. Readers of this book are expected to know basic operating system commands.

It is assumed that you are familiar with the information and concepts presented in the following publications:

- *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.1 System User's Guide: Operating System and Devices*
- *AIX 5L Version 5.1 System User's Guide: Communications and Networks*
- *AIX 5L Version 5.1 Installation Guide*.

How to Use This Book

This book is organized to help you quickly find the information you need. The tasks of each chapter are arranged in the following order:

- Overview of topic/task group
- Configuration tasks
- Maintenance tasks
- Troubleshooting.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Related Publications

- *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.1 Installation Guide*
- *AIX 5L Version 5.1 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.1 Communications Programming Concepts*
- *AIX 5L Version 5.1 Kernel Extensions and Device Support Programming Concepts*
- *AIX 5L Version 5.1 Files Reference*
- *Performance Toolbox Version 2 and 3 for AIX: Guide and Reference*
- *AIX 5L Version 5.1 Network Installation Management Guide and Reference*
- *Distributed SMIT 2.2 for AIX: Guide and Reference*
- *AIX 5L Version 5.1 Performance Management Guide*
- *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*.

Chapter 1. System Management

System management is the task of an individual who is usually referred to, in UNIX literature, as the system administrator. Unfortunately, only a few system administrator activities are straightforward enough to be correctly called administration. This and related guides are intended to help system administrators with their numerous duties.

Topics covered in this chapter are:
- "The System Administrator's Objectives"
- "Unique Aspects of System Management" on page 2
- "Unique Aspects of System Management" on page 2
- "man Command" on page 4
- "Operating System Updates" on page 4.

The System Administrator's Objectives

The system administrator has three main objectives:
- Ensure that the system does its job effectively and efficiently
- Ensure that the information stored on the system is secure from intentional or accidental destruction
- Administer the system owner's rules for the use of the system.

To achieve these objectives the system administrator must understand more than just the structure and interaction of the hardware and software under their control. They must also understand the interconnected environment in which almost all current systems exist and the effects that environment has on function and performance of the local system.

A System Is More Than a Computer

A contemporary computer system includes a number of hardware, software, and information elements that must work cooperatively if the system is to satisfy the needs of its users. The main elements and their management functions are:
- Fixed-disk drives
  - Control the grouping and subdivision of disk space
  - Control the location of data and programs for optimum performance
  - Control the amount of space allocated for different purposes.
- Application programs
  - Control the use of sensitive or costly programs
  - Install and performance-tune major applications.
- Application data
  - Control access to sensitive data
  - Ensure that appropriate backup measures are taken.
- Individual computer processors and memory
  - Ensure that resources are used in accordance with the priorities of the organization
  - Control access to the system by individuals and groups
  - Tune the operating system for optimal use of the available resources.
- Local area networks
  - Ensure that networks are tuned for optimum performance
  - Control network addressing mechanisms.
- Local terminals

  - Control the connection of terminals to processors
  - Ensure that terminals and processors are set up for maximum performance.
- Connections to other networks
  - Ensure that bridges and gateways to other networks are configured correctly
  - Ensure that interaction with remote networks does not degrade local systems.
- Access to and from remote systems
  - Control the access permissions in both directions
  - Monitor and performance-tune the workload imposed by remote connections.
- Access to remotely owned data
  - Control the methods and availability of access.

Unique Aspects of System Management

This operating system provides its own particular version of system-management support in order to promote ease of use and to improve security and integrity. This chapter presents information on these unique features:

- "Available Interfaces"
- "Unique Features of the Operating System" on page 3
- "man Command" on page 4.

Available Interfaces

In addition to conventional command line system administration, this operating system provides the following interfaces:

- System Management Interface Tool (SMIT), a menu-based user interface that constructs commands from the options you choose and executes them.

  With SMIT, you can:

  - Install, update, and maintain software
  - Configure devices
  - Configure disk storage units into volume groups and logical volumes
  - Make and extend file systems and paging space
  - Manage users and groups
  - Configure networks and communication applications
  - Print
  - Perform problem determination
  - Schedule jobs
  - Manage system resources and workload
  - Manage system environments.

  See "System Management Interface Tool (SMIT) Overview" on page 167 for more information on managing your system with SMIT.
- Web-based System Manager , an object-oriented graphical user interface that supports the same system management tasks as SMIT, but eases system management tasks by:
  - Reducing user errors through error checking and dialog design
  - Offering step-by-step procedures for new or complex tasks
  - Offering advanced options for more experienced administrators
  - Making it easier to visualize complex data or relationships among system objects
  - Monitoring system activity and alerting the administrator when predefined events occur
  - Providing context-sensitive helps, overviews, tips, and links to online documentation

Web-based System Manager can be run from an AIX desktop or through a secure remote connection to any client with a Java-enabled browser. Multiple AIX machines can be managed from a single Web-based System Manager console.

Unique Features of the Operating System

Following are brief discussions of unique system-management features of the operating system.

Logical Volume Manager

The Logical Volume Manager (LVM) allows logical volumes to span multiple physical volumes. Data on logical volumes appears to be contiguous to the user, but can be discontiguous on the physical volume. This allows file systems, paging space, and other logical volumes to be resized or relocated, span multiple physical volumes, and have their contents replicated for greater flexibility and availability.

For more detailed information, see the “Logical Volume Storage Overview” on page 47.

System Resource Controller

The System Resource Controller (SRC) provides a set of commands and subroutines for creating and controlling subsystems and is designed to minimize the need for human intervention in system processing. It provides a mechanism to control subsystem processes by using a command-line C interface. This allows you to start, stop, and collect status information on subsystem processes with shell scripts, commands, or user-written programs.

For more detailed information, see the “System Resource Controller Overview” on page 155.

Object Data Manager

The Object Data Manager (ODM) is a data manager intended for the storage of system data. Many system management functions use the ODM database. Information used in many commands and SMIT functions is stored and maintained as objects with associated characteristics. System data managed by ODM includes:

- Device configuration information
- Display information for SMIT (menus, selectors, and dialogs)
- Vital product data for installation and update procedures
- Communications configuration information
- System resource information.

Software Vital Product Data

Certain information about software products and their installable options is maintained in the Software Vital Product Data (SWVPD) database. The SWVPD consists of a set of commands and Object Data Manager (ODM) object classes for the maintenance of software product information. The SWVPD commands are provided for the user to query (**lslpp**) and verify (**lppchk**) installed software products. The ODM object classes define the scope and format of the software product information that is maintained.

The **installp** command uses the ODM to maintain the following information in the SWVPD database:

- Name of the installed software product
- Version of the software product
- Release level of the software product, which indicates changes to the external programming interface of the software product
- Modification level of the software product, which indicates changes that do not affect the external programming interface of the software product
- Fix level of the software product, which indicates small updates that are to be built into a regular modification level at a later time
- Fix identification field
- Names, checksums, and sizes of the files that make up the software product or option

- Installation state of the software product: available, applying, applied, committing, committed, rejecting, or broken.

Workload Manager (WLM)

Workload Manager (WLM) lets you create different classes of service for jobs, as well as specify attributes for those classes. These attributes specify minimum and maximum amounts of CPU, physical memory, and disk I/O throughput to be allocated to a class. WLM then assigns jobs automatically to classes using class assignment rules provided by a system administrator. These assignment rules are based on the values of a set of attributes for a process. Either the system administrator or a privileged user can also manually assign jobs to classes, overriding the automatic assignment.

man Command

The **man** command is used mainly to access reference information on commands, subroutines, and files. For example, to view information on the **gprof** command, enter:

```
>man gprof
```

Most of the information displayed is actually taken from formatted HTML files. Many system managers find using the **man** command more convenient than starting a web browser session when they simply need to find out about a certain flag or the syntax of a given command.

For more information on the **man** command, see the *AIX 5L Version 5.1 Commands Reference*. Also see “Online Documentation and man Command for BSD 4.3 System Managers” on page 209.

Operating System Updates

For detailed information on software and termed service updates, see Installing Optional Software and Service Updates in *AIX 5L Version 5.1 Installation Guide*.

Chapter 2. Starting and Stopping the System

This chapter explains system startup activities such as booting, creating boot images or files for starting the system, and setting the system run level. The chapter also focuses on stopping the system using the **reboot** and **shutdown** commands.

Topics covered in this chapter are:

Starting the System

When the base operating system starts, the system initiates a complex set of tasks. Under normal conditions, these tasks are performed automatically. For additional information about restarting the system, see:

- “Understanding the Boot Process” on page 6
- Diagnosing Boot Problems.

Booting the System

There are some situations when you want to instruct the system to reboot; for example, to cause the system to recognize newly installed software, to reset peripheral devices, to perform routine maintenance tasks like checking file systems, or to recover from a system hang or crash. For information on these procedures, see:

- Booting an Uninstalled System
- Rebooting a Running System
- Booting a System That Crashed.

Creating Boot Images

When the system is first installed, the **bosboot** command creates a boot image from a RAM (random access memory) disk file system image and the operating system kernel. The boot image is transferred to a particular media such as the hard disk. When the machine is rebooted, the boot image is loaded from the media into memory.

For more information, see ″Creating Boot Images″ .

Identifying and Changing the System Run Level

The system run level specifies the system state and defines which processes are started. For example, when the system run level is 3, all processes defined to operate at that run level are started. Near the end of the system boot phase of the boot process, the run level is read from the initdefault entry of the **/etc/inittab** file. The system run level can be changed with the **init** command. The **/etc/inittab** file contains a record for each process that defines run levels for that process. When the system boots, the **init** command reads the **/etc/inittab** file to determine which processes to start. For information on these procedures, see:

- Identifying System Run Levels
- Changing System Run Levels

- Changing the /etc/inittab File.

Understanding the Boot Process

During the boot process, the system tests the hardware, loads and runs the operating system, and configures devices. To boot the operating system, the following resources are required:

- A *boot image* that can be loaded after the machine is turned on or reset.
- Access to the root (**/**) and **/usr** file systems.

There are three types of system boots:

Hard Disk Boot	A machine is started for normal operations with the key in the Normal position. For more information, see “Understanding System Boot Processing”.
Diskless Network Boot	A diskless or dataless workstation is started remotely over a network. A machine is started for normal operations with the key in the Normal position. One or more remote file servers provide the files and programs that diskless or dataless workstations need to boot.
Service Boot	A machine is started from a hard disk, network, tape, or CD-ROM with the key set in the Service position. This condition is also called *maintenance mode*. In maintenance mode, a system administrator can perform tasks such as installing new or updated software and running diagnostic checks. For more information, see “Understanding the Service Boot Process” on page 8.

During a hard disk boot, the boot image is found on a local disk created when the operating system was installed. During the boot process, the system configures all devices found in the machine and initializes other basic software required for the system to operate (such as the Logical Volume Manager). At the end of this process, the file systems are mounted and ready for use. For more information about the file system used during boot processing, see “Understanding the RAM File System” on page 9.

The same general requirements apply to diskless network clients. They also require a boot image and access to the operating system file tree. Diskless network clients have no local file systems and get all their information by way of remote access.

Understanding System Boot Processing

Most users perform a hard disk boot when starting the system for general operations. The system finds all information necessary to the boot process on its disk drive.

When the system is started by turning on the power switch (a cold boot) or restarted with the **reboot** or **shutdown** commands (a warm boot), a number of events must occur before the system is ready for use. These events can be divided into the following phases:

1. Read Only Storage (ROS) Kernel Init Phase
2. Base Device Configuration Phase
3. System Boot Phase.

ROS Kernel Init Phase

The ROS kernel initialization phase involves the following steps:

1. The On-Chip Sequencer (OCS) bring-up microprocessor (BUMP) checks to see if there are any problems with the system motherboard. Control is passed to ROS, which performs a power-on self-test (POST).

2. The ROS initial program load (IPL) checks the user boot list, a list of available boot devices. This boot list can be altered to suit your requirements using the **bootlist** command. If the user boot list in Non-Volatile Random Access Memory (NVRAM) is not valid or if a valid boot device is not found, the default boot list is then checked. In either case, the first valid boot device found in the boot list is used for system startup. If a valid user boot list exists in NVRAM, the devices in the list are checked in order. If no user boot list exists, all adapters and devices on the bus are checked. In either case, devices are checked in a continuous loop until a valid boot device is found for system startup.

   Note: The system maintains a default boot list located in ROS and a user boot list stored in NVRAM, for a normal boot. Separate default and user boot lists are also maintained for booting from the Service key position.
3. When a valid boot device is found, the first record or program sector number (PSN) is checked. If it is a valid boot record, it is read into memory and is added to the IPL control block in memory. Included in the key boot record data are the starting location of the boot image on the boot device, the length of the boot image, and instructions on where to load the boot image in memory.
4. The boot image is read sequentially from the boot device into memory starting at the location specified in the boot record. The disk boot image consists of the kernel, a random access memory (RAM) file system, and base customized device information.
5. Control is passed to the kernel, which begins system initialization.
6. ROS runs **init**, which runs phase 1 of the **rc.boot** script.

When the kernel initialization phase is completed, base device configuration begins.

Base Device Configuration Phase

The **init** process starts the **rc.boot** script. Phase 1 of the **rc.boot** script performs the base device configuration, and it includes the following steps:

1. The boot script calls the **restbase** program to build the customized Object Database Manager (ODM) database in the RAM file system from the compressed customized data.
2. The boot script starts the configuration manager, which accesses phase 1 ODM configuration rules to configure the base devices.
3. The configuration manager starts the **sys**, **bus**, **disk**, SCSI, and the Logical Volume Manager (LVM) and rootvg volume group (RVG) configuration methods.
4. The configuration methods load the device drivers, create special files, and update the customized data in the ODM database.

System Boot Phase

The System Boot Phase involved the following steps:

1. The **init** process starts phase 2 running of the **rc.boot** script. Phase 2 of **rc.boot** includes the following steps:
   a. Call the **ipl_varyon** program to vary on the rootvg volume group (RVG).
   b. Mount the hard disk file systems onto their normal mount points.
   c. Run the **swapon** program to start paging.
   d. Copy the customized data from the ODM database in the RAM file system to the ODM database in the hard disk file system.
   e. Exit the **rc.boot** script.
2. After phase 2 of **rc.boot**, the boot process switches from the RAM file system to the hard disk root file system.
3. Then the **init** process runs the processes defined by records in the **/etc/inittab** file. One of the instructions in the **/etc/inittab** file runs phase 3 of the **rc.boot** script, which includes the following steps:
   a. Mount the **/tmp** hard disk file system.

b. Start the configuration manager phase 2 to configure all remaining devices.
c. Use the **savebase** command to save the customized data to the boot logical volume
d. Exit the **rc.boot** script.

At the end of this process, the system is up and ready for use.

Understanding the Service Boot Process

Occasions might arise when a service boot is needed to perform special tasks such as installing new or updated software, performing diagnostic checks, or for maintenance. In this case, the system starts from a bootable medium like the CD-ROM or tape drive, a network, or from the disk drive with the key in the Service position.

The service boot sequence of events is similar to the sequence of a normal boot.

1. The On-Chip Sequencer (OCS) checks to see if there are any problems with the system motherboard.
2. Control is passed to ROS, which performs a power-on self-test (POST).
3. ROS checks the user boot list. You can use the **bootlist** command to alter the user boot list to suit your requirements. If the user boot list in NVRAM is not valid or if no valid boot device is found, the default boot list is checked. In either case, the first valid boot device found in the boot list is used for system startup.

   Note: For a normal boot, the system maintains a default boot list located in ROS, and a user boot list stored in NVRAM. Separate default and user boot lists are also maintained for booting from the Service key position.
4. When a valid boot device is found, the first record or program sector number (PSN) is checked. If it is a valid boot record, it is read into memory and is added to the initial program load (IPL) control block in memory. Included in the key boot record data are the starting location of the boot image on the boot device, the length of the boot image, and the offset to the entry point to start running when the boot image is in memory.
5. The boot image is read sequentially from the boot device into memory, starting at the location specified in the boot record.
6. Control is passed to the kernel, which begins running programs in the RAM file system.
7. The Object Data Manager (ODM) database contents determine which devices are present, and the **cfgmgr** command dynamically configures all devices found, including all disks which are to contain the root file system.
8. If a CD-ROM, tape, or the network is used to boot the system, the rootvg volume group (RVG) is not varied on, since the RVG might not exist (as is the case when installing the operating system on a new system). Network configuration can occur at this time. No paging occurs when a service boot is performed.

At the end of this process, the system is ready for installation, maintenance, or diagnostics.

Note: If the system is booted from the hard disk, the RVG is varied on, the hard disk root file system and the hard disk user file system are mounted in the RAM file system, a menu is displayed which allows you to enter various diagnostics modes or single-user mode. Selecting single-user mode allows the user to continue the boot process and enter single-user mode, where the **init** run level is set to ″S″. The system is then ready for maintenance, software updates, or running the **bosboot** command.

Understanding the RAM File System

The random access memory (RAM) file system, part of the boot image, is totally memory-resident and contains all programs that allow the boot process to continue. The files in the RAM file system are specific to the type of boot.

A service boot RAM file system might not have the logical volume routines, because the rootvg volume group might not need to be varied on. During a hard disk boot, however, it is desirable that the rootvg volume group be varied on and paging activated as soon as possible. Although there are differences in these two boot scenarios, the structure of the RAM file system does not vary to a great extent.

The **init** command on the RAM file system used during boot is actually the simple shell (**ssh**) program. The **ssh** program controls the boot process by calling the **rc.boot** script. The first step for **rc.boot** is to determine from what device the machine was booted. The boot device determines which devices are to be configured on the RAM file system. If the machine is booted over the network, the network devices need to be configured so that the client file systems can be remotely mounted. In the case of a tape or CD-ROM boot, the console is configured to display the BOS install menus. After the **rc.boot** script finds the boot device, then the appropriate configuration routines are called from the RAM file system. The **rc.boot** script itself is called twice by the **ssh** program to match the two configuration phases during boot. A third call to **rc.boot** occurs during a disk or a network boot when the real **init** command is called. The **inittab** file contains an **rc.boot** stanza that does the final configuration of the machine.

The RAM file system for each boot device is also unique due to the various types of devices to be configured. A prototype file is associated with each type of boot device. The prototype file is a template of files making up the RAM file system. The **bosboot** command uses the **mkfs** command to create the RAM file system using the various prototype files. See the **bosboot** command for more details.

Understanding the Shutdown Process

There are several controlled situations in which you might want to shut down your system:

- After installing new software or changing the configuration for existing software
- When a hardware problem exists
- When the system is irrevocably hung
- When system performance is degraded
- When the file system is possibly corrupt.

System Hang Detection

All processes (also known as threads) run at a priority. This priority is numerically inverted in the range 0-126. Zero is highest priority and 126 is the lowest priority. The default priority for all threads is 60. The priority of a process can be lowered by any user with the **nice** command. Anyone with root authority can also raise a process's priority.

The kernel scheduler always picks the highest priority runnable thread to put on a CPU. It is therefore possible for a sufficient number of high priority threads to completely tie up the machine such that low priority threads can never run. If the running threads are at a priority higher than the default of 60, this can lock out all normal shells and logins to the point where the system appears hung.

The System Hang Detection (SHD) feature provides a mechanism to detect this situation and allow the system administrator a means to recover. This feature is implemented as a daemon (**shdaemon**) that runs at the highest process priority. This daemon queries the kernel for the lowest priority thread run over a specified interval. If the priority is above a configured threshold, the daemon can take one of several actions. Each of these actions can be independently enabled, and each can be configured to trigger at any priority and over any time interval. The actions and their defaults are:

	Action	Default Enabled	Default Priority	Default Timeout	Default Device
1)	Log an error	no	60	2	
2)	Console message	no	60	2	/dev/console
3)	High priority login shell	yes	60	2	/dev/tty0
4)	Run a command at high priority	no	60	2	
5)	Crash and reboot	no	39	5	

For more information on system hang detection, see System Hang Management in *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

Chapter 3. Security

This chapter explains advanced system security topics. System security is explained in *AIX 5L Version 5.1 System User's Guide: Operating System and Devices*.

Topics covered in this chapter are:
- "Security Administration"
- "System Security Guidelines" on page 15
- "Trusted Computing Base Overview" on page 19
- "Pluggable Authentication Module (PAM) Overview" on page 24
- "Auditing Overview" on page 30
- "PKCS #11 Overview" on page 33 (POWER-based platform only)
- "LDAP Exploitation of the Security Subsystem" on page 34

Security Administration

Precise system administration is vital to maintaining the security of information resources on a computer system. Security is based on establishing and maintaining correct access control and accountability policies. It is the responsibility of the administrator to configure the following aspects of security:

Managing Protected Resources with Access Control	Addresses the privacy, integrity, and availability of information on your system
"Identification and Authentication" on page 12	Determines how users are identified and how their identities are authenticated
"Trusted Computing Base Overview" on page 19	Enforces the information security policies of the system
"Auditing Overview" on page 30	Records and analyzes events that take place on the system.

Aspects of Computer Security

The primary goal of security is the detection and prevention of security violations on a system. Computer security includes the following fundamental aspects.
- "User Administration"
- "Identification and Authentication" on page 12.

User Administration

User administration consists of creating users and groups and defining their attributes. A major attribute of users is how they are authenticated. Users are the primary agents on the system. Their attributes control their access rights, environment, how they are authenticated, and how, when, and where their accounts can be accessed.

Groups are collections of users who can share the same access permissions for protected resources. A group has an ID and is composed of members and administrators. The creator of the group is usually the first administrator.

The operating system supports the standard user attributes usually found in the **/etc/passwd** and **/etc/group** files, such as:

Authentication Information	Specifies the password
Credentials	Specifies the user identifier, principal group, and the supplementary group ID

Environment	Specifies the home or shell environment.

If desired, the operating system allows for greater control with extended attributes. Security information can be separately protected from public access.

Some users and groups can be defined as administrative. These users and groups can be created and modified only by the root user.

User Account Control

Each user account has a set of associated attributes. These attributes are created from default values when a user is created using the **mkuser** command. They can be altered by using the **chuser** command. The following are examples of user attributes:

ttys	Limits certain accounts to physically secure areas
expires	Manages student or guest accounts; also can be used to turn off accounts temporarily
logintimes	Restricts when a user can log in. For example, a user may be restricted to accessing the system only during normal business hours.

The complete set of user attributes is defined in the **/usr/lib/security/mkuser.default**, **/etc/security/user**, **/etc/security/limits,** and **/etc/security/lastlog** files. Several of these attributes control how a user can log in and they can be configured to lock the user account (prevent further logins) under specified conditions.

Once the user account is locked, the user is not be able to log in until the system administrator resets the user **unsuccessful_login_count** attribute in the **/etc/security/lastlog** file to be less than the value of login retries. This can be done using the following **chsec** command:

```
chsec -f /etc/security/lastlog -s username -a
     unsuccessful_login_count=0
```

The defaults can be changed by using the **chsec** command to edit the default stanza in the appropriate security file, such as the **/etc/security/user**, **/usr/lib/security/mkuser.default**, or **/etc/security/limits** files. Many of the defaults are defined to be the standard behavior.

Identification and Authentication

Identification and authentication establish a user's identity. Each user is required to log in to the system. The user supplies the user name of an account and a password, if the account has one (in a secure system, all accounts must either have passwords or be invalidated). If the password is correct, the user is logged in to that account; the user acquires the access rights and privileges of the account. The **/etc/passwd** and **/etc/security/passwd** files maintain user passwords.

Alternative methods of authentication are integrated into the system by means of the **SYSTEM** attribute that appears in **/etc/security/user**. For instance, the Distributed Computing Environment (DCE) requires password authentication but validates these passwords in a manner different from the encryption model used in /**etc/passwd** and **/etc/security/passwd**. Users who authenticate by means of DCE can have their stanza in **/etc/security/user** set to `SYSTEM=DCE`.

Other **SYSTEM** attribute values are **compat**, **files**, and **NONE**. The **compat** token is used when name resolution (and subsequent authentication) follows the local database, and if no resolution is found, the Network Information Services (NIS) database is tried. The **files** token specifies that only local files are to be used during authentication. Finally, the **NONE** token turns off method authentication. To turn off all authentication, the **NONE** token must appear in the **SYSTEM** and **auth1** lines of the user's stanza.

Other acceptable tokens for the **SYSTEM** attribute can be defined in **/usr/lib/security/methods.cfg**.

Note: The root user is always authenticated by means of the local system security file. The **SYSTEM** attribute entry for the root user is specifically set to `SYSTEM = "compat"` in **/etc/security/user**.

See the *AIX 5L Version 5.1 System User's Guide: Operating System and Devices* for more information on protecting passwords.

Configuring Password Restrictions

Proper password management can only be accomplished through user education. But to provide some additional security, the operating system provides configurable password restrictions. These allow the administrator to constrain the passwords chosen by users and to force passwords to be changed regularly. These restrictions are recorded in the **/etc/security/user** attribute file and are enforced whenever a new password is defined for a user. All password restrictions are defined per user. By keeping restrictions in the default stanza of the **/etc/security/user** file, the same restrictions are enforced on all users. To maintain password security, all passwords must be similarly protected.

The operating system also provides a method for administrators to extend the password restrictions. Using the **pwdchecks** attribute of the **/etc/security/user** file, an administrator can add new subroutines (known as methods) to the password restrictions code. Thus, local site policies can be added to and enforced by the operating system. See "Extending Password Restrictions" on page 14 for more information.

Apply password restrictions sensibly. Attempts to be too restrictive, such as limiting the password space (making guessing easier) or forcing the user to select difficult-to-remember passwords (which are then written down) can jeopardize password security. Ultimately, password security rests with the user. Simple password restrictions, coupled with sensible guidelines and an occasional audit (checking current passwords to see if they are unique), are the best policy.

The restrictions that can be applied are:

minage	Minimum number of weeks that must pass before a password can be changed.
maxage	Maximum number of weeks that can pass before a password must be changed.
maxexpired	Maximum number of weeks beyond **maxage** that a password can be changed before administrative action is required to change the password. (Root is exempt.)
minalpha	Minimum number of alphabetic characters the new password must contain.
minother	Minimum number of nonalphabetic characters the new password must contain. (Other characters are any ASCII printable characters that are nonalphabetic and are not national language code points).
minlen	Minimum number of characters the new password must contain.

Note: The minimum length of a password on the system is **minlen** or **minalpha** plus **minother**, whichever is greater. The maximum length of a password is eight characters. **minalpha** plus **minother** must never be greater than eight. If **minalpha** plus **minother** is greater than eight, then **minother** is reduced to eight minus **minalpha**.

maxrepeats	Maximum number of times a character can be used in the new password.
mindiff	Minimum number of characters in the new password that must be different from the characters in the old password.
histexpire	Number of weeks that a user is unable to reuse a password.
histsize	Number of previous passwords that cannot be reused.

Note: If both **histexpire** and **histsize** are set, the system retains the number of passwords required to satisfy both conditions up to the system limit of 50 passwords per user. Null passwords are not retained.

dictionlist	List of dictionary files checked when a password is changed. Dictionary files contain passwords that are not allowable.
pwdchecks	List of external password restriction methods that are used when a password is changed.

Recommended, Default, and Maximum Password Attribute Values

Restriction Values	Advised Values	Default Values	Maximum Values
minage	0	0	52
maxage	8	0	52
maxexpired	4	-1	52
minalpha	4	0	8
minother	1	0	8
minlen	6	0	8
mindiff	3	0	8
maxrepeats	1	8	8
histexpire	26	0	260*
histsize	0	0	50
dictionlist	NA	NA	NA
pwdchecks	NA	NA	NA
*A maximum of 50 passwords are retained. NA = not applicable			

Set restrictions so that passwords are hard to guess, yet not hard to remember. Passwords that are hard to remember are often written down somewhere, which compromises system security.

If text processing is installed on the system, the administrator can use the **/usr/share/dict/words** file as a **dictionlist** dictionary file. In such a case, the administrator can set **minother** to 0. Because most words in this dictionary file do not contain characters that fall into the **minother** category, setting **minother** to 1 or more eliminates the need for the vast majority of words in this dictionary file.

Extending Password Restrictions

The rules used by the password program to accept or reject passwords (the password composition restrictions) can be extended by system administrators to provide site-specific restrictions. Restrictions are extended by adding subroutines, known as methods, which are called during a password change. The **pwdchecks** attribute in the **/etc/security/user** file specifies the methods called.

The *AIX 5L Version 5.1 Technical Reference* contains a description of the **pwdrestrict_method**, the subroutine interface to which specified password restriction methods must conform. To correctly extend the password composition restrictions, the system administrator must program this interface when writing a password restriction method. Use caution in extending the password composition restrictions. These extensions directly affect the **login** command, the **passwd** command, the **su** command, and other programs. The security of the system could easily be subverted by malicious or defective code. Only use code that you trust.

Login User IDs

All audit events recorded for this user are labeled with this ID and can be examined when you generate audit records. See the *AIX 5L Version 5.1 System User's Guide: Operating System and Devices* for more information about login user IDs.

System Security Guidelines

The following guidelines are for system administrators who need to implement and maintain system security.

> **Attention:** Some operating environmente might have unique security requirements that are not addressed in these guidelines. To ensure a secure system, system administrators might need to implement additional security measures not discussed here.

This information does not provide security guidelines for all operational environments. It is impossible to create a single set of guidelines for all security requirements. These guidelines are not intended to represent the only requirements for achieving a secure system.

It is helpful to plan and implement your security policies before you begin using the system. Security policies are very time-consuming to change later, so a little planning now can save a lot of time later.

The security guidelines by category are:
- “Basic Security”
  - “User Accounts”
  - “Groups” on page 16
  - “File System Security” on page 17
  - “Root Access” on page 17
  - “PATH Environment Variable” on page 18
- “Advanced Security” on page 19
  - “Accounting” on page 19
  - “Auditing” on page 19
  - “Trusted Computing Base (TCB)” on page 19
- “Networks and Communications Security” on page 19.

Basic Security

Every system should achieve and maintain the level of security represented by these basic security policies.

User Accounts

Many attributes can be set for each user account, including password and login attributes. (For a list of configurable attributes, see “Chapter 5. Disk Quota System” on page 45 in *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*. The following are recommended:
- Each user should have a user ID that is not shared with any other user. All of the security safeguards and accountability tools only work if each user has a unique ID.
- Give user names that are meaningful to the users on the system. Actual names are best, since most electronic mail systems use the user ID to label incoming mail.
- Add, change, and delete users using the Web-based System Manager or SMIT interface. Although you can perform all these tasks from the command line, these interfaces help reduce small errors.
- Do not give a user account an initial password until the user is ready to log in to the system. If the password field is defined as an * (asterisk) in the **/etc/passwd** file, account information is kept, but no one can log in to that account.
- Do not change the system-defined user IDs that are needed by the system to function correctly. The system-defined user IDs are listed in the **/etc/passwd** file.
- In general, do not set the **admin** parameter to **true** for any user IDs. Only the root user can change attributes for users with **admin=true** set in the `/etc/security/user` file.

File Ownership and User Groups

When a file is created, the operating system assigns the user ID of the new file the effective user ID of the process that created it. The group ID of the file is either the effective group ID of the process or the group ID of the directory that contains the file, based on the set group ID (SUID) bit of that directory.

File ownership can be changed with the **chown** command.

The **id** command shows your user ID (UID), group ID (GID), and the names of all groups you belong to.

In file listings (such as the listings shown by the **ls** command), the three groups of users are always represented in the following order: user, group, and others. If you need to find out your group name, the **groups** command shows all the groups for a user ID.

The File Ownership and User Groups in *AIX 5L Version 5.1 System User's Guide: Operating System and Devices* contains more information about file and directory access modes.

Groups

Groups are collections of users who can share access permissions for protected resources. Plan your system groups before you begin creating them. Groups can make administration easier, but once you start using the system, it is harder to change your group organization. There are three types of groups:

1. User
2. System administrator
3. System-defined.

User Groups: In general, create as few user groups as possible.

Groups are made for people who need to share files on the system, such as people who work in the same department, or people who are working on the same project.

For example, consider a small engineering office with three sets of people in the office: office support personnel, system administrators, and engineers. Only two user groups, one for each function in the office, are needed: OFFICE (for the office management staff), and ENGINEER (for the engineers). Later, for example, if a small group of engineers begins work on a special project, a new group called PROJECT can be created and those engineer user IDs can be added to the PROJECT group. Though users can be in more than one group at a time, as in this case, they can only have one primary group at a time. Users can change their primary group with the **newgrp** command.

It is also recommended for simple systems that you do not set the **admin** characteristic when creating groups. If a group has `admin=true` set in the **/etc/security/group** file, only the root user can administer that group.

System Administrator Groups: System administrators are members of the SYSTEM group. SYSTEM group membership allows an administrator to perform some system maintenance tasks without having to operate with root authority.

System-Defined Groups: Several system-defined groups exist in AIX. The STAFF group is the default group for all nonadministrative users created in the system. You can change the default group by using the **chsec** command to edit the **/usr/lib/security/mkuser.default** file.

The SECURITY group is a system-defined group having limited privileges for performing security administration. SECURITY group members have access to programs and files in **/etc/security** directory. SECURITY group members can change most attributes for nonadministrative users and groups, such as the user's login shell or the membership of a nonadministrative group.

Most systems do not need to use this group; only multiuser systems with many users should consider using this group. Otherwise, system administrators can perform the same tasks as SECURITY group members by using the **su** command to gain root privilege.

The other system-defined groups are used to control certain subsystems. Consult the subsystem information to see if certain users can be defined as a member of those groups. The system-defined groups and users appear in the **/etc/group** file.

File System Security

All file system objects (including files, directories, special files, link files, symbolic link files, and pipes) have security mechanisms associated with them. The most commonly used is the access control list (ACL), but the following additional ways of controlling file security can also be used:

Base ACLs	Specifies the permissions for the owner, group, and others. These permissions are controlled through the **chmod** command. For more information, see the section on File Directory and Access Modes in the *AIX 5L Version 5.1 System User's Guide: Operating System and Devices*.
Extended ACLs	Provides finer access control than the base ACLs. For more information about the extended ACLs, see the section on Access Control Lists in the *AIX 5L Version 5.1 System User's Guide: Operating System and Devices*.
Status of Extended ACLs	The extended ACL must be enabled for a file system object; otherwise, the extended ACL is ignored.
Owner ID	This is the user ID of the owner of the file system object. Only this user ID has the permissions granted for the owner of the object.
Group ID	This is the ID of the group associated with the object. Only members of this group have the permissions granted for the group associated with the object.
Sticky bit	If the sticky bit is set for a directory, only the owner of the directory or the owner of a file can delete or rename a file within that directory, even if others have write permission to the directory. This can be set with the **t** flag with the **chmod** command.
TCB bit	If the TCB bit is set for a file system object, it identifies that object as part of the Trusted Computing Base (TCB).
umask	The **umask** environment parameter specifies the default permissions for any file or directory created.
Status of the file system	A file system can be mounted with read/write or read-only permissions.

Follow these guidelines when dealing with file system objects:

- In general, do not use extended ACLs. For most systems, base ACLs are sufficient for administration. If you do need the extra security control provided by extended ACLs, use them only when necessary and in an organized manner. Maintaining relevant entries in many extended ACLs can become very time-consuming. Also, do not use extended ACLs at all if you are in a heterogeneous network because they are only recognized by this operating system.
- Use the sticky bit on directories where everyone has write permissions.
- Protect user **.profile** files with 740 permissions.
- Do not let users have write access to system directories.
- Do not change permissions on any files or directories installed as part of the system. Changing those permissions affects system integrity.

Root Access

Attention: The root account should always have a password, and that password should never be shared. Only the system administrator should know the root password. System administrators should only operate as root to perform system administration functions that require root privileges. For all other operations, they shoult return to their normal user account. Routinely operating as root can result in damage to the system as root overrides many safeguards in the system.

The system administrator must have the root password to gain root authority by using the **su** command. The root account should be given a password immediately after the system is installed.

The root account is always authenticated by means of the local security files.

PATH Environment Variable

The **PATH** environment variable is an important security control. It specifies the directories to be searched to find a command. The default systemwide **PATH** value is specified in the **/etc/profile** file, and each user normally has a **PATH** value in the user's **$HOME/.profile** file. The **PATH** value in the **.profile** file either overrides the systemwide **PATH** value or adds extra directories to it.

Unauthorized changes to the **PATH** environment variable can enable a user on the system to ″spoof″ other users (including root users). Spoofing programs (also called Trojan Horse programs) replace system commands and then capture information meant for that command, such as user passwords.

For example, suppose a user changes the **PATH** value so that the system searches the **/tmp** directory first when a command is run. Then the user places in the **/tmp** directory a program called **su** that asks for the root password just like the **su** command. Then the **/tmp/su** program mails the root password to the user and calls the real **su** command before exiting. In this scenario, any root user who used the **su** command would give away the root password and not even be aware of it. This is just one of many scenarios for gaining confidential information by altering **PATH** values.

However, following a few simple steps will prevent any problems with the **PATH** environment variable for system administrators and users:

- When in doubt, specify full path names. If a full path name is specified, the **PATH** environment variable is ignored.
- Never put the current directory (specified by **.** (period)) in the **PATH** value specified for the root user. Never allow the current directory to be specified in **/etc/profile**.
- The **PATH** value in the **/etc/profile** file is used by the root user. Only specify directories that are secure and can only be written to by the root user. Do not create a **.profile** file in the **/** (root) directory. The **.profile** files should only be in user **$HOME** directories.
- Warn other users not to change their **.profile** files without consulting the system administrator. Otherwise, an unsuspecting user could make changes that allow unintended access. A user **.profile** file should have permissions set to 740.
- System administrators should not use the **su** command to gain root privilege from a user session, because the user's **PATH** value specified in the **.profile** file is in effect. User's can set their **.profile** files to whatever they please. System administrators should log on to the user's machine as root, or should use the following command:

  ```
  su - root
  ```

  This ensures that the root environment is used during the session. If a system administrator does operate as root in another user session, then the system administrator should specify full path names throughout the session.
- Protect the input field separator (**IFS**) environment variable from being changed in the **/etc/profile** file. And beware of any user who changes the **IFS** variable in the **.profile** file. It too can be used to alter the **PATH** value.

Advanced Security

These security policies provide a greater level of security, but also require more work to maintain. Consequently, many system administrators use these security features only in a limited way, or not at all.

Accounting

System accounting is not a direct security function, but the information it gathers is important for detecting security problems. You should activate basic accounting on your system, as explained in Setting Up an Accounting System, although you may want to consider not activating disk accounting and printing accounting as specified in the procedure. Both of these accounting functions produce a large amount of data, and they are not vital to system security.

Auditing

For smaller systems, auditing is generally not necessary. However, on large multiuser systems, it can provide useful information about system activity. For more information, see “Auditing Overview” on page 30.

Trusted Computing Base (TCB)

The Trusted Computing Base (TCB) allows administrators to closely monitor trusted programs and enhance the security of the system.

Most systems use only two parts of the TCB: the **tcbck** command and the default **/etc/security/sysck.cfg** configuration file. The **tcbck** command uses information in the **/etc/security/sysck.cfg** file to compare the security status of key elements of the system against the base data stored in the **sysck.cfg** file. All the administrator must do is protect the **sysck.cfg** file and run the **tcbck** command regularly.

For larger systems, the TCB can monitor the system more extensively and provide a secure set of system components. But this extra security requires extra administrative effort. For more information, see “Trusted Computing Base Overview”.

Administrative Roles

AIX 4.2.1 and later supports the system administrator assigning to a user a set of administrative roles with various levels of authority. For more information, see “Roles Overview” on page 39.

Networks and Communications Security

Networks and communications security are described fully in *AIX 5L Version 5.1 System Management Guide: Communications and Networks*.

- TCP/IP Security
- TCP/IP Command Security
- Understanding BNU Security
- Data Security and Information Protection
- NIS Maintenance
- Configuring Secure NFS.

Trusted Computing Base Overview

The Trusted Computing Base (TCB) is the part of the system that is responsible for enforcing the information security policies of the system. All of the computer hardware is included in the TCB, but a person administering the system should be concerned primarily with the software components of the TCB.

Many of the TCB functions are now optionally enabled at installation time. Selecting **yes** for the **Install Trusted Computing Base** option on the Installation and Settings menu enables the trusted path, trusted shell, and system integrity checking (**tcbck** command). Selecting **no** disables these features. These features can only be enabled at installation time.

The TCB software consists of:

- The kernel (operating system)
- The configuration files that control system operation
- Any program that is run with the privilege or access rights to alter the kernel or the configuration files.

Most system files are accessible only by the root user; however, some can also be accessed by members of an administrative group. Only the root user can alter the operating system kernel. The TCB contains the following trusted programs:

- All **setuid** root programs
- All **setgid** programs to administrative groups
- Any program that is exclusively run by the root user or by a member of the system group
- Any program that must be run by the administrator while on the trusted communication path (for example, the **ls** command).

In the operating system, the person who administers the system can mark trusted files as part of the Trusted Computing Base (the **chtcb** command), so that they can be clearly distinguished.

The person who administers the system must be careful to add only software that can be fully trusted to the TCB. Consider trusting software if, for example:

- You have fully tested the program.
- You have examined the program's code.
- The program is from a trusted source that has tested or examined the program.

The system administrator must determine how much trust can be given to a particular program. This determination includes considering the value of the information resources on the system in deciding how much trust is required for a program to be installed with privilege.

tcbck Checking Programs

An important aspect of the **tcbck** program is the **program** attribute, located in the **/etc/security/sysck.cfg** file. The **program** attribute lists an associated program that can check additional status. This attribute allows for more thorough and flexible checking than other attributes provide.

You can use these checking programs to check the integrity and consistency of the contents of a file and its relationship with other files. Checking programs need not be bound to a particular file.

For example, assume you wrote a program, **/etc/profile**, which verifies that each user **.profile** file is writable only by that user. The program should have the following aspects:

- Owned by **root**
- Member of **system** group
- Has a mode of 0750
- Tagged as part of the TCB.

You can add your program, **/etc/profile**, to the system security checker by entering the following:

```
tcbck -a /etc/profile
"program=/etc/profile" class=profiles \ owner group mode
```

This command creates the following entry in the database:

```
/etc/profile:

class = profiles

owner = root
```

```
group = system

mode = TCB,rwxr-x---

program = "/etc/profile"
```

The following **tcbck** command verifies the installation of the **/etc/profile** program and runs the program:

```
tcbck -t profiles
```

There are several requirements for **tcbck** checking programs:

- **tcbck** must accept the **-n**, **-y**, **-p**, and **-t** flags and handle these similarly to the **tcbck** command.
- **tcbck** must return 0 to indicate that no errors were found and write all error messages to standard error.
- It is important to note that these programs are run with an effective user ID of 0; therefore, they are fully privileged. They should be written and inspected as **setuid-root** programs.

TCB Checking Programs

The operating system supplies the following TCB checking programs:

pwdck	Checks the **/etc/passwd** and **/etc/security/passwd** files for internal and mutual consistency.
grpck	Checks the **/etc/group** and **/etc/security/group** files for internal and mutual consistency.
usrck	Verifies the accuracy of the user definitions in the user database files by checking the definitions for all the users or for specific users.

Secure System Installation and Update

Installing or updating a program consists of importing files into the system, usually creating new directories for the program, and occasionally reconfiguring the system itself. From a security standpoint, the program might need to add user accounts, define new audit events, and assign privileges to one of the program files.

In the simplest mode, a program installation consists of installing a new subdirectory tree (from **/usr/lpp**) and possibly adding new symbolic links in the **/usr/bin** directory. However, two problems exist:

- Most real programs need to alter the system configuration and contain commands that need to be installed with some degree of administrative privilege.
- Each installation should be done under a separate access domain so that the installation procedure of one program cannot interfere with that of another.

To provide for secure program installation and update, two strategies are employed. First, privilege and access rights are delineated and limited during installation and update. This minimizes the potential for damage by untrustworthy installation packages. Second, the entire process is auditable. The auditing can be done by examining the system audit trail after installation or update of the program is complete, or auditing can be interactive. The **watch** command is provided for interactive use. It can be used to run a specified program and display the stream of audit records (if any) that are generated during the running of that program.

This approach provides a great deal of flexibility in installation, while still providing a high degree of security. Even though the security is detection rather than prevention, this is still effective. Because the process is interactive, the installation of malicious programs can be halted quickly.

Guidelines for Ownership and Modes for Files

Normal User Commands: Normal user commands do not need a real owner or group since they can be run by anyone and are not set user ID (SUID) or set group ID (SGID). Set the TCB bit if the command is expected to be run on the trusted path (for example, using the **vi**, **grep**, and **cat** commands). The following is an example of ownership and modes:

```
owner:  bin     r-x
group:  bin     r-x
others:         r-x
```

Administrative User Commands: Administrative user commands are run only by the root user, members of an administrative group, and members who are specified in the extended access control list (ACL) entries. Because they usually perform privileged operations, some of these commands might need to be run with privilege (set user ID, or SUID). The following is an example of ownership and modes:

```
owner: root   r-x (possibly SUID)
group: system r-x (possibly SGID)
others:(possibly extended Access Control List entries)
```

For an example of a typical administrative user command scenario, consider network files containing critical network configuration information:

```
owner:  root            rw-
group:  netgroup        rw-
others:                 ---
```

Use the **changenet** command to modify the information in the network file.

```
owner: root       r-x
group: netgroup   —s
others:(extended Access Control List entry)
permit r-x g:net
permit r-x u:john
```

In this example, group netgroup is an administrative privilege group with no members. Only processes running with group netgroup (or root user) can read or write the network files. Running the **changenet** command, which is the set group ID (SGID) netgroup, gives users the ability to change the network files. Only root users, members of group `net`, and user `john` can execute this command.

The following figure shows how the extended ACL is associated with the command instead of with the data files themselves.

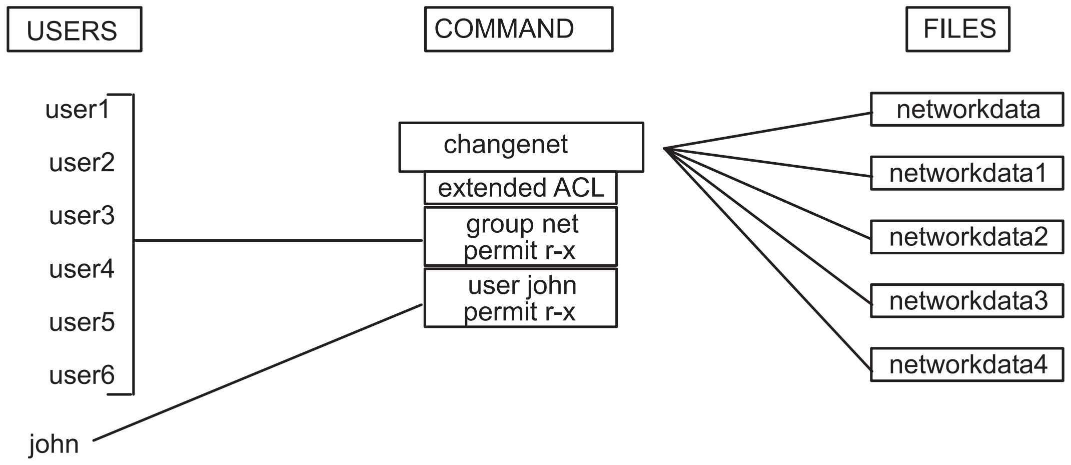

Figure 1. Extended ACL Controls. . This illustration shows users 1-6 grouped with the extended ACLs **group net permit r-x***. Another user, John, is designated with the extended ACLs* **user john permit r-x***. Through the* **changenet** *command, these users are provided access to network data files depending on their extended ACLs.*

The **changenet** command is the gateway to the network configuration files. The extended Access Control List on the **changenet** command is the guard that allows only certain users through.

This example assumes that there are more data files than programs that operate on them. If the reverse were true, it might make more sense to have ACLs on the data files.

Configuration Files

The following example shows the admin group as an administrative privilege group with no members. The exact name of the admin group depends on what type of configuration file the group applies to (for example, auditing, authentication, and mail).

```
owner:  root    rw-
group:  admin   rw-
others:         r--
mode:           TCB
```

In general, most configuration files are readable by all users, with the exception of auditing and authentication data.

Device Special Files

In general, devices should not be readable or writable by normal users. Exceptions to this rule are terminal devices that are writable so users can send messages to each other, and floppy disks that are both readable and writable so users can transfer files.

Trusted Communication Path

The operating system trusted communication path allows for secure communication between users and the Trusted Computing Base. Users start the trusted communication path by pressing the secure attention key (SAK). This allows only trusted processes to access the user's terminal. A trusted communication path is used whenever the user must enter data that must not be compromised (a password, for example). A trusted communication path can also be used by the person who administers the system to establish a secure environment for administration.

Note: If the **Install Trusted Computing Base** option was not selected during the initial installation, the trusted communication path is disabled. The path can be enabled only by reinstalling the system.

The trusted communication path is based on:

- A trusted command interpreter (**tsh** command) that only runs commands that are marked as being a member of the Trusted Computing Base
- Restricting access to a terminal to trusted programs
- A reserved key sequence, called the SAK, that allows the user to request a trusted communication path.

After the SAK has been pressed, the **init** command starts either the **getty** command or the **shell** command that:

- Changes the owner and mode of the terminal so that only processes run by a user can open the terminal
- Issues a **frevoke** subroutine to invalidate all previous **open** calls of the terminal.

This ensures that only the current **getty** or **shell** command has access to the terminal.

Trusted Command Interpreter

The **getty** command runs the **shell** command in response to a SAK only if the user was logged in to this terminal. This command establishes the terminal modes and runs the trusted shell (**tsh** command).

The **tsh** command provides a subset of the functions of the normal shell (Korn shell). The trusted shell runs only trusted programs (for example, programs tagged with the TCB bit). The built-in **shell** command allows the user to drop the trusted communication path and run the user login shell.

Pluggable Authentication Module (PAM) Overview

The Pluggable Authentication Module (PAM) framework allows for new authentication technologies to be *plugged-in* without changing your application commands such as *web_auth*, *file_copy*, or *screen_save*. It can be used to integrate an AIX login with other security mechanisms sich as DCE or Kerberos. Mechanisms for account, session, and password management can also be plugged in using this framework.

PAM allows the system administrator to choose any combination of services to provide authentication. The list below includes some of the advantages of PAM to the system administrator.

- Flexible configuration policy
  - Authentication can be selected on a per-application policy.
  - A default-authentication mechanism can be set up for non-specified applications.
  - High-security systems can require multiple passwords.
- Ease of use
  - Identical passwords do not require reytping.
  - Password mapping lets users type a single password, even if the passwords associated with separate authentication methods are different.
- Can pass optional parameters to the services
- All PAM interfaces supported:
  - pam_sm_authenticate
  - pam_sm_setcred
  - pam_sm_acct_mgmt
  - pam_sm_open_session
  - pam_sm_close_session
  - pam_sm_chauthtok

PAM Terminology

PAM employs run-time pluggable modules that provide authentication-related services. These modules are broken into four different categories: authentication, account management, session management, and password management.

Authentication Modules
: Authenticate users and set, refresh, or destroy credentials. These modules identify the user based on his or her authentication and credentials.

Account Modules
: Check for password aging, account expiration, and access hour restrictions. After the user is identified using the authentication modules, account modules determine if the user is given access based on his or her account restrictions.

Session Modules
: Manage the opening and closing of an authentication session. Optionally, they can log activity or provide for clean-up after the session is over.

Password Modules
: Manage changes to the password and the password-related attributes.

PAM uses a concept called *stacking* to fully identify users through multiple authentication methods. Depending on the configuration, a user can be prompted for one or more passwords for each authentication method. The user does not need to remember to execute another command to get fully authenticated. The order in which methods are used is set in the**/etc/pam.conf** configuration file.

Stacking can require that a user must remember several passwords. However, if password mapping is supported by the underlying module, a primary password can be used to decrypt the other passwords, so the user only needs to remember the primary password. Alternatively, an administrator can synchronize the password across each authentication mechanism, but this method might increase the security risk because the security of each mechanism would be limited by the least secure password method used in the stack.

Administering a PAM Framework

The following figure illustrates the relationship between the applications, the PAM library, and the PAM modules. The applications (web_auth, file_copy, and screen_save) use the PAM library to access the appropriate module. The **pam.conf** file defines which modules are used with each application. Responses from the modules are passed back through the library to the application.

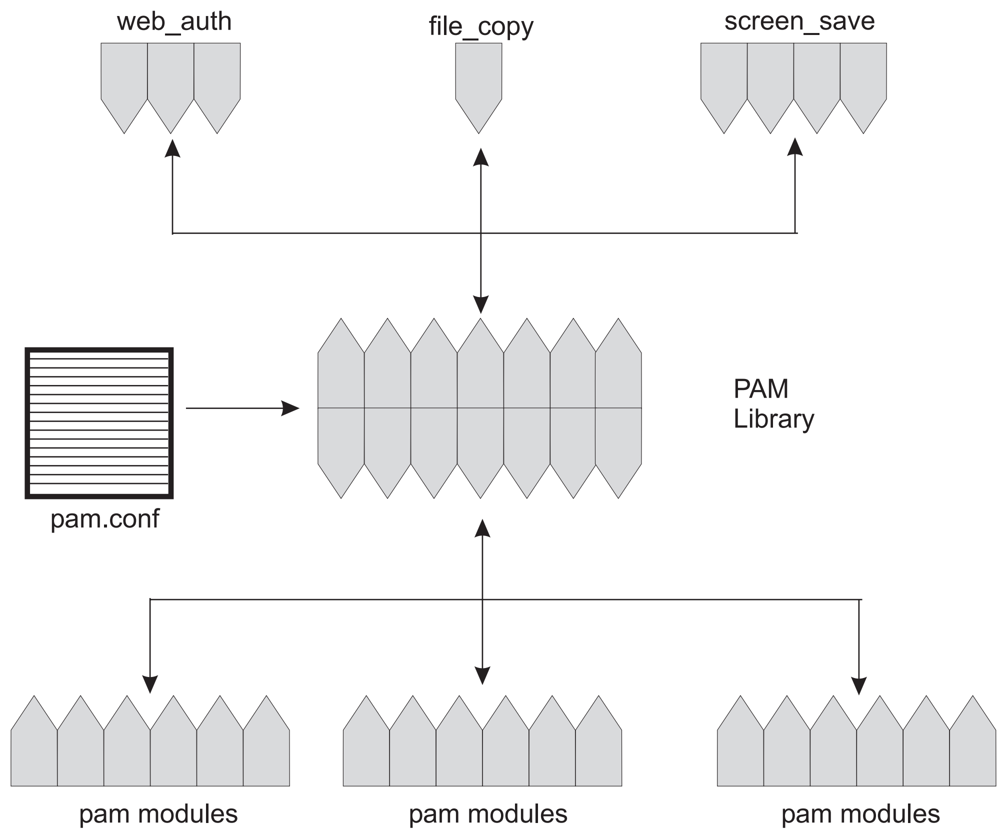

Figure 2. PAM Framework and Entities. This illustration shows ficticious application commands using the PAM library to access the appropriate PAM module.

The PAM software consists of a library, several modules and a configuration file. The current release also includes new versions of several commands which use the PAM interfaces.

PAM Library

The PAM library, **/usr/lib/libpam**, provides the framework to load the appropriate modules and manage stacking. It provides a generic structure for all of the modules to plug into.

PAM Modules

Each PAM module provides the implementation of a specific mechanism. More than one module type (authentication, account, session, or password) can be associated with each module, but each module needs to manage at least one module type. Here is a description of some of the modules.

Because the pam_unix module, **/usr/lib/security/pam_unix.so.1**, provides support for authentication, account management, session management, and password management, any module type definition can be used with this module. The pam_unix module uses AIX passwords for authentication.

The dial_auth module, **/usr/lib/security/pam_dial_auth.so.1**, can only be used for authentication. It uses data stored in the **/etc/dialups** and **/etc/d_passwd** files for authentication. This module is mainly used by the **login** command.

The rhosts_auth module, **/usr/lib/security/pam_rhosts_auth.so.1**, can also only be used for authentication. It uses data stored in the **˜/.rhosts** and **/etc/host.equiv** files through **ruserok** subroutine. Thismodule is mainly used by the **rlogin** and **rsh** command.

The pam_dce module, **/usr/lib/security/pam_dce.so.1**, provides support for authentication, account management, and password management. Any of these module type definitions can be used with this module. The module uses DCE registry for authentication.

For security reasons, these files must be owned by the root user and the permissions must be set so `group` or `other` cannot write to the files. If files are not owned by the root user, then PAM does not load the module.

PAM Configuration File

The **/etc/pam.conf** configuration file can be edited to select authentication mechanisms for each system-entry application. The file consists of entries that use the following syntax:

```
service_name module_type control_flag module_path module_options
```

where:

service_name
Indicates the name of the service

module_type
Denotes the module type for the service

control_flag
Selects the continuation and failure semantics for the module

module_path
Specifies the pathname to a library object which implements the service functionality

module_options
List specific options that can be passed to the service modules

The only optional component is *module_options*. The other values must be defined. Comments can be added to the file by starting a line with a pount sign (#). Whitespace delimits the fields.

An entry in this file is ignored if one of the following conditions exist:

- The line has less than four fields.
- An invalid value is given for *module_type* or *control_flag*.
- The named module is not found.

One of three *control_flags* values must be selected for each entry to determine continuation or failure behavior from the module. These flags determine what the ultimate result (success or failure) will be. The values are defined below:

required
This module must return success for the overall result be successful.

optional
If this module fails, the overall result can be successful if another module in this stack returns success.

sufficient
If this module is successful, skip the remaining modules in the stack, even if any are flagged as **required**.

If all of the modules are flagged as **required**, then authentication through all modules must succeed for the user to be authenticated. If some of the modules fail, then a error value from the first failed module is reported. If a failure occurs for a required module, all modules in the stack are still tried but the access is denied.

If none of the modules are flagged as required, then at least one of the entries for that service must succeed for the user to be authenticated. The **optional** flag should be used when one success in the stack is enough. This flag should *only* be used when it is not important for this mechanism to succeed. For instance, if your users must have permission associated with a specific mechanism to get their work done, then that entry should not be flagged as **optional**.

The **sufficient** flag allows for one successful authentication to be enough for the user to get in.

Configuring PAM

The section below discusses some of the tasks that might be required for full PAM functionality. In particular, you should be note the security issues associated with each potential change to the configuration file.

When configuring PAM for your site, use the following general procedure:

1. Determine what your authentication needs are and which modules fulfill your requirements.
2. Identify which services need special attention.
3. Decide on the sequence in which the modules should be run.
4. Select a control flag for each module.
5. Choose options, if necessary, for each module

Consider the following before changing the configuration file:

- Use the OTHER entry for every module type to avoid having to explicitly include each application.
- Ensure you understand the security implications of the **sufficient** and **optional** control flags before you use them.
- Review the documentation associated with the modules to understand how each one functions and what options each allows.
- Review the documentation to understand the interactions between stacked modules.

Disabling the ˜/.rhosts Entries

To disable reading the **˜/.rhosts** files during an rlogin session, remove the **rlogin auth rhosts_auth.so.1** entry from the configuration file. This action prevents unauthenticated access to the local system from remote systems and requires a password for any rlogin access, regardless of the presence or contents of any **˜/.rhosts** or **/etc/hosts.equiv files**.

To prevent other non-authenticated access using the **˜/.rhosts** files remember to disable the **rsh** service. The best way to disable a particular service is to remove the service entry from the **/etc/inetd.conf** file. Making changes to the PAM configuration file does not prevent the service from starting.

Adding the DCE PAM Module

To add the DEC PAM module, edit the **/etc/pam.conf** file to look like the following:

```
# # PAM configuration # # Authentication management # login auth sufficient
/usr/lib/security/pam_dce.so.1 login auth required /usr/lib/security/pam_unix.so.1
rlogin auth required /usr/lib/security/pam_unix.so.1 rsh auth required
   /usr/lib/security/pam_rhost_auth.so.1
OTHER auth required /usr/lib/security/pam_unix.so.1 # # Account management
# login account required /usr/lib/security/pam_dce.so.1 login account required
/usr/lib/security/pam_unix.so.1 OTHER account required
   /usr/lib/security/pam_unix.so.1
# # Session management # OTHER session required /usr/lib/security/pam_unix.so.1
# # Password management # passwd password required /usr/lib/security/pam_dce.so.1
passwd password required /usr/lib/security/pam_unix.so.1
```

The **sufficient** flag on the login entry indicates that a user is authenticated if the pam_dce module suceeds, regardless of other modules, such as the pam_unix module. In this case, the pam_unix module is checked only when authentication through the pam_dce module fails.

The two entries for **login** authentication allow the root user to access the local system. This extra line is necessary because DCE does not grant root access by default. Because of these lines, non-root users without AIX password cannot get in, but a locally defined root account is able gain access.

The above example illustrates the DCE module's use only with the **login** command. However, the DCE module can be added for other services. If the pam_dce module is added as an auth module for login, it must also be added as an account module as well. The DCE entry for the passwd service ensures that the DCE password is also changed when the user runs the **password** command.

Changing /etc/pam.conf

Note: If the PAM configuration file is misconfigured or becomes corrupted, it is possible that even the root user could not be able to login. The root user could use the **su** command, since it does not use PAM, then reboot the machine as a single-user machine to fix the **/etc/pam.conf** configuration file.

You must be logged in as the root user to edit the **/etc/pam.conf** file. Whenever you change the **/etc/pam.conf** file, and while you are still logged in as root, test each command that might be affected by your changes to ensure the configuration file remains valid. For example, if you add a new module to the **telnet** service, use the **telnet** command to test your changes and verify that the **/etc/pam.conf** file performs as expected.

Adding a Module

Use the following procedure to add a PAM module:

1. Study the documentation on the module and determine which control flags and other options should be used.
2. Copy the new module to **/usr/lib/security**.
3. Set the permissions so the module file is owned by `root` and permissions are `555`.
4. Edit the PAM configuration file, **/etc/pam.conf**, to add this module to the appropriate services.
5. Test your changes.

Initiating Error Reporting

Add entries to **/etc/syslog.conf** to initiate error reporting. The **syslogd** daemon must be restarted or a SIGHUP signal sent to it for any changes to take effect. The following selections can be added to the file to gather information about PAM:

auth.alert
Messages about conditions that should be fixed now

auth.crit
Critical messages

auth.err
Error messages

auth.info
Informational messages

auth.debug
Debugging messages

The following entry prints all alert messages to the console. Critical messages are mailed to the root user. Informational and debug messages are added to **/var/log/pamlog**.

```
auth.alert /dev/console
auth.crit 'root'
auth.info;auth.debug /var/log/pamlog
```

Each line in the log contains a time stamp, the name of the system that generated it, and a message. The **/var/log/pamlog** file can log a large amount of information.

Auditing Overview

The auditing subsystem provides the system administrator with the means to record security-relevant information, which can be analyzed to detect potential and actual violations of the system security policy. The auditing subsystem has three functions:

- Event detection
- Information collection
- Information processing.

Each of these functions can be configured by the system administrator.

Event Detection

Event detection is distributed throughout the Trusted Computing Base (TCB), both in the kernel (supervisor state code) and the trusted programs (user state code). An auditable event is any security-relevant occurrence in the system. A security-relevant occurrence is any change to the security state of the system, any attempted or actual violation of the system access control or accountability security policies, or both. The programs and kernel modules that detect auditable events are responsible for reporting these events to the system audit logger, that runs as part of the kernel and can be accessed either with a subroutine (for trusted program auditing) or within a kernel procedure call (for supervisor state auditing). The information reported includes the name of the auditable event, the success or failure of the event, and any additional event-specific information that is relevant to security auditing.

Event detection configuration consists of turning event detection on or off, either at the global (system) level or at the local (process) level. To control event detection at the global level, use the **audit** command to enable or disable the audit subsystem. To control event detection at the local level, you can audit selected users for groups of audit events (audit classes).

Information Collection

Information collection encompasses logging the selected auditable events. This function is performed by the kernel audit logger, which provides both an SVC (subroutine) and an intra-kernel procedure call interface that records auditable events.

The audit logger is responsible for constructing the complete audit record, consisting of the audit header, that contains information common to all events (such as the name of the event, the user responsible, the time and return status of the event), and the audit trail, which contains event-specific information. The audit logger appends each successive record to the kernel audit trail, which can be written in either (or both) of two modes:

BIN mode	The trail is written into alternating files, providing for safety and long-term storage.
STREAM mode	The trail is written to a circular buffer that is read synchronously through an audit pseudo-device. STREAM mode offers immediate response.

Information collection can be configured at both the front end (event recording) and at the back end (kernel trail processing). Event recording is selectable on a per-user basis. Each user has a defined set of audit events that are actually logged in the kernel trail when they occur. At the back end, the modes are individually configurable, so that the administrator can employ the back-end processing best suited for a particular environment. In addition, BIN mode auditing can be configured to shut down the system in the event of failure.

Information Processing

The operating system provides several options for processing the kernel audit trail. The BIN mode trail can be compressed, filtered or formatted for output, or any reasonable combination of these before archival storage of the audit trail, if any. Compression is done through Huffman encoding. Filtering is done with standard query language (SQL)-like audit record selection (using the **auditselect** command) and provides for both selective viewing and selective retention of the audit trail. Formatting of audit trail records can be used to examine the audit trail, to generate periodic security reports, and to print a paper audit trail. The STREAM mode audit trail can be monitored in real time, to provide immediate threat monitoring capability. Configuration of these options is handled by separate programs that can be invoked as daemon processes to filter either BIN or STREAM mode trails, although some of the filter programs are more naturally suited to one mode or the other.

Event Selection

The set of auditable events on the system defines which occurrences can actually be audited and the granularity of the auditing provided. The auditable events must cover the security-relevant events on the system, as defined previously. The level of detail you use for auditable event definition must tread a fine line between insufficient detail, leading to excessive information collection, and too much detail, making it difficult for the administrator to logically understand the selected information. The definition of events takes advantage of similarities in detected events. For the purpose of this discussion, a detected event is any single instance of an auditable event; for instance, a given event might be detected in various places. The underlying principle is that detected events with similar security properties are selected as the same auditable event. The following list shows an event classification:

```
Security Policy Events
   Subject Events
      - process creation
      - process deletion
      - setting subject security attributes: user IDs, group IDs,
      - process group, control terminal
   Object Events
      - object creation
      - object deletion
      - object open (including processes as objects)
      - object close (including processes as objects)
      - setting object security attributes: owner, group, ACL
   Import/Export Events
      - importing or exporting an object
   Accountability Events
      - adding a user, changing user attributes in the password
        database
      - adding a group, changing group attributes in the group
        database
      - user login
      - user logoff
      - changing user authentication information
      - trusted path terminal configuration
      - authentication configuration
      - auditing administration: selecting events and audit trails,
        switching on/off, defining user auditing classes
   General System Administration Events
      - use of privilege
      - file system configuration
      - device definition and configuration
      - system configuration parameter definition
      - normal system IPL and shutdown
      - RAS configuration
      - other system configuration
   Security Violations (potential)
```

```
- access permission refusals
- privilege failures
- diagnostically detected faults and system errors
- (attempted) alteration of the TCB.
```

Configuration

The auditing subsystem has a global state variable that indicates whether the auditing subsystem is on or off. In addition, each process has a local state variable that indicates whether the auditing subsystem should record information about this process. Both of these variables determine whether events are detected by the Trusted Computing Base (TCB) modules and programs. Turning TCB auditing off for a specific process allows that process to do its own auditing and not to bypass the system accountability policy. Permitting a trusted program to audit itself allows for more efficient and effective collection of information.

Information Collection

Information collection addresses **event selection** and **kernel audit trail** modes. It is done by a kernel routine that provides interfaces to log information, used by the TCB components that detect auditable events, and configuration interfaces, used by the auditing subsystem to control the audit logger routine.

Audit Logging

Auditable events are logged with one of two interfaces, the user state and supervisor state. The user state part of the TCB uses the **auditlog** or **auditwrite** subroutine, while the supervisor state portion of the TCB uses a set of kernel procedure calls.

For each record, the audit event logger prefixes an audit header to the event-specific information. This header identifies the user and process for which this event is being audited, as well as the time of the event. The code that detects the event supplies the event type and return code or status and optionally, additional event-specific information (the event tail). Event-specific information consists of object names (for example, files refused access or tty used in failed login attempts), subroutine parameters, and other modified information.

Events are defined symbolically, rather than numerically. This lessens the chances of name collisions, without using an event registration scheme. Because subroutines are auditable and the extendable kernel definition has no fixed SVC numbers, it is difficult to record events by number. The number mapping would have to be revised and logged every time the kernel interface was extended or redefined.

Audit Record Format

The audit records consist of a common header, followed by audit trails peculiar to the audit event of the record. The structures for the headers are defined in the **/usr/include/sys/audit.h** file. The format of the information in the audit trails is peculiar to each base event and is shown in the **/etc/security/audit/events** file.

The information in the audit header is generally collected by the logging routine to ensure its accuracy, while the information in the audit trails is supplied by the code that detects the event. The audit logger has no knowledge of the structure or semantics of the audit trails. For example, when the **login** command detects a failed login it records the specific event with the terminal on which it occurred and writes the record into the audit trail using the **auditlog** subroutine. The audit logger kernel component records the subject-specific information (user IDs, process IDs, time) in a header and appends this to the other information. The caller supplies only the event name and result fields in the header.

Logger Configuration

The audit logger is responsible for constructing the complete audit record. You must select the audit events that you want to be logged.

Event Selection

Audit event selection has two different types: per-process and per-object.

Per-Process Auditing	To select process events with reasonable efficiency and usability, the operating system allows the system administrator to define audit classes. An audit class is a subset of the base auditing events in the system. Auditing classes provide for convenient logical groupings of the base auditing events. For each user on the system, the system administrator defines a set of audit classes that determines the base events that could be recorded for that user. Each process run by the user is tagged with its audit classes.
Per-Object Auditing	The operating system provides for the auditing of object accesses by name, that is, the auditing of specific objects (normally files). Most objects are not that interesting from a security perspective. By-name object auditing prevents having to cover all object accesses to audit the few pertinent objects. In addition, the auditing mode can be specified, so that only accesses of the specified mode (read/write/execute) are recorded.

Kernel Audit Trail Modes

Kernel logging can be set to BIN or STREAM modes to define where the kernel audit trail is to be written. If the BIN mode is used, the kernel audit logger must be given (before audit startup) at least one file descriptor to which records are to be appended.

BIN mode consists of writing the audit records into alternating files. At auditing startup, the kernel is passed two file descriptors and an advisory maximum bin size. It suspends the calling process and starts writing audit records into the first file descriptor. When the size of the first bin reaches the maximum bin size, and if the second file descriptor is valid, it switches to the second bin and reactivates the calling process. It keeps writing into the second bin until it is called again with another valid file descriptor. If at that point the second bin is full, it switches back to the first bin, and the calling process returns immediately. Otherwise, the calling process is suspended, and the kernel continues writing records into the second bin until it is full. Processing continues this way until auditing is turned off.

The STREAM mode is much simpler than the BIN mode. The kernel writes records into a circular buffer. When the kernel reaches the end of the buffer, it simply wraps to the beginning. Processes read the information through a pseudo-device called **/dev/audit**. When a process opens this device, a new channel is created for that process. Optionally, the events to be read on the channel can be specified as a list of audit classes.

The main purpose of the STREAM mode is to allow for timely reading of the audit trail, which is desirable for real-time threat monitoring. Another use is to create a paper trail that is written immediately, preventing any possible tampering with the audit trail, as is possible if the trail is stored on some writable media.

PKCS #11 Overview

Note: The information in this section is specific to POWER-based.

The PKCS #11 subsystem provides applications a method for accessing hardware devices (tokens) in a device neutral manner. The content in this document conforms to Version 2.01 of the PKCS #11 standard.

This subsystem has been implemented using three components:

- A slot manager daemon (**pkcsslotd**) which provides the subsystem with information regarding the state of available hardware devices. This deamon is started automatically during installation and when the system is rebooted.
- An API shared object (**/usr/lib/pkcs11/pkcs11_API.so**) is provided as a generic interface to the adapters for which PKCS #11 support has been implemented.

- An adapter specific library which provides the PKCS #11 support for the adapter. This tiered design allows the user to easily use new PKCS #11 devices when they come available with no recompilations of existing applications.

For detailed information on implementing the PKCS #11 subsystem, see “PKCS #11 Overview” on page 33 in *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

LDAP Exploitation of the Security Subsystem

The Light Directory Access Protocol (LDAP) defines a standard method for accessing and updating information in a directory (a database) either locally or remotely in a client-server model. The LDAP method is exploited by a cluster of hosts to allow centralized security authentication as well as access to user and group information. This functionality is intended to be used in a clustering environment to keep authentication, user, and group information common across the cluster.

The LDAP exploitation of the security subsystem is implemented as the LDAP authentication load module. It is conceptually similar to the other load modules such as NIS, DCE, and Kerberos 5. The load modules are defined in the **/usr/lib/security/methods.cfg** file. The implementation of the LDAP authentication load module is at a low level and is handled by the libraries.

Once the LDAP authentication load module is enabled to serve user and group information, most high-level APIs, commands, and system management tools work in their usual manner. An **-R** flag is introduced for most high-level commands to work through different load modules. For example, to create an LDAP user named joe from a client machine, use the following command:

```
mkuser -R LDAP joe
```

The client system checks whether the user is an LDAP user through the user’s SYSTEM attribute in the **/etc/security/user** file. If the user’s SYSTEM is set to LDAP, that user can only authenticate through LDAP. If the SYSTEM attribute in the default stanza is set to LDAP, then all users who do not have a SYSTEM attribute set are considered LDAP users. The LDAP keyword can be used with other SYSTEM attribute values as described in “Security Administration” on page 11. The client side communicates to the server through the **secldapclntd** daemon. The daemon accepts requests from applications (through the library APIs), queries the LDAP server, and returns data to the application. The **secldapclntd** daemon is also responsible for caching.

Setting Up an LDAP Security Information Server

To set up a system as an LDAP security information server that serves authentication, user, and group information through LDAP, the LDAP server and client packages must be installed. The LDAP server must be configured as a client as well as a server. A DB2 database is also required by the LDAP server. If the Secure Socket Layer (SSL) is required, then the GSKit must be installed. The system administrator must create a key using the **ikeyman** command. The certificate of the server key must be carried to the clients.

The **mksecldap** command can be used to set up an LDAP security information server. It sets up a database named **ldapdb2**, populates the database with the user and group information from the local host, and sets the ldap server administrator DN (distinguished name) and password. Optionally, it can set up SSL for client/server communication. Then **mksecldap** loads a server plugin (**libsecldap.a**) and starts the LDAP server process (**slapd**). The **mksecldap** command also adds an entry into the **/etc/inittab** file to start the LDAP server at every reboot. The entire LDAP server setup is done through the **mksecldap** command, which updates the **slapd.conf** file (SecureWay Directory Version 3.1) or **slapd32.conf** file (SecureWay Directory Version 3.2). There is no need to configure the LDAP web management interface.

All the user and group information is stored under an AIX tree (suffix). The default suffix is "cn=aixsecdb". The **mksecldap** command accepts a user-supplied suffix through the **-d** flag. If the user-supplied suffix does not have "cn=aixsecdb" as its first RDN (Relative Distinguished Name), the **mksecldap** command prefixes the user-supplied suffix with "cn=aixsecdb". This AIX tree is ACL (Access Control List) protected. A client must bind as the LDAP server administrator to be able to access the AIX tree.

The **mksecldap** command works even if an LDAP server has been setup for other purposes, for example, for blue page information. In this case, **mksecldap** adds the AIX tree and populates it with the AIX security information to the existing database. This tree is ACL-protected independently from other trees. In this case, the LDAP server works as usual, in addition to it also serves as an AIX LDAP Security Server. It is strongly recommended that you backup your existing database before running the **mksecldap** command to setup the security server to share the same database.

After the LDAP security information server is successfully set up, the same host must be set up as a client.

If the LDAP security information server setup is not successful, you can undo the setup by running the **mksecldap** command with the **-U** flag. This restores the **slapd.conf** (or **slapd32.conf**) file to its pre-setup state. Run the **mksecldap** command with the **-U** flag after any unsuccessful setup attempt before trying to run the **mksecldap** command again. Otherwise, residual setup information might remain in the configuration file and cause a subsequent setup to fail. As a safety precaution, the undo option does not do anything to the database or to its data, because the database could have existed before the **mksecldap** command was run. You must remove any database manually if it was created by the **mksecldap** command. If the **mksecldap** command has added data to a pre-existing database, you must decide what steps to recover from a failed setup attempt.

For more information on setting up a LDAP security information server, refer the **mksecldap** command documentation.

Setting Up an LDAP Client

Each client must have the LDAP client package installed. If the SSL is required, the GSKit must be installed, a key must be created, and the LDAP server SSL key certificate must be added to this key.

The **mksecldap** command can be used to set up the client. To have this client contact the LDAP security information server, the server name must be supplied during setup. The server's administrator domain name and password are also needed for client access the AIX tree on the server. The **mksecldap** command saves the server administrator domain name, password, server name, AIX tree domain name on the server, and the SSL key path and password to the **/etc/security/ldap/ldap.cfg** file.

Multiple servers can be supplied to the **mksecldap** command during client setup. In this case, the client contacts the servers in the supplied order and establishes connection to the first server that the client can successfully bind to. If bad connection occurs between the client and the server, a reconnection request is tried using the same logic. The Security LDAP exploitation model does not support referral. It is important that the replicate servers are kept synchronized.

The client communicates to the LDAP security information server through a client side daemon (**secldapclntd**). If the LDAP load module is enabled on this client, high-level commands eventually find this daemon through the library APIs. The daemon queries the server and returns the information back to the caller.

Other fine-tuning options can be supplied to the **mksecldap** command during client setup, such as settings for the number of threads used by the daemon, the cache entry size, and the cache expiration timeout. These options are for experienced users only. For most environments, the default values are sufficient.

A comma-seperated user list can be supplied to the **mksecldap** command during client setup. These users' SYSTEM attributes are set to LDAP. Once this is done, these users can only authenticate through the LDAP load module. Note that the **mksecldap** command does not add these users to the LDAP security information server to avoid duplicate user IDs in the LDAP database. It is recommended that you use the **mkuser** command with **-R LDAP** flag to create these users on an LDAP server.

In the final steps of the client setup, the **mksecldap** command starts the client-side daemon and adds an entry in the **/etc/inittab** file so the daemon starts at every reboot. You can check whether the setup is successful by checking the **secldapclntd** process. Provided that the LDAP security information sever is setup and running, this daemon will be running if the setup was successful.

LDAP User Management

You can manage users and groups on an LDAP security information server from any LDAP client by using high-level commands. An **-R** flag added to most of the high-level commands can manage users and groups using LDAP as well as other authentication load modules such as DCE, NIS, and Kerberos 5. For more information concerning the use of the **-R** flag, refer to each of the user or group management commands.

To enable a user to authenticate through LDAP, run the **chuser** command to change the user's SYSTEM attribute value to LDAP. By setting the SYSTEM attribute value according the defined syntax, a user can be authenticated through more than one load module (for example, compat and LDAP). For more information on setting users' authentication methods, refer to "Security Administration" on page 11 and the SYSTEM attribute syntax defined in the **/etc/security/user** file.

A user can be made a LDAP user at client setup time by running the **mksecldap** command with the **-u** flag in either of the following forms:

1. Run `mksecldap -c -u` ***`user1,user2,...`*** , where **user1,user2,...** is a list of users. The users in this list can be either locally defined or remotely LDAP-defined users. A "SYSTEM = LDAP" entry is added to each of the above users' stanzas in the **/etc/security/user** file. Such users are only authenticated through LDAP. Run the **chuser** command to modify the SYSTEM attribute and allow authentication through multiple methods (for example, both local and LDAP). The users in this list must exist on the LDAP security information server; otherwise, they can not log in from this host.
2. Run "mksecldap -c -u ALL" . This command adds a "SYSTEM = ″LDAP″" entry to each user's stanza in the **/etc/security/user** file for all locally defined users. All suche users only authenticate through LDAP. The locally defined users must exist on the LDAP security information server; otherwise they can not login from this host. A user that is defined on the LDAP server but not defined locally cannot login from this host. To allow a remotely LDAP-defined user to login from this host, run the **chuser** command with "SYSTEM = ″LDAP″" for that user.

Alternatively, you can enable all LDAP users, whether they are defined locally or not, to authenticate through LDAP on a local host by modifying the ″default″ stanza of the **/etc/security/user** file to use ″LDAP″ as its value. All users that do not have a value defined for their SYSTEM attribute must follow what is defined in the default stanza. For example, if the default stanza has "SYSTEM = ″compat″" , changing it to "SYSTEM = ″compat OR LDAP″" allows authentication of these users either through AIX or LDAP. Changing the default stanza to "SYSTEM = ″LDAP″" enables these users to authenticate exclusively through LDAP. Those users who have a SYSTEM attribute value defined are not affected by the default stanza.

LDAP Security Information Server Auditing

IBM SecureWay Directory version 3.2 provides a default server audit logging functionality. Once enabled, this default audit plugin logs LDAP server activities to a log file. Refer to the LDAP documentation in the *Packaging Guide for LPP Installation* for more information on this default audit plugin.

For your convenience, an LDAP security information server auditing functionality has been implemented in AIX 5.1 and later, called the *LDAP security audit plugin*. It is independent of the IBM SecureWay Directory default auditing service, so you can enable either or both of these auditing subsystems. The AIX audit plugin records only those events that update or query the AIX security information on an LDAP server. It works within the framework of AIX system auditing.

To accomodate LDAP, the following audit events have been added to the **/etc/security/audit/event** file:

- **LDAP_Bind**
- **LDAP_Unbind**
- **LDAP_Add**
- **LDAP_Delete**
- **LDAP_Modify**
- **LDAP_Modifydn**
- **LDAP_Search**

An **ldapserver** audit class definition is also created in the **/etc/security/audit/config** file that contains all of the above events.

To audit the LDAP security information server, add the following line to each user's stanza in the **/etc/security/audit/config** file:

```
ldap = ldapserver
```

Because the LDAP security information server audit plugin is implemented within the frame of the AIX system auditing, it actually is part of the AIX system auditing subsystem. You can enable or disable the LDAP security information server audit using system audit commands, such as "audit start" or "audit shutdown". All audit records are added to the system audit trails, which can be reviewed by using the **auditpr** command. For more information, refer to the "Auditing Overview" on page 30.

Chapter 4. Administrative Roles

AIX 4.3 supports assigning portions of root user authority to non-root users. Different root user tasks are assigned different authorizations. These authorizations are grouped into roles and assigned to different users.

This chapter covers the following topics:

- "Roles Overview"
- "Understanding Authorizations".

Roles Overview

Roles consist of authorizations that allow a user to run functions that normally would require root user permission.

The following is a list of valid roles:

Add and Remove Users	Allows any user to act as the root user for this role. They are able to add and remove users, change information about a user, modify audit classes, manage groups, and change passwords. Anyone who performs user administration must be in the **security** group.
Change Users Password	Allows a user to change a passwords.
Manage Roles	Allows a user to create, change, remove and list roles. The user must be in the **security** group.
Backup and Restore	Allows a user to back up and restore file systems and directories. This role requires authorizations to enable a system backup and restore.
Backup Only	Allows a user only to back up file systems and directories. The user must have the proper authorization to enable a system backup.
Run Diagnostics	Allows a user or service representative to run diagnostics and diagnostic tasks. The user must have **system** specified as the primary group and also a group set that includes **shutdown**. **Note:** Users in the Run Diagnostics role can change the system configuration, update microcode, and so on. Users in this role must understand the responsibility the role requires.
System Shutdown	Allows a user to shut down, reboot, or halt a system.

Understanding Authorizations

Authorizations are authority attributes for a user. These authorizations allow a user to do certain tasks. For example, a user with the UserAdmin authorization can create an administrative user by running the **mkuser** command. A user without this authority cannot create an administrative user.

Authorization has two types:

Primary Authorization	Allows a user to run a specific command. For example, RoleAdmin authorization is a primary authorization allowing a user administrator to run the **chrole** command. Without this authorization, the command terminates without modifying the role definitions.
Authorization modifier	Increases the capability of a user. For example, UserAdmin authorization is an authorization modifier that increases the capability of a user administrator belonging to the **security** group. Without this authorization, the **mkuser** command only creates non-administrative users. With this authorization, the **mkuser** command also creates administrative users.

The authorizations perform the following functions:

Backup
Performs a system backup.

The following command uses the Backup authorization:

Backup	Backs up files and file systems. The user administrator must have Backup authorization.

Diagnostics
Allows a user to run diagnostics. This authority is also required to run diagnostic tasks directly from the command line.

The following command uses the Diagnostics authorization:

diag	Runs diagnostics on selected resources. If the user administrator does not have Diagnostics authority, the command ends.

GroupAdmin
Performs the functions of the root user on group data.

The following commands use the GroupAdmin authorization:

chgroup	Changes any group information. If the user does not have GroupAdmin authorization, they can only change non-administrative group information.
chgrpmem	Administers all groups. If the group administrator does not have GroupAdmin authorization, they can only change the membership of the group they administer or a user in group security to administer any non-administrative group.
chsec	Modifies administrative group data in the **/etc/group** and **/etc/security/group** files. The user can also modify the default **stanza values**. If the user does not have GroupAdmin authorization, they can only modify non-administrative group data in the **/etc/group** and **/etc/security/group** files.
mkgroup	Creates any group. If the user does not have GroupAdmin authorization, the user can only create non-administrative groups.
rmgroup	Removes any group. If the user does not have GroupAdmin authorization, the user can only remove non-administrative groups.

ListAuditClasses
Views the list of valid audit classes. The user administrator who uses this authorization does not have to be the root user or in the **audit** group.

Use the **smit mkuser** or **smit chuser** fast path to list audit classes available to make or change a user. Enter the list of audit classes in the AUDIT classes field.

PasswdAdmin
Performs the functions of the root user on password data.

The following commands use the PasswdAdmin authorization:

	chsec	Modifies the **lastupdate** and **flags** attributes of all users. Without the PasswdAdmin authorization, the **chsec** command allows the user administrator to only modify the **lastupdate** and **flags** attribute of non-administrative users.
	lssec	Views the **lastupdate** and **flags** attributes of all users. Without the PasswdAdmin authorization, the **lssec** command allows the user administrator to only view the **lastupdate** and **flags** attribute of non-administrative users.
	pwdadm	Changes the password of all users. The user administrator must be in group security.
PasswdManage	Performs password administration functions on non-administrative users. The following command uses the PasswdManage authorization:	
	pwdadm	Changes the password of a non-administrative user. The administrator must be in group security or have the PasswdManage authorization.
UserAdmin	Performs the functions of the root user on user data. Only users with UserAdmin authorization can modify the role information of a user. You cannot access or modify user auditing information with this authorization. The following commands use the UserAdmin authorization:	
	chfn	Changes any user general information (gecos) field. If the user does not have UserAdmin authorization but is in group security, they can change any non-administrative user gecos field. Otherwise, users can only change their own gecos field.
	chsec	Modifies administrative user data in the **/etc/passwd**, **/etc/security/environ**, **/etc/security/lastlog**, **/etc/security/limits**, and **/etc/security/user** files including the roles attribute. The user administrator can also modify the default stanza values and the **/usr/lib/security/mkuser.default** file, excluding the auditclasses attributes.
	chuser	Changes any user's information except for the auditclasses attribute. If the user does not have UserAdmin authorization, they can only change non-administrative user information, except for the auditclasses and roles attributes.

mkuser	Creates any user, except for the auditclasses attribute. If the user does not have UserAdmin authorization, the user can only create non-administrative users, except for the auditclasses and roles attributes.
rmuser	Removes any user. If the user administrator does not have UserAdmin authorization, they can only create non-administrative users.

UserAudit

Allows the user to modify user-auditing information.

The following commands use the UserAudit authorization:

chsec	Modifies the auditclasses attribute of the **mkuser.default** file for non-administrative users. If the user has UserAdmin authorization, they can also modify the auditclasses attribute of the **mkuser.default** file for administrative and non-administrative users.
chuser	Modifies the auditclasses attribute of a non-administrative user. If the user administrator has UserAdmin authorization, they can also modify the auditclasses attribute of all users.
lsuser	Views the auditclasses attribute of a non-administrative user if the user is root user or in group security. If the user has UserAdmin authorization, they can also view the auditclasses attribute of all users.
mkuser	Creates a new user and allows user administrator to assign the auditclasses attribute of a non-administrative user. If the user has UserAdmin authorization, they can also modify the auditclasses attribute of all users.

RoleAdmin

Performs the functions of the root user on role data.

The following commands use the RoleAdmin authorization:

chrole	Modifies a role. If the user administrator does not have the RoleAdmin authorization, the command ends.
lsrole	Views a role.
mkrole	Creates a role. If the user administrator does not have the RoleAdmin authorization, the command ends.
rmrole	Removes a role. If the user administrator does not have the RoleAdmin authorization, the command ends.

Restore

Performs a system restoration.

The following command uses the Restore authorization:

Restore	Restores backed-up files. The user administrator must have Restore authorization.

See Command to Authorization List in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices* for a mapping of commands to authorizations.

Chapter 5. Disk Quota System

This chapter contains the following topics:
- “Understanding the Disk Quota System”
- “Recovering from Over-Quota Conditions”
- “Implementing the Disk Quota System” on page 46.

Disk Quota System Overview

Note: The information in this section is specific to POWER-based.

The disk quota system allows system administrators to control the number of files and data blocks that can be allocated to users or groups. The following sections provide further information about the disk quota system, its implementation, and use:
- “Understanding the Disk Quota System”
- “Recovering from Over-Quota Conditions”
- “Implementing the Disk Quota System” on page 46

Understanding the Disk Quota System

The disk quota system, based on the Berkeley Disk Quota System, provides an effective way to control the use of disk space. The quota system can be defined for individual users or groups, and is maintained for each journaled file system.

The disk quota system establishes limits based on three parameters that can be changed with the **edquota** command:
- User’s or group’s soft limits
- User’s or group’s hard limits
- Quota grace period.

The *soft limit* defines the number of 1 KB disk blocks or files below which the user must remain. The *hard limit* defines the maximum amount of disk blocks or files the user can accumulate under the established disk quotas. The *quota grace period* allows the user to exceed the soft limit for a short period of time (the default value is one week). If the user fails to reduce usage below the soft limit during the specified time, the system will interpret the soft limit as the maximum allocation allowed, and no further storage is allocated to the user. The user can reset this condition by removing enough files to reduce usage below the soft limit.

The disk quota system tracks user and group quotas in the **quota.user** and **quota.group** files that reside in the root directories of file systems enabled with quotas. These files are created with the **quotacheck** and **edquota** commands and are readable with the quota commands.

Recovering from Over-Quota Conditions

There are several methods available to reduce file system usage when you have exceeded quota limits:
- Abort the current process that caused the file system to reach its limit, remove surplus files to bring the limit below quota, and retry the failed program.
- If you are running an editor such as vi, use the shell escape sequence to check your file space, remove surplus files, and return without losing your edited file. Alternatively, if you are using the C or Korn shells, you can suspend the editor with the Ctrl-Z key sequence, issue the file system commands, and then return with the **fg** (foreground) command.

- Temporarily write the file to a file system where quota limits have not been exceeded, delete surplus files, and then return the file to the correct file system.

Implementing the Disk Quota System

You should consider implementing the disk quota system under the following conditions:
- Your system has limited disk space.
- You require more file system security.
- Your disk-usage levels are large, such as at many universities.

If these conditions do not apply to your environment, you might not want to create disk-usage limits by implementing the disk quota system.

Typically, only those file systems that contain user home directories and files require disk quotas. The disk quota system works only with the journaled file system.

Note: Do not establish disk quotas for the **/tmp** file system.

Chapter 6. Logical Volumes

This chapter describes concepts for managing logical volume storage. This chapter covers the following topics:

Logical Volume Storage Overview

A hierarchy of structures is used to manage fixed-disk storage. Each individual fixed-disk drive, called a *physical volume* (PV) has a name, such as `/dev/hdisk0`. Every physical volume in use belongs to a *volume group* (VG). All of the physical volumes in a volume group are divided into *physical partitions* (PPs) of the same size (by default 2 MB in volume groups that include physical volumes smaller than 300MB, 4 MB otherwise). For space-allocation purposes, each physical volume is divided into five regions (outer_edge, inner_edge, outer_middle, inner_middle and center). The number of physical partitions in each region varies, depending on the total capacity of the disk drive. If the volume group is created with **-B** option in **mkvg** command, the above limits increase to 128 physical volumes and 512 logical volumes.

Within each volume group, one or more *logical volumes* (LVs) are defined. Logical volumes are groups of information located on physical volumes. Data on logical volumes appears to be contiguous to the user but can be discontiguous on the physical volume. This allows file systems, paging space, and other logical volumes to be resized or relocated, to span multiple physical volumes, and to have their contents replicated for greater flexibility and availability in the storage of data.

Each logical volume consists of one or more *logical partitions* (LPs). Each logical partition corresponds to at least one physical partition. If mirroring is specified for the logical volume, additional physical partitions are allocated to store the additional copies of each logical partition. Although the logical partitions are numbered consecutively, the underlying physical partitions are not necessarily consecutive or contiguous.

Logical volumes can serve a number of system purposes, such as paging, but each logical volume that holds ordinary system or user data or programs contains a single journaled file system (JFS). Each JFS consists of a pool of page-size (4KB) blocks. When data is to be written to a file, one or more additional blocks are allocated to that file. These blocks might not be contiguous with one another or with other blocks previously allocated to the file. In AIX 4.1 or later, a given file system can be defined as having a fragment size of less than 4 KB (512 bytes, 1 KB, 2 KB).

After installation, the system has one volume group (the rootvg volume group) consisting of a base set of logical volumes required to start the system and any others you specify to the installation script. Any other physical volumes you have connected to the system can be added to a volume group (using the **extendvg** command). You can add the physical volume either to the rootvg volume group or to another volume group (defined by using the **mkvg** command). Logical volumes can be tailored using the commands or the menu-driven System Management Interface Tool (SMIT) interface.

This overview contains information about the following:

  - "Logical Volumes" on page 50
  - "Logical Partitions" on page 51
  - "File Systems" on page 51
- "Logical Volume Manager" on page 52
- "Quorum Concepts" on page 52
  - "Vary-On Process" on page 52
  - "Quorum" on page 52
  - "Forcibly Varying On" on page 53
  - "Nonquorum Volume Groups" on page 54.

Logical Volume Storage Concepts

The five basic logical storage concepts are:
- "Physical Volumes" on page 49
- "Volume Groups" on page 49
- "Physical Partitions" on page 50
- "Logical Volumes" on page 50
- "Logical Partitions" on page 51

The following figure illustrates the relationships among these concepts.

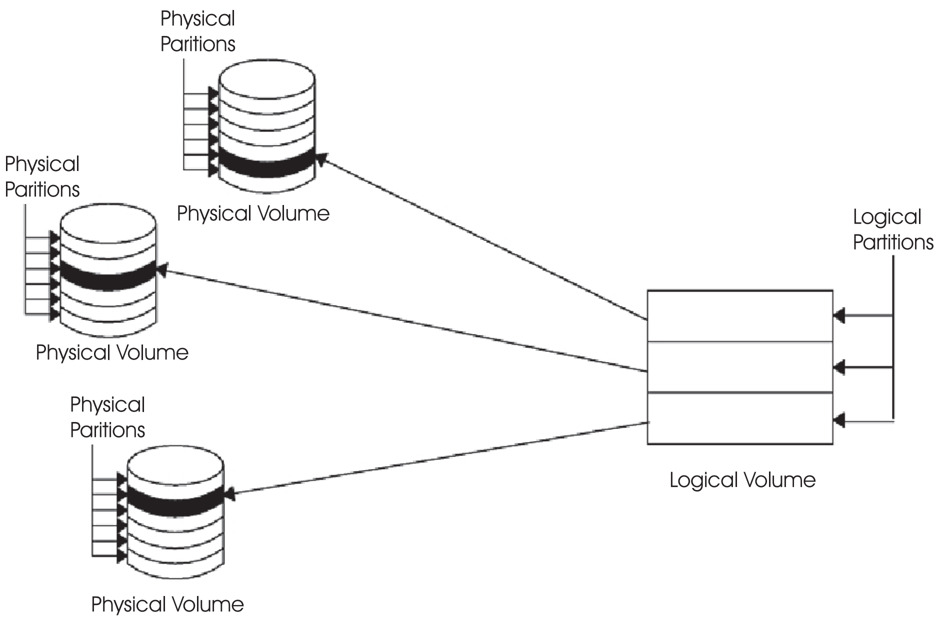

Figure 3. Volume Group. This illustration shows a volume group composed of three physical volumes with the maximum range specified. The logical volume (which can span physical volumes) is composed of logical partitions allocated onto physical partitions.

Physical Volumes

A disk must be designated as a physical volume and be put into an available state before it can be assigned to a volume group. A physical volume has certain configuration and identification information written on it. This information includes a physical volume identifier that is unique to the system. When a disk becomes a physical volume, it is divided into 512-byte *physical blocks*. You designate a disk as a physical volume with the **mkdev** or **chdev** commands or by using the System Management Interface Tool (SMIT) to add a physical volume.

The first time you start up the system after connecting a new disk, the operating system detects the disk and examines it to see if it already has a unique physical volume identifier in its boot record. If it does, the disk is designated as a physical volume and a physical volume name (typically, hdisk*x* where *x* is a unique number on the system) is permanently associated with that disk until you undefine it.

Volume Groups

The physical volume must now become part of a *volume group*. A volume group is a collection of 1 to 32 physical volumes of varying sizes and types. A physical volume can belong to only one volume group per system; there can be up to 255 volume groups per system.

When a physical volume is assigned to a volume group, the physical blocks of storage media on it are organized into physical partitions of a size you specify when you create the volume group. See “Physical Partitions” on page 50 for more information.

When you install the system, one volume group (the root volume group, called rootvg) is automatically created**.** The rootvg contains a base set of logical volumes required to start the system plus any other logical volumes you specify to the install script. The rootvg volume group includes paging space, the journal log, boot data, and dump storage, each in its own separate logical volume. The rootvg has attributes that differ from other, user-defined volume groups. For example, the rootvg cannot be imported or exported. When performing a command or procedure on the rootvg, you need to be familiar with its unique characteristics.

You create a new volume group with the **mkvg** command. You add a physical volume to a volume group with the **extendvg** command and remove it from a volume group with the **reducevg** command. Some of the other commands that you use on volume groups include: change (**chvg**), list (**lsvg**), remove (**exportvg**), install (**importvg**), reorganize (**reorgvg**), synchronize (**syncvg**), make available for use (**varyonvg**), and make unavailable for use (**varyoffvg**).

Small systems mightrequire only one volume group to contain all the physical volumes attached to the system. You might want to create separate volume groups, however, for security reasons, because each volume group can have its own security permissions. Separate volume groups also make maintenance easier because groups other than the one being serviced can remain active. Because the rootvg must always be online, it contains only the minimum number of physical volumes necessary for system operation.

You can move data from one physical volume to other physical volumes *in the same volume group* with the **migratepv** command. This command allows you to free a physical volume so it can be removed from the volume group. For example, you could move data off a physical volume that is to be replaced.

A VG that is created with smaller physical and logical volume limits can be converted to big format which can hold more PVs (upto 128) and more LVs (upto 512). This operation requires that there be enough free partitions on every PV in the VG for the Volume group descriptor area (VGDA) expansion. The number of free partitions required depends on the size of the current VGDA and the physical partition size. Because the VGDA resides on the edge of the disk and it requires contiguous space, the free partitions are required on the edge of the disk. If those partitions are allocated for user usage, they are migrated to other free partitions on the same disk. The rest of the physical partitions are renumbered to reflect the loss of the partitions for VGDA usage. This will change the mappings of the logical to physical partitions in all the PVs of this VG. If you have saved the mappings of the LVs for a potential recovery operation, generate the maps again after the completion of the conversion operation. Also, if the backup of the VG is taken with map option and you plan to restore using those maps, the restore operation might fail since the partition number might no longer exist (due to reduction). It is recommended that backup is taken before the conversion and right after the conversion if the map option is utilized. Because the VGDA space has been increased substantially, every VGDA update operation (creating an LV, changing an LV, adding a PV, and so on) might take considerably longer to run.

Note: Once you create a big volume group or convert a volume group to big volume group format, you cannot import it back to any level before to AIX 4.3.2.

Physical Partitions

When you add a physical volume to a volume group, the physical volume is partitioned into contiguous, equal-sized units of space called *physical partitions*. A physical partition is the smallest unit of storage space allocation and is a contiguous space on a physical volume.

Physical volumes inherit the volume group's physical partition size, which you can set only when you create the volume group (for example, using the **mkvg -s** command). The following figure shows the relationship between physical partitions on physical volumes and volume groups.

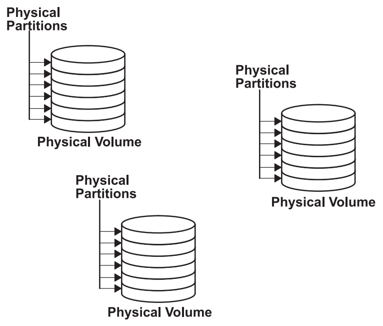

Figure 4. A Volume Group Containing Three Physical Volumes. This illustration shows three physical volumes, each with six physical partitions, within a single volume group.

Logical Volumes

After you create a volume group, you can create logical volumes within that volume group. A *logical volume,* although it can reside on noncontiguous physical partitions or even on more than one physical volume, appears to users and applications as a single, contiguous, extensible disk volume. You can create additional logical volumes with the **mklv** command. This command allows you to specify the name of the logical volume and define its characteristics, including the number and location of logical partitions to allocate for it.

After you create a logical volume, you can change its name and characteristics with the **chlv** command, and you can increase the number of logical partitions allocated to it with the **extendlv** command. The default maximum size for a logical volume at creation is 128 logical partitions, unless specified to be larger. The **chlv** command is used to relax this limitation.

Note: After you create a logical volume, the characteristic LV STATE, which can be seen using the **lslv** command, is closed. It becomes open when, for example, a file system has been created in the logical volume and mounted.

Logical volumes can also be copied with the **cplv** command, listed with the **lslv** command, removed with the**rmlv** command, and have the number of copies they maintain increased or decreased with the **mklvcopy** and the **rmlvcopy** commands respectively. They can also be relocated when the volume group is reorganized.

The system allows you to define up to 256 (512 in case of a big volume group) logical volumes per volume group, but the actual number you can define depends on the total amount of physical storage defined for that volume group and the size of the logical volumes you define.

Logical Partitions

When you create a logical volume, you specify the number of *logical partitions* for the logical volume. A logical partition is one, two, or three physical partitions, depending on the number of instances of your data you want maintained. Specifying one instance means there is only one copy of the logical volume (the default). In this case, there is a direct mapping of one logical partition to one physical partition. Each instance, including the first, is termed a copy. Where physical partitions are located (that is, how near each other physically) is determined by options you specify when you create the logical volume.

File Systems

The logical volume defines allocation of disk space down to the physical-partition level. Finer levels of data management are accomplished by higher level software components such as the Virtual Memory Manager or the file system. Therefore, the final step in the evolution of a disk is the creation of *file systems*. You can create one file system per logical volume. To create a file system, use the **crfs** command. For more information on file systems, see “File Systems Overview” on page 67.

Limitations for Logical Storage Management

The following table shows the limitations for logical storage management. Although the default maximum number of physical volumes per volume group is 32 (128 in case of big volume group), you can set the maximum for user-defined volume groups when you use the **mkvg** command. For the rootvg, however, this variable is automatically set to the maximum by the system during the installation.

```
MAXPVS: 32 (128 big volume group)
MAXLVS: 255 (512 big volume group)
```

Limitations for Logical Storage Management	
Volume group	255 per system
Physical volume	(MAXPVS / volume group factor) per volume group
Physical partition	(1016 x volume group factor) per physical volume up to 1024MB each in size
Logical volume	MAXLVS per volume group
Logical partition	(MAXPVS * 1016) per logical volume

If you previously created a volume group before the 1016 physical partitions per physical volume restriction was enforced, stale partitions in the volume group are not correctly tracked unless you convert the volume group to a supported state. You can convert the volume group with the **chvg -t** command. A suitable factor value is chosen by default to accommodate the largest disk in the volume group.

For example, if you created a volume group with a 9 GB disk and 4 MB partition size, this volume group will have approximately 2250 partitions. Using a conversion factor of 3 (1016 * 3 = 3048) allows all 2250 partitions to be tracked correctly. Converting a volume group with a higher factor enables inclusion of a larger disk of partitions up to the 1016* factor. You can also specify a higher factor when you create the volume group in order to accommodate a larger disk with a small partition size.

These operations reduce the total number of disks that you can add to a volume group. The new maximum number of disks you can add would be a MAXPVS/factor. For example, for a regular volume group, a factor of 2 decreases the maximum number of disks in the volume group to 16 (32/2). For a big volume group, a factor of 2 decreases the maximum number of disks in the volume group to 64 (128/2).

Note: Once you convert a volume group, you cannot import it back to any level before AIX 4.3.1.

Logical Volume Manager

The set of operating system commands, library subroutines, and other tools that allow you to establish and control logical volume storage is called the Logical Volume Manager (LVM). The Logical Volume Manager (LVM) controls disk resources by mapping data between a more simple and flexible *logical* view of storage space and the actual *physical* disks. The LVM does this using a layer of device driver code that runs above traditional disk device drivers.

The Logical Volume Manager (LVM) consists of the logical volume device driver (LVDD) and the LVM subroutine interface library. The *logical volume device driver* (LVDD) is a pseudo-device driver that manages and processes all I/O. It translates logical addresses into physical addresses and sends I/O requests to specific device drivers. The *LVM subroutine interface library* contains routines that are used by the system management commands to perform system management tasks for the logical and physical volumes of a system. The programming interface for the library is available to anyone who wishes to expand the function of the system management commands for logical volumes.

For more information about how LVM works, see Understanding the Logical Volume Device Driver in *AIX 5L Version 5.1 Kernel Extensions and Device Support Programming Concepts* and Logical Volume Programming Overview in *AIX 5L Version 5.1 General Programming Concepts: Writing and Debugging Programs*.

Quorum Concepts

The following sections describe the vary-on process and the quorum, which are how the LVM ensures that a volume group is ready to use and contains the most up-to-date data.

Vary-On Process

The **varyonvg** and **varyoffvg** commands activate or deactivate (make available or unavailable for use) a volume group that you have defined to the system. The volume group must be varied on before the system can access it. During the vary-on process (activation), the LVM reads management data from the physical volumes defined in the volume group. This management data, which includes a volume group descriptor area (VGDA) and a volume group status area (VGSA), is stored on each physical volume of the volume group.

The VGDA contains information that describes the mapping of physical partitions to logical partitions for each logical volume in the volume group, as well as other vital information, including a time stamp. The VGSA contains information such as which physical partitions are stale and which physical volumes are missing (that is, not available or active) when a vary-on operation is attempted on a volume group.

If the vary-on operation cannot access one or more of the physical volumes defined in the volume group, the command displays the names of all physical volumes defined for that volume group and their status. This helps you decide whether to continue operating with this volume group.

Quorum

A quorum is a vote of the number of Volume Group Descriptor Areas and Volume Group Status Areas (VGDA/VGSA) that are active. A quorum ensures data integrity of the VGDA/VGSA areas in the event of a disk failure. Each physical disk in a volume group has at least one VGDA/VGSA. When a volume group is created onto a single disk, it initially has two VGDA/VGSA areas residing on the disk. If a volume group consists of two disks, one disk still has two VGDA/VGSA areas, but the other disk has one VGDA/VGSA. When the volume group is made up of three or more disks, then each disk is allocated just one VGDA/VGSA.

A quorum is lost when enough disks and their VGDA/VGSA areas are unreachable so that a 51% majority of VGDA/VGSA areas no longer exists. In a two-disk volume group, if the disk with only one VGDA/VGSA is lost, a quorum still exists because two of the three VGDA/VGSA areas still are reachable. If the disk with two VGDA/VGSA areas is lost, this statement is no longer true. The more disks that make up a volume group, the lower the chances of quorum being lost when one disk fails.

When a quorum is lost, the volume group varies itself off so that the disks are no longer accessible by the Logical Volume Manager (LVM). This prevents further disk I/O to that volume group so that data is not lost or assumed to be written when physical problems occur. Additionally, as a result of the vary-off, the user is notified in the error log that a hardware error has occurred and service must be performed.

There are cases when it is desirable to continue operating the volume group even though a quorum is lost. In these cases, quorum checking can be turned off for the volume group. This type of volume group is referred to as a nonquorum volume group. The most common case for a nonquorum volume group occurs when the logical volumes have been mirrored. When a disk is lost, the data is not lost if a copy of the logical volume resides on a disk that is not disabled and can be accessed. However, there can be instances in nonquorum volume groups, mirrored or nonmirrored, when the data (including copies) resides on the disk or disks that have become unavailable. In those instances, the data might not be accessible even though the volume group continues to be varied on.

Forcibly Varying On

Attention: Overriding a vary-on failure is an unusual operation; check all other possible problem sources such as hardware, cables, adapters, and power sources before proceeding. Overriding the quorum failure during a vary-on process is used only in an emergency and only as a last resort (for example, to salvage data from a failing disk). Data integrity cannot be guaranteed for management data contained in the chosen copies of the VGDA and the VGSA when a quorum failure is overridden.

When you choose to forcibly vary-on a volume group by overriding the absence of a quorum, the PV STATE of all physical volumes that are missing during this vary-on process will be changed to removed. This means that all the VGDA and VGSA copies are removed from these physical volumes. After this is done, these physical volumes will no longer take part in quorum checking, nor are they allowed to become active within the volume group until you return them to the volume group.

Under one or more of the following conditions, you might want to override the vary-on failure so that the data on the available disks in the volume group can be accessed:

- Unavailable physical volumes appear permanently damaged.
- You can confirm that at least one of the presently accessible physical volumes (which must also contain a good VGDA and VGSA copy) was online when the volume group was last varied on. Unconfigure and power off the missing physical volumes until they can be diagnosed and repaired.

The following procedure provides a way to avoid losing quorum when one disk is missing or might soon fail and requires repair.

1. Use the **chpv -vr** command to temporarily remove the volume from the volume group. This physical volume is no longer factored in quorum checking. However, in a two-disk volume group, this command fails if you try the **chpv** command on the disk that contains the two VGDA/VGSAs. The command does not allow you to cause quorum to be lost.
2. If the goal was to remove the disk for repair, power off the system, and remove the disk. After fixing the disk and returning the disk to the system, run **chpv -v** in order to make it available to the volume group for quorum checking.

Note: The **chpv** command is used only for quorum-checking alteration. The data that resides on the disk is still there and must be moved or copied to other disks if the disk is not to be returned to the system.

Nonquorum Volume Groups

The Logical Volume Manager (LVM) automatically deactivates the volume group when it lacks a quorum of VGDAs or VGSAs. However, you can choose an option that allows the group to stay online as long as there is one VGDA/VGSA intact. This option produces a *nonquorum volume group.* The LVM requires access to all of the disks in nonquorum volume groups before allowing reactivation to ensure up-to-date VGDA and VGSA.

You might want to use this procedure in systems where every logical volume has at least two copies.

If a disk failure occurs, the volume group remains active as long as there is one logical volume copy intact on a disk.

Note: Both user-defined and rootvg volume groups can operate in nonquorum status, but the methods used to configure user-defined volume groups and rootvg volume groups as nonquorum and for recovery after hardware failures are different. Be sure you use the correct method for the appropriate volume group.

Developing a Volume Group Strategy

Disk failure is the most common hardware failure in the storage system, followed by failure of adapters and power supplies. Protection against disk failure primarily involves the configuration of the logical volumes. See “Developing a Logical Volume Strategy” on page 56 for more information. However, volume group size also plays a part.

To protect against adapter and power supply failure, consider a special hardware configuration for any specific volume group. Such a configuration includes two adapters and at least one disk per adapter, with mirroring across adapters, and a nonquorum volume group configuration. The additional expense of this configuration is not appropriate for all sites or systems. It is recommended only where high (up-to-the-last-second) availability is a priority. Depending on the configuration, high availability can cover hardware failures that occur between the most recent backup and the current data entry. High availability does not apply to files deleted by accident.

Prerequisites

It is important that you understand the material contained in the “Logical Volume Storage Overview” on page 47.

When to Create Separate Volume Groups

You might want to organize physical volumes into volume groups separate from rootvg for the following reasons:

- For safer and easier maintenance.
  - Operating system updates, reinstallations, and crash recoveries are safer because you can separate user file systems from the operating system so that user files are not jeopardized during these operations.
  - Maintenance is easier because you can update or reinstall the operating system without having to restore user data. For example, before updating, you can remove a user-defined volume group from the system by unmounting its file systems. Deactivate it using the **varyoffvg** command, then export the group using the **exportvg** command. After updating the system software, you can reintroduce the user-defined volume group using the **importvg** command, then remount its file systems.

- For different physical-partition sizes. All physical volumes within the same volume group must have the same physical partition size. To have physical volumes with different physical partition sizes, place each size in a separate volume group.
- When different quorum characteristics are required. If you have a file system for which you want to create a nonquorum volume group, maintain a separate volume group for that data; all of the other file systems should remain in volume groups operating under a quorum.
- To have multiple JFS logs or JFS logs dedicated on one physical volume for the purpose of reducing bottlenecks, especially on server machines.
- For security. For example, you might want to remove a volume group at night.
- To switch physical volumes between systems. If you create a separate volume group for each system on an adapter that is accessible from more than one system, you can switch the physical volumes between the systems that are accessible on that adapter without interrupting the normal operation of either (see the **varyoffvg**, **exportvg**, **importvg**, and **varyonvg** commands).
- To remove disks from the system while the system continues to run normally. By making a separate volume group for removable disks, provided the volume group is not rootvg, you can make removable disks unavailable and physically remove them during normal operation without affecting other volume groups.

High Availability in Case of Disk Failure

The primary methods used to protect against disk failure involve logical volume configuration settings, such as mirroring. While the volume group considerations are secondary, they have significant economic implications because they involve the number of physical volumes per volume group:

- The quorum configuration, which is the default, keeps the volume group active (varied on) as long as a quorum (51%) of the disks is present. For more information about quorum requirements, see the section on “Vary-On Process” on page 52 in the ″Logical Volume Storage Overview″ . In most cases, you need at least three disks with mirrored copies in the volume group to protect against disk failure.
- The nonquorum configuration keeps the volume group active (varied on) as long as one VGDA is available on a disk (see ″Changing a Volume Group to Nonquorum Status″). With this configuration, you need only two disks with mirrored copies in the volume group to protect against disk failure.

When deciding on the number of disks in each volume group, you also need to plan for room to mirror the data. Keep in mind that you can only mirror and move data between disks that are in the same volume group. If the site uses large file systems, finding disk space on which to mirror could become a problem at a later time. Be aware of the implications on availability of inter-disk settings for logical volume copies (see “Inter-Disk Settings for Logical Volume Copies” on page 60) and intra-disk allocation (see “Choosing an Intra-Disk Allocation Policy for Each Logical Volume” on page 62) for a logical volume.

High Availability in Case of Adapter or Power Supply Failure

To protect against adapter or power supply failure, depending on your requirements, do one or more of the following:

- Use two adapters, located in the same or different chassis. Locating the adapters in different chassis protects against losing both adapters if there is a power supply failure in one chassis.
- Use two adapters, attaching at least one disk to each adapter. This protects against a failure at either adapter (or power supply if adapters are in separate cabinets) by still maintaining a quorum in the volume group, assuming *cross-mirroring* (copies for a logical partition cannot share the same physical volume) between the logical volumes on disk A (adapter A) and the logical volumes on disk B (adapter B). This means that you copy the logical volumes that reside on the disks attached to adapter A to the disks that reside on adapter B and also that you copy the logical volumes that reside on the disks attached to adapter B to the disks that reside on adapter A as well.

- Configure all disks from both adapters into the same volume group. This ensures that at least one logical volume copy remains intact in case an adapter fails, or, if cabinets are separate, in case a power supply fails.
- Make the volume group a nonquorum volume group. This allows the volume group to remain active as long as one Volume Group Descriptor Area (VGDA) is accessible on any disk in the volume group. See Changing a Volume Group to Nonquorum Status for more information.
- If there are two disks in the volume group, implement cross-mirroring between the adapters. If more than one disk is available on each adapter, implement double-mirroring. In that case, you create a mirrored copy on a disk that uses the same adapter and one on a disk using a different adapter.

Deciding on the Size of Physical Partitions

The physical partition size is set when the volume group is created. The default size is 4 MB. The default is designed to suit most sites and systems but might not be appropriate in every case. You can choose a partition size as small as 1 MB to gain flexibility in sizing but this requires more partitions. The additional partitions create more overhead for the Logical Volume Manager (LVM) and are likely to affect performance.

If you make the partitions larger than 4 MB, you lose some sizing flexibility and might also waste space. For example, if you have 20 MB partitions, then your JFS log will have to be 20 MB when it only needs 4 MB. Some waste may be an acceptable tradeoff if the particular site or system requires larger partitions.

Note that you can only create and extend physical partitions in increments that are a factor of their size; for example, 20 MB partitions are created or extended in 20 MB increments.

Developing a Logical Volume Strategy

The policies described in this section help you set a strategy for logical volume use that is oriented toward a combination of availability, performance, and cost that is appropriate for your site.

Availability is the ability to recover data that is lost because of disk, adapter, or other hardware problems. The recovery is made from copies of the data that are made and maintained on separate disks and adapters during normal system operation.

Performance is the average speed at which data is accessed. Policies such as write-verify and mirroring enhance availability but add to the system processing load, and thus degrade performance. Mirroring doubles or triples the size of the logical volume. In general, increasing availability degrades performance. Disk striping can increase performance. Beginning with AIX 4.3.3, disk striping is allowed with mirroring.

By controlling the allocation of data on the disk and between disks, you can tune the storage system for the highest possible performance. See Monitoring and Tuning Memory Use, and Monitoring and Turning Disk I/O, in *AIX 5L Version 5.1 Performance Management Guide* for detailed information on how to maximize storage-system performance.

The sections that follow should help you evaluate the tradeoffs among performance, availability, and cost. Remember that increased availability often decreases performance, and vice versa. Mirroring may increase performance, however, if the LVM chooses the copy on the least busy disk for Reads.

> **Note:** Mirroring does not protect against the loss of individual files that are accidentally deleted or lost because of software problems. These files can only be restored from conventional tape or diskette backups.

This section discusses:
- “Choosing an Inter-Disk Allocation Policy for Your System” on page 59

- “Choosing an Intra-Disk Allocation Policy for Each Logical Volume” on page 62
- “Combining Allocation Policies” on page 62
- “Using Map Files for Precise Allocation” on page 62
- “Determining a Write-Verify Policy” on page 63.

Prerequisites

It is important that you understand the material contained in the “Logical Volume Storage Overview” on page 47.

Analyzing Needs for Performance and Availability

Determine whether the data that is stored in the logical volume is valuable enough to warrant the processing and disk-space costs of mirroring.

Performance and mirroring are not always opposed. If the different instances (copies) of the logical partitions are on different physical volumes, preferably attached to different adapters, the LVM can improve Read performance by reading the copy on the least busy disk. Write performance, unless disks are attached to different adapters, always cost the same because you must update all copies. It is only necessary to read one copy for a Read operation.

If you have a large sequential-access file system that is performance-sensitive, you may want to consider disk striping.

Normally, whenever data on a logical partition is updated, all the physical partitions containing that logical partition are automatically updated. However, physical partitions can become *stale* (no longer containing the most current data) because of system malfunctions or because the physical volume was unavailable at the time of an update. The LVM can refresh stale partitions to a consistent state by copying the current data from an up-to-date physical partition to the stale partition. This process is called *mirror synchronization*. The refresh can take place when the system is restarted, when the physical volume comes back online, or when you issue the **syncvg** command.

While mirroring improves storage system availability, it is not intended as a substitute for conventional tape backup arrangements.

Beginning with AIX 4.3.3, the startup logical volume can be mirrored.

Any change that affects the physical partition makeup of a boot logical volume requires that you run **bosboot** after that change. This means that actions such as changing the mirroring of a boot logical volume require a **bosboot**.

Avoid dumping to a mirrored logical volume because it results in an inconsistent dump. Because the default dump device is the primary paging logical volume, create a separate dump logical volume if you mirror your paging logical volumes. If you mirror your root volume group, also create a separate dump logical volume.

Determining Scheduling Policy for Mirrored Writes to Disk

For data that has only one physical copy, the logical volume device driver (LVDD) translates a logical Read or Write request address into a physical address and calls the appropriate physical device driver to service the request. This single-copy or nonmirrored policy handles bad block relocation for Write requests and returns all Read errors to the calling process.

If you use mirrored logical volumes, four different scheduling policies for writing to disk can be set for a logical volume with multiple copies. They are sequential, parallel, parallel write with sequential read, and parallel write with round robin read.

Sequential-scheduling policy
Performs Writes to multiple copies or mirrors in order. The multiple physical partitions representing the mirrored copies of a single logical partition are designated primary, secondary, and tertiary. In sequential scheduling, the physical partitions are written to in sequence. The system waits for the Write operation for one physical partition to complete before starting the Write operation for the next one. When all write operations have been completed for all mirrors, the Write operation is complete.

Parallel-scheduling policy
Simultaneously starts the Write operation for all the physical partitions in a logical partition. When the Write operation to the physical partition that takes the longest to complete finishes, the Write operation is completed. Specifying mirrored logical volumes with a parallel-scheduling policy might improve I/O read-operation performance, because multiple copies allow the system to direct the Read operation to the copy that can be most quickly accessed.

Parallel write with sequential read-scheduling policy
Simultaneously starts the Write operation for all the physical partitions in a logical partition. The primary copy of the read is always read first. If that Read operation is unsuccessful, the next copy is read. During the Read retry operation on the next copy, the failed primary copy is corrected by the LVM with a hardware relocation. This patches the bad block for future access.

Parallel write with round robin read-scheduling policy
Simultaneously starts the Write operation for all the physical partitions in a logical partition. Reads are switched back and forth between the mirrored copies.

Determine Mirror Write Consistency (MWC) Policy for a Logical Volume

When Mirror Write Consistency (MWC) is turned **ON**, logical partitions that might be inconsistent if the system or the volume group is not shut down properly are identified. When the volume group is varied back online, this information is used to make logical partitions consistent. This is referred to as active MWC.

When a logical volume is using active MWC, then requests for this logical volume are held within the scheduling layer until the MWC cache blocks can be updated on the target physical volumes. When the MWC cache blocks have been updated, the request proceeds with the physical data Write operations. Only the disks where the data actually resides must have these MWC cache blocks written to it before the Write can proceed.

When active MWC is being used, system performance can be adversely affected. Adverse affects are caused by the overhead of logging or journaling that a Write request in which a Logical Track Group (LTG) is active. The allowed LTG sizes for a volume group are 128 K, 2456 K, 512 K, and 102 4K, with the default size as 128 K.

Note: To have an LTG size greater than 128 K, the disks contained in the volume group must support I/O requests of this size from the disk's strategy routines. The LTG is a contiguous block contained within the logical volume and is aligned on the size of the LTG. All I/Os must be no larger than an LTG and must be contained in a single LTG. This overhead is for mirrored Writes only.

It is necessary to guarantee data consistency between mirrors only if the system or volume group crashes before the Write to all mirrors has been completed. All logical volumes in a volume group share the MWC log. The MWC log is maintained on the outer edge of each disk. Locate the logical volumes that use Active MWC at the outer edge of the disk so that the logical volume is in close proximity to the MWC log on disk.

When MWC is set to passive, the volume group logs that the logical volume has been opened. After a crash when the volume group is varied on, an automatic force sync of the logical volume is started.

Consistency is maintained while the force sync is in progress by using a copy of the read recovery policy that propagates the blocks being read to the other mirrors in the logical volume. This policy is only supported on the BIG volume group type.

When MWC is turned **OFF**, the mirrors of a mirrored logical volume can be left in an inconsistent state in the event of a system or volume group crash. There is no automatic protection of mirror consistency. Writes outstanding at the time of the crash can leave mirrors with inconsistent data the next time the volume group is varied on. After a crash, any mirrored logical volume that has MWC turned **OFF** should perform a forced sync before the data within the logical volume is used. For example,

```
syncvg -f -l LTVname
```

An exception to forced sync is logical volumes whose content is only valid while the logical volume is open, such as paging spaces.

A mirrored logical volume is no different from a non-mirrored logical volume with respect to a Write. When LVM completely finishes with a Write request, the data has been written to all the drive(s) below LVM. The outcome of the Write is unknown until LVM issues an **iodone** on the Write. After this is complete, no recovery after a crash is necessary. Any blocks being written that have not been completed (**iodone**) when a machine crashes should be checked and rewritten whether or not they are mirrored, regardless of the MWC setting.

Because a mirrored logical volume is no different from a non-mirrored logical volume, there is no such thing as latest data. All applications that care about data validity need to determine the validity of the data of outstanding or inflight writes that did not complete before the volume group or system crashed, whether or not the logical volume was mirrored.

Active and passive MWC make mirrors consistent only when the volume group is varied back online after a crash by picking one mirror and propagating that data to the other mirrors. These MWC policies do not keep track of the latest data. active MWC only keeps track of LTGs currently being written, therefore MWC does not guarantee that the latest data will be propagated to all the mirrors. Passive MWC makes mirrors consistent by going into a propagate-on-read mode after a crash. It is the application above LVM that has to determine the validity of the data after a crash. From the LVM prospective, if the application always reissues all outstanding Writes from the time of the crash, the possibly inconsistent mirrors will be consistent when these Writes finish, (as long as the same blocks are written after the crash as were outstanding at the time of the crash).

Note: Mirrored logical volumes containing either JFS logs or file systems must be synchronized after a crash either by forced sync before use, by turning MWC on, or turning passive MWC on.

Choosing an Inter-Disk Allocation Policy for Your System

The inter-disk allocation policy specifies the number of disks on which the physical partitions of a logical volume are located. The physical partitions for a logical volume might be located on a single disk or spread across all the disks in a volume group. Two options in the **mklv** and **chlv** commands are used to determine inter-disk policy:

- The *Range* option determines the number of disks used for a single physical copy of the logical volume.
- The *Strict* option determines whether the **mklv** operation succeeds if two or more copies must occupy the same physical volume.
- Striped logical volumes can only have a maximum range and a strict inter-disk policy.

Inter-Disk Settings for a Single Copy of the Logical Volume

If you select the minimum inter-disk setting (`Range = minimum`), the physical partitions assigned to the logical volume are located on a single disk to enhance availability. If you select the maximum inter-disk setting (`Range = mimum`), the physical partitions are located on multiple disks to enhance performance. The allocation of mirrored copies of the original partitions is discussed in the following section.

For nonmirrored logical volumes, use the **minimum** setting to provide the greatest availability (access to data in case of hardware failure). The **minimum** setting indicates that one physical volume contains all the original physical partitions of this logical volume if possible. If the allocation program must use two or more physical volumes, it uses the minimum number, while remaining consistent with other parameters.

By using the minimum number of physical volumes, you reduce the risk of losing data because of a disk failure. Each additional physical volume used for a single physical copy increases that risk. A nonmirrored logical volume spread across four physical volumes is four times as likely to lose data because of one physical volume failure than a logical volume contained on one physical volume.

The following figure illustrates a minimum inter-disk allocation policy.

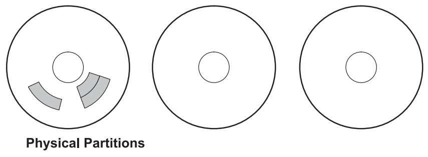

Figure 5. Minimum Inter-Disk Allocation Policy. This illustration shows three disks. One disk contains three physical partitions; the others have no physical partitions.

The maximum setting, considering other constraints, spreads the physical partitions of the logical volume as evenly as possible over as many physical volumes as possible. This is a performance-oriented option, because spreading the physical partitions over several disks tends to decrease the average access time for the logical volume. To improve availability, the maximum setting is only used with mirrored logical volumes.

The following figure illustrates a maximum inter-disk allocation policy.

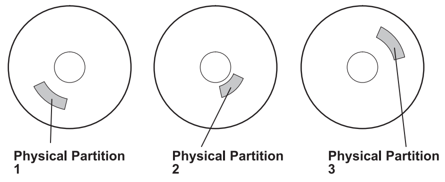

Figure 6. Maximum Inter-Disk Allocation Policy. This illustration shows three disks, each containing one physical partition.

These definitions are also applicable when extending or copying an existing logical volume. The allocation of new physical partitions is determined by your current allocation policy and where the existing used physical partitions are located.

Inter-Disk Settings for Logical Volume Copies

The allocation of a single copy of a logical volume on disk is fairly straightforward. When you create mirrored copies, however, the resulting allocation is somewhat complex. The figures that follow show minimum maximum and inter-disk (Range) settings for the first instance of a logical volume along with the available Strict settings for the mirrored logical volume copies.

For example, if there are mirrored copies of the logical volume, the minimum setting causes the physical partitions containing the first instance of the logical volume to be allocated on a single physical volume, if possible. Then, depending on the setting of the Strict option, the additional copy or copies are allocated on the same or on separate physical volumes. In other words, the algorithm uses the minimum number of physical volumes possible, within the constraints imposed by other parameters such as the Strict option, to hold all the physical partitions.

The setting `Strict = y` means that each copy of the logical partition is placed on a different physical volume. The setting `Strict = n` means that the copies are not restricted to different physical volumes.

> **Note:** If there are fewer physical volumes in the volume group than the number of copies per logical partition you have chosen, set Strict to **n**. If Strict is set to **y**, an error message is returned when you try to create the logical volume.

The following figure illustrates a minimum inter-disk allocation policy with differing Strict settings:

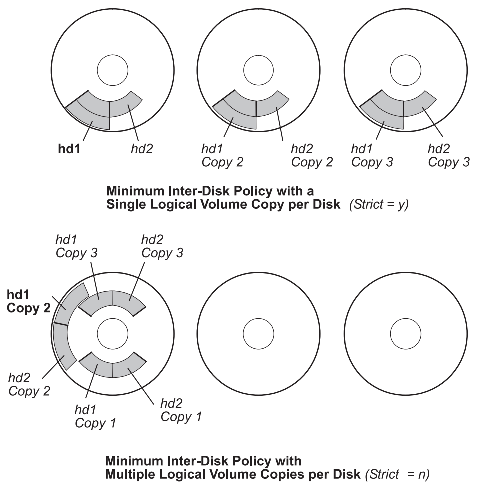
hd1
hd2
hd1 Copy 2
hd2 Copy 2
hd1 Copy 3
hd2 Copy 3
Minimum Inter-Disk Policy with a Single Logical Volume Copy per Disk (Strict = y)
hd1 Copy 3
hd2 Copy 3
hd1 Copy 2
hd2 Copy 2
hd1 Copy 1
hd2 Copy 1
Minimum Inter-Disk Policy with Multiple Logical Volume Copies per Disk (Strict = n)

Figure 7. Minimum Inter-Disk Policy/Strict. This illustration shows that if Strict is equal to Yes, each copy of the logical partition is on a different physical volume. If Strict is equal to No, all copies of the logical partitions are on a single physical volume.

The following figure illustrates a maximum inter-disk allocation policy with differing *Strict* settings:

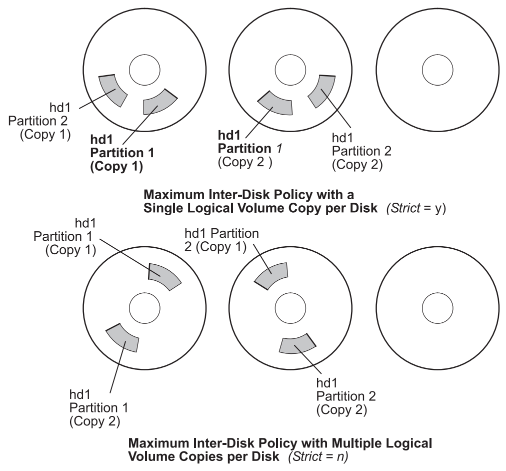
hd1 Partition 2 (Copy 1)
hd1 Partition 1 (Copy 1)
hd1 Partition 1 (Copy 2)
hd1 Partition 2 (Copy 2)
Maximum Inter-Disk Policy with a Single Logical Volume Copy per Disk (Strict = y)
hd1 Partition 1 (Copy 1)
hd1 Partition 2 (Copy 1)
hd1 Partition 1 (Copy 2)
hd1 Partition 2 (Copy 2)
Maximum Inter-Disk Policy with Multiple Logical Volume Copies per Disk (Strict = n)

Figure 8. Maximum Inter-Disk Policy/Strict. This illustration shows that if Strict is equal to Yes, each copy of a partition is on a separate physical volume. If Strict is equal to No, all copies are on a single physical volume.

Choosing an Intra-Disk Allocation Policy for Each Logical Volume

The closer a given physical partition is to the center of a physical volume, the lower the average seek time because the center has the shortest average seek distance from any other part of the disk.

The file system log is a good candidate for allocation at the center of a physical volume because it is used by the operating system so often. At the other extreme, the boot logical volume is used infrequently and therefore is allocated at the edge or middle of the physical volume.

The general rule is that the more I/Os, either absolutely or during the running of an important application, the closer to the center of the physical volumes the physical partitions of the logical volume needs to be allocated. This rule has two important exceptions:

1. Logical volumes on 200 MB, 540 MB, or 1 GB disks that contain large, sequential files are at the edge because sequential performance is better there (there are more blocks per track at the edge than farther in).
2. Mirrored logical volumes with Mirror Write Consistency (MWC) set to On are at the outer edge because that is where the system writes MWC data. If mirroring is not in effect, MWC does not apply and does not affect performance.

The intra-disk allocation policy choices are based on the five regions of a disk where physical partitions can be located. The five regions are:

1. *outer edge*
2. *inner edge,*
3. outer middle
4. *inner middle*
5. *center*

The edge partitions have the slowest average seek times, which generally result in longer response times for any application that uses them. The center partitions have the fastest average seek times, which generally result in the best response time for any application that uses them. There are, however, fewer partitions on a physical volume at the center than at the other regions.

Combining Allocation Policies

If you select inter-disk and intra-disk policies that are not compatible, you might get unpredictable results. The system assigns physical partitions by allowing one policy to take precedence over the other. For example, if you choose an intra-disk policy of center and an inter-disk policy of minimum, the inter-disk policy takes precedence. The system places all of the partitions for the logical volume on one disk if possible, even if the partitions do not all fit into the center region. Make sure you understand the interaction of the policies you choose before implementing them.

Using Map Files for Precise Allocation

If the default options provided by the inter- and intra-disk policies are not sufficient for your needs, consider creating map files to specify the exact order and location of the physical partitions for a logical volume.

You can use Web-based System Manager, SMIT, or the **-m** option for the **mklv** command to create map files.

Note: The **-m** option is not permitted with disk striping.

For example, to create a ten-partition logical volume called lv06 in the rootvg in partitions 1 through 3, 41 through 45, and 50 through 60 of hdisk1, you could use the following procedure from the command line.

1. Enter the command:
   ```
   lspv -p hdisk1
   ```
   to verify that the physical partitions you plan to use are free to be allocated.
2. Create a file, such as `/tmp/mymap1`, containing:
   ```
   hdisk1:1-3
   hdisk1:41-45
   hdisk1:50-60
   ```
   The **mklv** command allocates the physical partitions in the order that they appear in the map file. Be sure that there are sufficient physical partitions in the map file to allocate the entire logical volume that you specify with the **mklv** command. (You can list more than you need.)
3. Type the command:
   ```
   mklv -t jfs -y lv06 -m /tmp/mymap1 rootvg 10
   ```

Developing a Striped Logical Volume Strategy

Striped logical volumes are used for large sequential file systems that are frequently accessed and performance-sensitive. Striping is intended to improve performance.

NOTE:
1. You can import an AIX 3.2 created volume group into an AIX 4.1 system, and you can import an AIX 4.1 volume group into an AIX 3.2. system, provided striping has not been applied. Once striping is put onto a disk, its importation into AIX 3.2 is prevented. The current implementation of **mksysb** does not restore any striped logical volume after the **mksysb** image is restored.
2. A volume group with a mirrored striped logical volume cannot be imported into a version older than AIX 4.3.3.
3. A dump space or boot logical volume is not striped.

To create a 12-partition striped logical volume called 1v07 in VGName with a stripe size of 16 KB across hdisk1, hdisk2, and hdisk3, you would enter the following:
```
mklv -y lv07 -S 16K VGName 12 hdisk1 hdisk2
hdisk3
```

To create a 12-partition striped logical volume called 1v08 in VGName with a stripe size of 8 KB across any three disks within VGName, you would enter the following:
```
mklv -y lv08 -S 8K -u 3 VGName 12
```

For more information on how to improve performance by using disk striping, see *AIX 5L Version 5.1 Performance Management Guide*.

Determining a Write-Verify Policy

Using the write-verify option causes all Write operations to be verified by an immediate follow-up Read operation to check the success of the Write. If the Write operation is not successful, you get an error message. This policy enhances availability but degrades performance because of the extra time needed for the Read. You can specify the use of a write-verify policy on a logical volume either when you create it using the **mklv** command or later by changing it with the **chlv** command.

Implementing Volume Group Policies

1. Use the **lspv** command to check your allocated and free physical volumes. In a standard configuration, the more disks that make up a quorum volume group the better the chance of the quorum remaining when a disk failure occurs. In a nonquorum group, a minimum of two disks must make up the volume group.
2. To ensure a quorum, add one or more physical volumes. See Add fixed disk without data to existing volume group, or Add fixed disk without data to new volume groupfor more information. To change a volume group to nonquorum status, see Change a User-Defined Volume Group to Nonquorum Status.
3. The standard configuration provides a single volume group that includes multiple physical volumes attached to the same disk adapter and other supporting hardware. Reconfiguring the hardware is an elaborate step. Separate hardware is only necessary if your site requires high availability.

Logical Volume Manager Limitation Warnings

- In the design of Logical Volume Manager (LVM), each logical partition maps to one physical partition (PP). And, each physical partition maps to a number of disk sectors. The design of LVM limits the number of physical partitions that LVM can track per disk to 1016. In most cases, not all of the 1016 tracking partitions are used by a disk. The default size of each physical partition during a **mkvg** command is 4 MB, which implies that individual disks up to 4 GB can be included into a volume group.

  If a disk larger than 4 GB is added to a volume group (based on usage of the default 4 MB size for the physical partition), the disk addition fails. The warning message provided is:

  ```
  The Physical Partition Size of <number A> requires the creation of <number B>:
  partitions for hdiskX.  The system limitation is <number C> physical partitions
  per disk at a factor value of <number D>. Specify a larger Physical Partition
  Size or a larger factor value in order create a volume group on this disk.
  ```

  There are two instances where this limitation is enforced:

1. The user tries to use the **mkvg** command to create a volume group and the number of physical partitions on a disk in the volume group exceeds 1016.

   The workaround to this limitation is to select from the physical partition size ranges of:

   ```
   1, 2, (4), 8, 16, 32, 64, 128, 256, 512, 1024
   ```

   Megabytes and use the **mkvg -s**

   Use a suitable factor (**mkvg -t** command) that allows multiples of 1016 partitions per disk.
2. The disk that violates the 1016 limitation attempts to join a pre-existing volume group with the **extendvg** command. The user can convert the existing volume group to hold multiples of 1016 paritions per disk using the **-t** option of the **chvg** command. The value of the **-t** option (factor) with the **chvg** command can be chosen in such a way that the new disk can be accommodated within the new limit of (1016 * factor). The user can also recreate the volume group with a larger partition size allowing the new disk to work or create a standalone volume group consisting of a larger physical size for the new disk.

   If the install code detects that the rootvg drive is larger than 4 GB, it changes the **mkvg -s** value until the entire disk capacity can be mapped to the available 1016 tracks. This install change also implies that all other disks added to rootvg, regardless of size, are also defined at that physical partition size.

   For systems using a redundant array of identical discs (RAID) , the **/dev/hdiskX** name used by LVM may consist of many non-4 GB disks. In this case, the 1016 requirement still exists. LVM is unaware of the size of the individual disks that really make up **/dev/hdiskX**. LVM bases the 1016 limitation on the recognized size of **/dev/hdiskX**, and not the real physical disks that make up **/dev/hdiskX**.

Limitations:

1016 VGSA Regulations	The 1016 VGSA is used to track the staleness of mirrors. The staleness of mirrors indicates that one copy of the data does not look like the other two copies. If you are in volation of 1016, you might get a false report of a non-mirrored logical volume being stale, or you might get a false indication that one of your mirror copies has gone stale. Also, the **migratepv** command might fail because **migratepv** briefly uses mirroring to move a logical volume from one disk to another. If the target logical partition is incorrectly considered stale, the **migratepv** cannot remove the source logical partition, and the command fails in the middle of migration. The **reorgvg** is another command that performs its actions by using temporary mirroring.
Mirroring or **migratepv**	If you do not use mirroring or the **migratepv** command, your data is still safe as the day before you found out about 1016 violations. The data can be lost only if you are mirroring a logical volume, and: • All copies go bad at the same time, and • LVM is not aware of it because copies that go bad are beyond the 1016 tracking range. In this case, you still lose data if you were within the 1016 range. If you do not use mirror or the **migratepv** command, this issue would not be a problem.
Move Volume Group	A volume group can be moved between systems and versions of this operating system. The enforcement of this 1016 limit is only during using the **mkvg** and **extendvg** commands. The data is safe on all versions of this operating system.
Change the PP Size Limitation	The PP size limitation is not changed. The ability to change the PP size assures that regardless of the size of your disk drive, you will be able to accommodate the drive below 1016 limit. Also, if a change to the limitation is made, then a huge incompatibility occurs in the LVM.
Rebuild the Volume Group	The later versions of this operating system mentioned in this document only prevent the future creation of disks in volume groups that violate the 1016 limitation. Disks that already violate the 1016 limit must be recreated at larger PP sizes.

- In some instances, the user experiences a problem adding a new disk to an existing volume group or in creating of a new volume group. The warning message provided by LVM is:

  ```
  Not enough descriptor area space left in this
  volume group.
  Either try adding a smaller PV or use another volume group.
  ```

  On every disk in a volume group, there exists an area called the volume group descriptor area (VGDA). This space allows the user to take a volume group to another system using the **importvg** command. The VGDA contains the names of disks that make up the volume group, their physical sizes, partition mapping, logical volumes that exist in the volume group, and other pertinent LVM management information.

  When the user creates a volume group, the **mkvg** command defaults to allowing the new volume group to have a maximum of 32 disks in a volume group. However, as bigger disks have become more prevalent, this 32-disk limit is usually not achieved because the space in the VGDA is used up faster, as it accounts for the capacity on the bigger disks. This maximum VGDA space, for 32 disks, is a fixed size which is part of the LVM design. Large disks require more management-mapping space in the VGDA, causing the number and size of available disks to be added to the existing volume group to shrink.

When a disk is added to a volume group, not only does the new disk get a copy of the updated VGDA, but all existing drives in the volume group must be able to accept the new, updated VGDA.

The exception to this description of the maximum VGDA is the **rootvg** command. To provide users more free disk space, when rootvg is created, the **mkvg** command does not use the maximum limit of 32 disks that is allowed into a volume group. Instead, the number of disks picked in the install menu of is used as the reference number by the **mkvg -d** command during the creation of rootvg. This **-d** number is `7` for one disk and one more for each additional disk picked. For example, if two disks are picked, the number is `8` and if three disks are picked, the number is `9`, and so on. This limit does not prohibit the user from adding more disks to rootvg during post-installation. The amount of free space left in a VGDA, and the number size of the disks added to a volume group, depends on the size and number of disks already defined for a volume group.

If the customer requires more VGDA space in the rootvg, then they should use the **mksysb**, and **migratepv** commands to reconstruct and reorganize their rootvg (the only way to change the **-d** limitation is recreation of a volume group).

Note: Do not place user data onto rootvg disks. This separation provides an extra degree of system integrity.

- The logical volume control block (LVCB) is the first 512 bytes of a logical volume. This area holds important information such as the creation date of the logical volume, information about mirrored copies, and possible mount points in the journaled file system (JFS). Certain LVM commands are required to update the LVCB, as part of the algorithms in LVM. The old LVCB is read and analyzed to see if it is a valid. If the information is valid LVCB information, the LVCB is updated. If the information is not valid, the LVCB update is not performed and the user is given the warning message:

```
Warning, cannot write lv control block
data.
```

  Most of the time, this is a result of database programs accessing raw logical volumes (and bypassing the JFS) as storage media. When this occurs, the information for the database is literally written over the LVCB. Although this might seem fatal, it is not the case. After the LVCB is overwritten, the user can still:
  – Expand a logical volume
  – Create mirrored copies of the logical volume
  – Remove the logical volume
  – Create a journaled filesystem to mount the logical volume.

  There are limitations to deleting LVCBs. The logical volumes with deleted LVCB face possible, incomplete importation into other systems. During an importation, the LVM **importvg** command will scan the LVCBs of all defined logical volumes in a volume group for information concerning the logical volumes. If the LVCB is deleted, the imported volume group still defines the logical volume to the new system that is accessing this volume group, and the user can still access the raw logical volume. However, any journaled file system information is lost and the associated mount point is not imported into the new system. The user must create new mount points and the availability of previous data stored in the file system is not assured.

  Also, during an import of a logical volume with an erased LVCB, some non-jfs information concerning the logical volume (displayed by the **lslv** command) cannot be found. When this occurs, the system uses default logical volume information to populate the ODM information. Thus, some output from the **lslv** command might be inconsistent with the real logical volume. If any logical volume copies still exist on the original disks, the information is not be correctly reflected in the ODM database. Use the **rmlvcopy** and **mklvcopy** commands to rebuild any logical volume copies and synchronize the ODM.

Chapter 7. File Systems

This chapter explains file systems, including information about directories, disk space, access control, mounted file systems and directories, and file system recovery. Topics covered are:

File Systems Overview

A *file system* is a hierarchical structure (file tree) of files and directories. This type of structure resembles an inverted tree with the roots at the top and branches at the bottom. This file tree uses directories to organize data and programs into groups, allowing the management of several directories and files at one time.

Some tasks are performed more efficiently on a file system than on each directory within the file system. For example, you can back up, move, or secure an entire file system.

A file system resides on a single logical volume. The **mkfs** (make file system) command or the System Management Interface Tool (**smit** command) creates a file system on a logical volume. Every file and directory belongs to a file system within a logical volume.

To be accessible, a file system must be mounted onto a directory mount point. When multiple file systems are mounted, a directory structure is created that presents the image of a single file system. It is a hierarchical structure with a single root. This structure includes the base file systems and any file systems you create.

You can access both local and remote file systems using the **mount** command. This makes the file system available for read and write access from your system. Mounting or unmounting a file system usually requires system group membership. File systems can be mounted automatically, if they are defined in the **/etc/filesystems** file. You can unmount a local or remote file system with the **umount** command, unless a user or process is accessing that file system.

For information on the structure of the file system, see “Understanding the File Tree” on page 70.

File System Types

Multiple file system types are supported. These include the following:

Journaled File System

> **Note:** Journaled file system (JFS) is native to the POWER-based platform and is not available on the Itanium-based platform.

The native file system type is called the *journaled file system* (JFS). It supports the entire set of file system semantics. This file system uses database journaling techniques to maintain its structural consistency. This prevents damage to the file system when the system is halted abnormally.

Each journaled file system resides on a separate logical volume. The operating system mounts journaled file systems during initialization. This multiple file system configuration is useful for system management functions such as backup, restore, and repair, because it isolates a part of the file tree so that you can work on it.

Enhanced Journaled File System

> **Note:** Enhanced journaled file system (JFS2) is native to Itanium-based platform. Although JFS2 is not native to the POWER-based platform, it is available.

Enhanced journaled file system (JFS2) supports the entire set of file system semantics. The file system uses database journaling techniques to maintain its structural consistency. This prevents damage to the file system when the file system is halted abnormally.

Each JFS2 resides on a separate logical volume. The operating system mounts JFS2 during initialization. This multiple file system configuration is useful for system management functions such as backup, restore, and repair. It isolates a part of the file tree to allow system administrators to work on a particular part of the file tree.

Network File System

The *network file system* (NFS) is a distributed file system that allows users to access files and directories located on remote computers and use those files and directories as if they were local. For example, users can use operating system commands to create, remove, read, write, and set file attributes for remote files and directories.

CD-ROM File System

The *CD-ROM file system* (CDRFS) is a file system type that allows you to access the contents of a CD-ROM through the normal file system interfaces. It is a read-only local file system implementation under the logical file system (LFS) layer supporting the following volume and file structure formats:

The ISO 9660:1988(E) standard:	The CDRFS supports ISO 9660 level 3 of interchange and level 1 of implementation.
The High Sierra Group Specification:	Precedes the ISO 9660 and provides backward compatibility with previous CD-ROMs.
The Rock Ridge Group Protocol:	Specifies extensions to the ISO 9660 that are fully compliant with the ISO 9660 standard, and that provide full POSIX file system semantics based on the System Use Sharing Protocol (SUSP) and the Rock Ridge Interchange Protocol (RRIP), enabling mount/access CD-ROM as with any other UNIX file system.
The CD-ROM eXtended Architecture File Format (in Mode 2 Form 1 sector format only)	The CD-ROM eXtended Architecture (XA) file format specifies extensions to the ISO 9660 that are used in CD-ROM-based multimedia applications for example, Photo CD.

For all volume and file structure formats, the following restrictions apply:

- Single-volume volume set only
- Non-interleaved files only

The CDRFS is dependent upon the underlying CD-ROM device driver to provide transparency of the physical sector format (CD-ROM Mode 1 and CD-ROM XA Mode 2 Form 1), and the multisession format of the disks (mapping the volume descriptor set from the volume recognition area of the last session).

File System Commands

There are a number of commands designed to operate on file systems, regardless of type. The **/etc/filesystems** file controls the list of file systems that the following commands can manipulate:

chfs	Changes the characteristics of a file system
crfs	Adds a file system
lsfs	Displays the characteristics of a file system
rmfs	Removes a file system
mount	Makes a file system available for use

Four commands operate on virtual file systems types. The **/etc/vfs** file contains the information on the file system types that the following commands manipulate:

chvfs	Changes the characteristics of a file system type
crvfs	Adds a new file system type
lsvfs	Lists the characteristics of a file system type
rmvfs	Removes a file system type

File System Management Tasks

A file system is a complete directory structure, including a root directory and any subdirectories and files beneath it. File systems are confined to a single logical volume. Some of the most important system management tasks are concerning file systems, specifically:

- Allocating space for file systems on logical volumes
- Creating file systems
- Making file system space available to system users
- Monitoring file system space usage
- Backing up file systems to guard against data loss in the event of system failures
- Maintaining file systems in a consistent state.

Following is a list of system management commands that are used regularly for working with file systems:

backup	Performs a full or incremental backup of a file system
dd	Copies data directly from one device to another for making file system backups
df	Reports the amount of space used and free on a file system
fsck	Checks file systems and repairs inconsistencies
mkfs	Makes a file system of a specified size on a specified logical volume
mount	Attaches a file system to the systemwide naming structure so that files and directories in that file system can be accessed
restore	Restores files from a backup
umount	Removes a file system from the systemwide naming structure, making the files and directories in the file system inaccessible.

Understanding the File Tree

The file tree organizes files into directories containing similar information. This organization facilitates remote mounting of directories and files. System administrators can use these directories as building blocks to construct a unique file tree for each client mounting individual directories from one or more servers. Mounting files and directories remotely, rather than keeping all information local, has the following advantages:

- Conserves disk space
- Allows easy, centralized system administration.
- Provides a more secure environment.

The file tree has the following characteristics:

- Files that can be shared by machines of the same hardware architecture are located in the **/usr** file system.
- Variable per-client files, such as spool and mail files, are located in the **/var** file system.
- Architecture-independent, shareable text files, such as manual pages, are located in the **/usr/share** directory.
- The **/** (root) file system contains files and directories critical for system operation. For example, it contains a device directory, programs used for system startup, and mount points where file systems can be mounted onto the root file system.
- The **/home** file system is the mount point for user home directories.
- For servers, the **/export** directory contains paging-space files, per-client (unshared) root file systems, dump, home, and **/usr/share** directories for diskless clients, as well as exported **/usr** directories.

For information about the contents of a specific file system or directory, see the following:

- “Understanding the Root File System”
- “Understanding the /usr File System” on page 72
- “Understanding the /usr/share Directory” on page 74
- “Understanding the /var File System” on page 75
- “Understanding the /export Directory” on page 75.

Understanding the Root File System

The following diagram shows many of the subdirectories of the root file system.

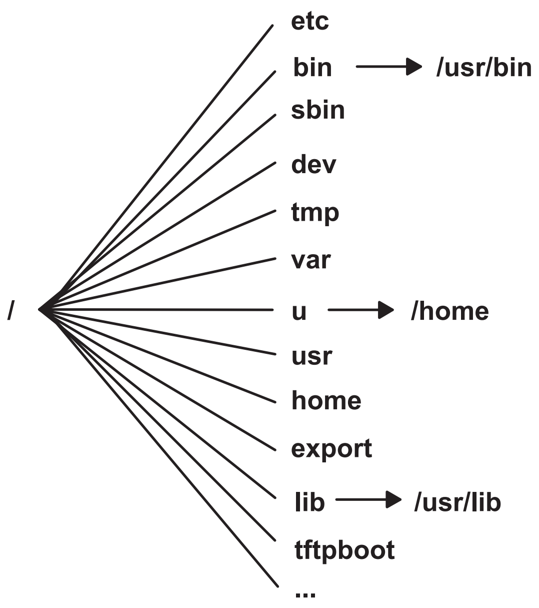

Figure 9. Root File System. . This diagram shows the root file system and its subdirectories. The **/bin** *subdirectory points to the* **/usr/bin** *directory. The* **/lib** *subdirectory points to the* **/usr/lib** *directory. The* **/u** *subdirectory points to the* **/home** *directory.*

The root file system is the top of the hierarchical file tree. It contains the files and directories critical for system operation, including the device directory and programs for booting the system. The root file system also contains mount points where file systems can be mounted to connect to the root file system hierarchy.

The following list provides information about the contents of some of the subdirectories of the **/** (root) file system.

/etc	Contains configuration files that vary for each machine. Examples include: • **/etc/hosts** • **/etc/passwd** The **/etc** directory contains the files generally used in system administration. Most of the commands that previously resided in the **/etc** directory now reside in the **/usr/sbin** directory. However, for compatibility, the **/usr/sbin** directory contains symbolic links to the locations of some executable files. Examples include: • **/etc/chown** is a symbolic link to **/usr/bin/chown**. • **/etc/exportvg** is a symbolic link to **/usr/sbin/exportvg**.
/bin	Symbolic link to the **/usr/bin** directory. In prior UNIX file systems, the **/bin** directory contained user commands that now reside in the **/usr/bin** directory.
/sbin	Contains files needed to boot the machine and mount the **/usr** file system. Most of the commands used during booting come from the boot image's RAM disk file system; therefore, very few commands reside in the **/sbin** directory.
/dev	Contains device nodes for special files for local devices. The **/dev** directory contains special files for tape drives, printers, disk partitions, and terminals.
/tmp	Serves as a mount point for a file system that contains system-generated temporary files. The **/tmp** file system is an empty directory.
/var	Serves as a mount point for files that vary on each machine. The **/var** file system is configured as a file system since the files it contains tend to grow. See "Understanding the /var File System" on page 75 for more information.
/u	Symbolic link to the **/home** directory.

/usr	Contains files that do not change and can be shared by machines such as executables and ASCII documentation. Standalone machines mount the root of a separate local file system over the **/usr** directory. Diskless machines and machines with limited disc resources diskmount a directory from a remote server over the **/usr** file system. See "Understanding the /usr File System" for more information about the file tree mounted over the **/usr** directory.
/home	Serves as a mount point for a file system containing user home directories. The **/home** file system contains per-user files and directories. In a standalone machine, the **/home** directory is contained in a separate file system whose root is mounted over the **/home** directory root file system. In a network, a server might contain user files that are accessible from several machines. In this case, the server copy of the **/home** directory is remotely mounted onto a local **/home** file system.
/export	Contains the directories and files on a server that are for remote clients. See "Understanding the /export Directory" on page 75 for more information about the file tree that resides under the **/export** directory.
/lib	Symbolic link to the **/usr/lib** directory. See "Understanding the /usr File System" for more information.
/tftpboot	Contains boot images and boot information for diskless clients.

Understanding the /usr File System

The **/usr** file system contains executable files that can be shared among machines. The major subdirectories of the /**usr** directory are shown in the following diagram.

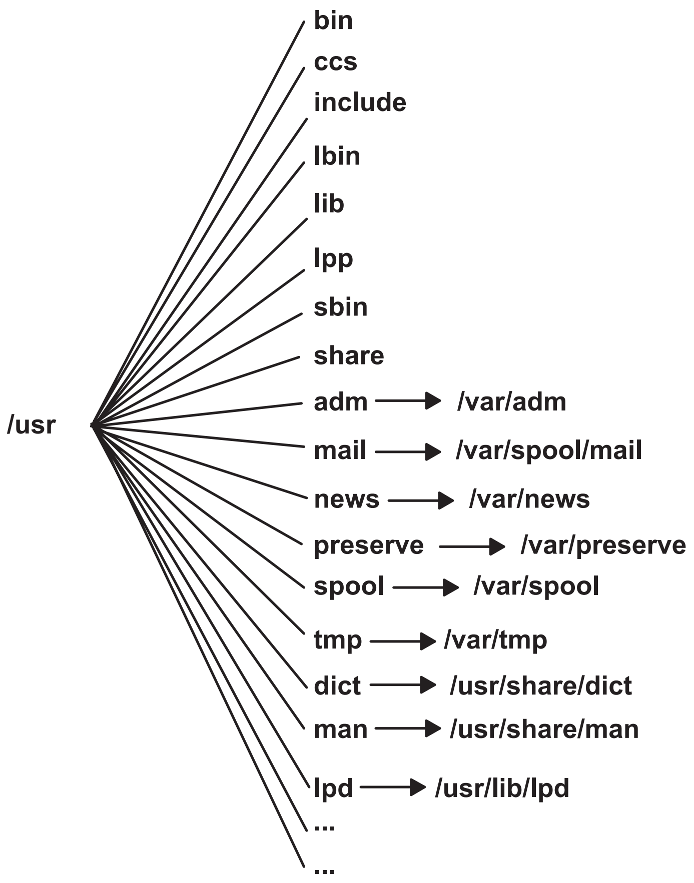

Figure 10. /usr File System. This diagram shows the major subdirectories of the /usr directory, which includes: **/bin**, **ccs**, **/lib**, **/lpp**, **/adm** *and its* **/var/adm** *subdirectory,* **/mail** *and its* **/var/spool/mail** *subdirectory, and* **/man** *and its* **/usr/share/man** *subdirectory.*

On a standalone machine the **/usr** file system is a separate file system (in the **/dev/hd2** logical volume). On a diskless machine or a machine with limited disk resources, a directory from a remote server is mounted with read-only permissions over the local **/usr** file system. The **/usr** file system contains read-only commands, libraries, and data.

Except for the contents of the **/usr/share** directory, the files and directories in the **/usr** file system can be shared by all machines of the same hardware architecture.

The **/usr** file system includes the following directories:

/usr/bin	Contains ordinary commands and shell scripts. For example, the **/usr/bin** directory contains the **ls**, **cat**, and **mkdir** commands.
/usr/ccs	Contains unbundled development package binaries.
/usr/include	Contains include, or header, files.
/usr/lbin	Contains executable files that are backends to commands.
/usr/lib	Contains architecture-independent libraries with names of the form **lib*.a**. The **/lib** directory in **/** (root) is a symbolic link to the **/usr/lib** directory, so all files that were once in the **/lib** directory are now in the **/usr/lib** directory. This includes a few nonlibrary files for compatibility.
/usr/lpp	Contains optionally installed products.
/usr/sbin	Contains utilities used in system administration, including System Management Interface Tool (SMIT) commands. Most of the commands that once resided in the **/etc** directory now reside in the **/usr/sbin** directory.
/usr/share	Contains files that can be shared among machines with different architectures. See "Understanding the /usr/share Directory" on page 74 for more information.

Symbolic Links to the /var Directory

/usr/adm	Symbolic link to the **/var/adm** directory
/usr/mail	Symbolic link to the **/var/spool/mail** directory
/usr/news	Symbolic link to the **/var/news** directory
/usr/preserve	Symbolic link to the **/var/preserve** directory
/usr/spool	Symbolic link to the **/var/spool** directory
/usr/tmp	Symbolic link to the **/var/tmp** directory, because the **/usr** directory is potentially shared by many nodes and is read-only.

Symbolic Links to the /usr/share and /usr/lib Directory

/usr/dict	Symbolic link to the **/usr/share/dict** directory
/usr/man	Symbolic link to the **/usr/share/man** directory
/usr/lpd	Symbolic link to the **/usr/lib/lpd** directory.

Understanding the /usr/share Directory

The **/usr/share** directory contains architecture-independent shareable text files. The contents of this directory can be shared by all machines, regardless of hardware architecture.

In a mixed architecture environment, the typical diskless client mounts one server directory over its own **/usr** directory and then mounts a different directory over the **/usr/share** directory. The files below the **/usr/share** directory are contained in one or more separately installable packages. Thus, a node might have the other parts of the **/usr** directory it depends on locally installed while using a server to provide the **/usr/share** directory.

Some of the files in the **/usr/share** directory include the directories and files shown in the following diagram.

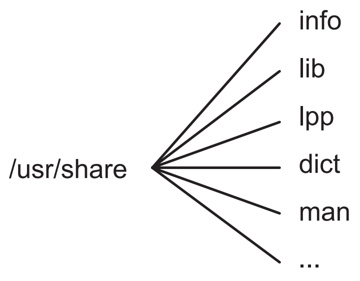

Figure 11. /usr/share Directory. . This diagram shows several directories under the **/usr/share** *directory, including* **/info***,* **/lib***,* **/lpp***,* **/dict***, and* **/man***.*

The **/usr/share** directory includes the following:

/usr/share/man	Contains the manual pages if they have been loaded
/usr/share/dict	Contains the spelling dictionary and its indexes
/usr/share/info	Contains the InfoExplorer database files
/usr/share/lib	Contains architecture-independent data files, including **terminfo**, **learn**, **tmac**, **me**, and **macros**
/usr/share/lpp	Contains data and information about optionally installable products on the system.

Understanding the /var File System

> **Attention:** The **/var** file system tends to grow because it contains subdirectories and data files that are used by busy applications such as accounting, mail, and the print spooler. If applications on your system use the **/var** file system extensively, increase the file system size beyond the 4MB **/var** default.

Specific **/var** files that warrant periodic monitoring are **/var/adm/wtmp** and **/var/adm/ras/errlog**.

Other **/var** files to monitor are:

/var/adm/ras/trcfile	If the trace facility is turned on
/var/tmp/snmpd.log	If the **snmpd** command is running on your system.

The /var directory diagram shows some of the directories in the **/var** file system.

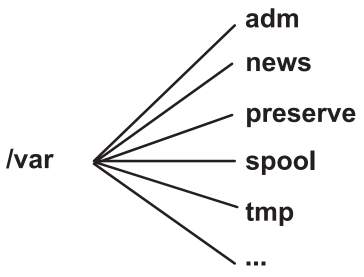

Figure 12. /var Directory. . This diagram shows the major subdirectories of the /var directory, including ***/adm****,* ***/news****,* ***/preserve****,* ***/spool****,* ***/tmp****, and* ***/lpp****.*

/var/adm	Contains system logging and accounting files
/var/news	Contains system news
/var/preserve	Contains preserved data from interrupted edit sessions; similar to the **/usr/preserve** directory in previous releases
/var/spool	Contains files being processed by programs such as electronic mail; similar to the **/usr/spool** directory in previous releases
/var/tmp	Contains temporary files; similar to the **/usr/tmp** directory in previous releases. The **/usr/tmp** directory is now a symbolic link to **/var/temp**.

Understanding the /export Directory

The **/export** directory contains server files exported to clients, such as diskless, dataless, or disk-poor machines. A server can export several types of disk space, including packages of executable programs, paging space for diskless clients, and root file systems for diskless clients or those with low disk resources. The standard location for such disk space in the file tree is the **/export** directory. Some of the subdirectories of the **/export** directory are shown in the following list:

/exec	Contains directories that diskless clients mount over their **/user** file systems.
/swap	Contains files for diskless clients' remote paging.
/share	Contains directories that diskless clients mount over their **/user/share** directory.
/root	Contains directories that diskless clients mount over their **/ (root)** file system.
/dump	Contains directories for diskless clients' remote dump files.

/home Contains directories that diskless clients mount over their **/home** file system.

The **/export** directory is the default location for client resources for the diskless commands. The **/export** directory is only the location of client resources on the server. Because clients mount these resources onto their own file tree, these resources appear to clients at the normal places in a file tree. The major subdirectories of the **/export** directory, and their corresponding mount points on a client file tree, include:

/export/root
This directory is mounted over the client root (/) file system. Client root directories are located in the **/export/root** directory by default and are named with the client's host name.

/export/exec
Also called the Shared Product Object Tree (SPOT) directory. This directory is mounted over the client **/usr** file system. SPOTs are versions of the **/usr** file system stored in the **/export/exec** directory and have names that reflect their release level. By default, the name is **RISCAIX**.

/export/share
This directory is mounted over the client **/usr/share** directory. This directory contains data that can be shared by many architectures. The default location is **/export/share/AIX/usr/share**.

/export/home
This directory is mounted over the client **/home** file system. It contains user directories grouped by client host names. The default location for client home directories is **/export/home**.

/export/swap
Also called the paging directory. In standalone or dataless systems, paging is provided by a local disk; for diskless clients, this service is provided by a file on a server. This file is named after the client's host name and by default is found in the **/export/swap** directory.

/export/dump
Standalone systems use a local disk as the dump device; diskless clients use a file on a server. The file resides in a directory named after the client host name and by default is found in the **/export/dump** directory.

microcode
This directory contains microcode for physical devices. The default location is **/export/exec/RISCAIX/usr/lib/microcode**.

Understanding Data Compression

Note: Data Compression is specific to JFS.

The journaled file system (JFS) supports fragmented and compressed file systems. Both types of file systems save disk space by allowing a logical block to be stored on the disk in units or ″fragments″ smaller than the full block size of 4096 bytes. In a fragmented file system, only the last logical block of files no larger than 32KB are stored in this manner, so that fragment support is only beneficial for file systems containing numerous small files. Data compression, however, allows all logical blocks of any-sized file to be stored as one or more contiguous fragments. On average, data compression saves disk space by about a factor of two.

The use of fragments and data compression does, however, increase the potential for fragmentation of the disk free space. Fragments allocated to a logical block must be contiguous on the disk. A file system experiencing free space fragmentation might have difficulty locating enough contiguous fragments for a logical block allocation, even though the total number of free fragments may exceed the logical block requirements. The JFS alleviates free space fragmentation by providing the **defragfs** program which ″defragments″ a file system by increasing the amount of contiguous free space. This utility can be used for fragmented and compressed file systems. The disk space savings gained from fragments and data compression can be substantial, while the problem of free space fragmentation remains manageable.

Data compression in the current JFS is compatible with previous versions of this operating system. The application programming interface (API) comprised of all the system calls remains the same in both versions of the JFS.

For more information on fragment support, disk utilization, free space fragmentation, and the performance costs associated with fragments, refer to “Understanding Fragments and a Variable Number of I-Nodes” on page 79.

Data Compression Implementation

Note: Data Compression is specific to JFS.

Attention: The root file system (**/**) must not be compressed. Compression **/usr** file system is not recommended because **installp** must be able to do accurate size calculations for updates and new installs. See the “Implicit Behavior” section for more information on size and calculations.

Data compression is an attribute of a file system which is specified when the file system is created with the **crfs** or **mkfs** command. Compression only applies to regular files and long symbolic links in such file systems. Fragment support continues to apply to directories and metadata that are not compressed. Each logical block of a file is compressed by itself before being written to the disk. Compression in this manner facilitates random seeks and updates, while losing only a small amount of freed disk space in comparison to compressing data in larger units.

After compression, a logical block usually requires less than 4096 bytes of disk space. The compressed logical block is written to the disk and allocated only the number of contiguous fragments required for its storage. If a logical block does not compress, then it is written to disk in its uncompressed form and allocated 4096 bytes of contiguous fragments.

Implicit Behavior

Note: Information in this section is specific to JFS.

Because a program that writes a file does not expect an out-of-space (ENOSPC) condition to occur after a successful write (or successful store for mapped files), it is necessary to guarantee that space be available when logical blocks are written to the disk. This is accomplished by allocating 4096 bytes to a logical block when it is first modified so that there is disk space available even if the block does not compress. If a 4096-byte allocation is not available, the system returns an ENOSPC or EDQUOT error condition even though there might be enough disk space to accommodate the compressed logical block. Premature reporting of an out-of-space condition is most likely when operating near disk quota limits or with a nearly full file system.

In addition to incurring a premature out-of-space error, compressed file systems may exhibit the following behavior:

- Because 4096 bytes are initially allocated to a logical block, certain system calls might receive an ENOSPC or EDQUOT error. For example, an old file might be mapped using the **mmap** system call, and a store operation into a previously written location can result in an ENOSPC error. The **ftruncate** system call might also incur an ENOSPC or EDQUOT error if the truncation is not to a logical block boundary.
- With data compression, a full disk block remains allocated to a modified block until it is written to disk. If the block had a previously committed allocation of less than a full block, the amount of disk space tied up by the block is the sum of the two, the previous allocation not being freed until the file (i-node) is committed. This is the case for normal fragments. The number of logical blocks in a file that can have previously committed allocations is at most one for normal fragments, but can be as many as the number of blocks in a file with compression.

- None of the previously committed resources for a logical block are freed until the **fsync** or **sync** system call is run by the application program.
- The **stat** system call indicates the number of fragments allocated to a file. The number reported is based on 4096 bytes being allocated to modified but unwritten blocks and the compressed size of unmodified blocks. Previously committed resources are not counted by the **stat** system call. The **stat** system call reports the correct number of allocated fragments after an i-node commit operation if none of the modified blocks compressed. Similarly, disk quotas are charged for the current allocation. As the logical blocks of a file are written to the disk, the number of fragments allocated to them decrease if they compress, and thereby change disk quotas and the result from **stat**.

Specifying Compression

Note: Information in this section is specific to JFS.

The **crfs**, **mkfs**, and **lsfs** commands have been extended for data compression. These commands as well as the System Management Interface Tool (SMIT) now contain options for specifying or identifying data compression.

Identifying Data Compression

Note: Information in this section is specific to JFS.

The **-q** flag of the **lsfs** command displays the current value for compression.

Compatibility and Migration

Note: Information in this section is specific to JFS.

Previous versions of this operating system are compatible with the current JFS. Disk image compatibility is maintained with previous versions of this operating system, so that file systems can be mounted and accessed without requiring disk migration activities or losing file system performance.

Backup and Restore

Note: Information in this section is specific to JFS.

While backup and restore sequences can be performed from compressed to noncompressed file systems or between compressed file systems with different fragment sizes, because of the enhanced disk utilization of compressed file systems, restore operations might fail due to a shortage of disk space. This is of particular interest for full file system backup and restore sequences and might even occur when the total file system size of the target file system is larger than that of the source file system.

Compression Algorithm

Note: Information in this section is specific to JFS.

The compression algorithm is an IBM version of LZ. In general, LZ algorithms compress data by representing the second and later occurrences of a given string with a pointer that identifies the location of the first occurrence of the string and its length. At the beginning of the compression process, no strings have been identified, so at least the first byte of data must be represented as a ″raw″ character requiring 9-bits (0,byte). After a given amount of data is compressed, say *N* bytes, the compressor searches for the longest string in the *N* bytes that matches the string starting at the next unprocessed byte. If the longest match has length 0 or 1, the next byte is encoded as a ″raw″ character. Otherwise, the string is represented as a (pointer,length) pair and the number of bytes processed is incremented by length.

Architecturally, IBM LZ supports values of *N* of 512, 1024, or 2048. IBM LZ specifies the encoding of (pointer,length) pairs and of raw characters. The pointer is a fixed-length field of size log2 *N*, while the length is encoded as a variable-length field.

Performance Costs

Note: Information in this section is specific to JFS.

Because data compression is an extension of fragment support, the performance associated with fragments also applies to data compression. Compressed file systems also affect performance in the following ways:

- It can require a great deal of time to compress and decompress data so that the usability of a compressed file system might be limited for some user environments.
- Most UNIX regular files are written only once, but some are updated in place. For the latter, data compression has the additional performance cost of having to allocate 4096 bytes of disk space when a logical block is first modified, and then reallocate disk space after the logical block is written to the disk. This additional allocation activity is not necessary for regular files in a noncompressed file system.
- Data compression increases the number of processor cycles. For the software compressor, the number of cycles for compression is approximately 50 cycles per byte, and for decompression 10 cycles per byte.

Understanding Fragments and a Variable Number of I-Nodes

Note: This article is specific to JFS. For related information pertaining to JFS2, see “Enhanced Journaled File System” on page 85.

The journaled file system (JFS) fragment support allows disk space to be divided into allocation units that are smaller than the default size of 4096 bytes. Smaller allocation units or ″fragments″ minimize wasted disk space by more efficiently storing the data in a file or directory partial logical blocks. The functional behavior of JFS fragment support is based on that provided by Berkeley Software Distribution (BSD) fragment support. Similar to BSD, JFS fragment support allows users to specify the number of i-nodes that a file system has.

Disk Utilization

Many UNIX file systems only allocate contiguous disk space in units equal in size to the logical blocks used for the logical division of files and directories. These allocation units are typically referred to as ″disk blocks″ and a single disk block is used exclusively to store the data contained within a single logical block of a file or directory.

Using a relatively large logical block size (4096 bytes for example) and maintaining disk block allocations that are equal in size to the logical block are advantageous for reducing the number of disk I/O operations that must be performed by a single file system operation. A file or directory data is stored on disk in a small number of large disk blocks rather than in a large number of small disk blocks. For example, a file with a size of 4096 bytes or less is allocated a single 4096-byte disk block if the logical block size is 4096 bytes. A read or write operation therefore only has to perform a single disk I/O operation to access the data on the disk. If the logical block size is smaller requiring more than one allocation for the same amount of data, then more than one disk I/O operation is be required to access the data. A large logical block and equal disk block size are also advantageous for reducing the amount of disk space allocation activity that must be performed as new data is added to files and directories, because large disk blocks hold more data.

Restricting the disk space allocation unit to the logical block size can, however, lead to wasted disk space in a file system containing numerous files and directories of a small size. Wasted disk space occurs when a logical block worth of disk space is allocated to a partial logical block of a file or directory. Because partial logical blocks always contain less than a logical block worth of data, a partial logical block only consumes a portion of the disk space allocated to it. The remaining portion remains unused because no other file or directory can write its contents to disk space that has already been allocated. The total amount of wasted disk space can be large for file systems containing a large number of small files and directories.

Optimizing Disk Utilization

In the JFS, the disk space allocation unit, referred to as a *fragment*, can be smaller than the logical block size of 4096 bytes. With the use of fragments smaller than 4096 bytes, the data contained within a partial logical block can be stored more efficiently by using only as many fragments as are required to hold the data. For example, a partial logical block that only has 500 bytes could be allocated a fragment of 512 bytes (assuming a fragment size of 512 bytes), thus greatly reducing the amount of wasted disk space. If the storage requirements of a partial logical block increase, one or more additional fragments are allocated.

Fragments

The fragment size for a file system is specified during its creation. The allowable fragment sizes for journaled file systems (JFS) are 512, 1024, 2048, and 4096 bytes. Different file systems can have different fragment sizes, but only one fragment size can be used within a single file system. Different fragment sizes can also coexist on a single system (machine) so that users can select a fragment size most appropriate for each file system.

JFS fragment support provides a view of the file system as a contiguous series of fragments rather than as a contiguous series of disk blocks. To maintain the efficiency of disk operations, however, disk space is often allocated in units of 4096 bytes so that the disk blocks or allocation units remain equal in size to the logical blocks. A disk-block allocation in this case can be viewed as an allocation of 4096 bytes of contiguous fragments.

Both operational overhead (additional disk seeks, data transfers, and allocation activity) and better utilization of disk space increase as the fragment size for a file system decreases. To maintain the optimum balance between increased overhead and increased usable disk space, the following factors apply to JFS fragment support:

- Where possible, disk space allocations of 4096 bytes of fragments are maintained for a file or the logical blocks of a directory .
- Only partial logical blocks for files or directories less than 32KB in size can be allocated less than 4096 bytes of fragments.

Maintaining 4096-byte disk space allocations allows disk operations to be more efficient as described previously in “Disk Utilization” on page 79.

As the files and directories within a file system grow beyond 32 KB in size, the benefit of maintaining disk space allocations of less than 4096 bytes for partial logical blocks diminishes. The disk space savings as a percentage of total file system space grows small while the extra performance cost of maintaining small disk space allocations remains constant. Since disk space allocations of less than 4096 bytes provide the most effective disk space utilization when used with small files and directories, the logical blocks of files and directories equal to or greater than 32 KB are always allocated 4096 bytes of fragments. Any partial logical block associated with such a large file or directory is also allocated 4096 bytes of fragments.

Variable Number of I-Nodes

Since fragment support optimizes disk space utilization, it increases the number of small files and directories that can be stored within a file system. However, disk space is only one of the file system resources required by files and directories: each file or directory also requires a disk i-node. The JFS allows the number of disk i-nodes created within a file system to be specified in case more or fewer than the default number of disk i-nodes is desired. The number of disk i-nodes can be specified at file system creation as the number of bytes per i-node (NBPI). For example, an NBPI value of 1024 causes a disk i-node to be created for every 1024 bytes of file system disk space. Another way to look at this is that a small NBPI value (512 for instance) results in a large number of i-nodes, while a large NBPI value (such as 16,384) results in a small number of i-nodes.

The set of allowable NBPI values vary according to the allocation group size (agsize). The default is 8 MB. In AIX 4.1, agsize is fixed at 8 MB. The allowable NBPI values are 512, 1024, 2048, 4096, 8192, and 16,384 with an agsize of 8 MB.

In AIX 4.2 or later, a larger agsize can be used. The allowable values for agsize are 8, 16, 32, and 64. The range of allowable NBPI values scales up as agsize increases. If the agsize is doubled to 16 MB, the range of NBPI values also double: 1024, 2048, 4096, 8193, 16384, and 32768.

Specifying Fragment Size and NBPI

Fragment size and the number-of-bytes-per-i-node (NBPI) value are specified during the file system's creation with the **crfs** and **mkfs** commands or by using the System Management Interface Tool (SMIT). The decision of fragment size and how many i-nodes to create for the file system is based on the projected number of files contained by the file system and their size.

Identifying Fragment Size and NBPI

The file-system-fragment size and the number-of-bytes-per-i-node (NBPI) value can be identified through the **lsfs** command or the System Management Interface Tool (SMIT). For application programs, use the **statfs** subroutine to identify the file system fragment size.

Compatibility and Migration

Previous versions of this operating system are compatible with the current JFS, although file systems with a nondefault fragment size, NBPI value, or allocation group size might require special attention if migrated to a previous version.

File System Images

The JFS fully supports JFS file system images created under previous versions of this operating system. These file system images and any JFS file system image created with the default fragment size and NBPI value of 4096 bytes, and default allocation group size (agsize) of 8 can be interchanged with the current and previous versions of this operating system without requiring any special migration activities.

JFS file system images created with a fragment size or NBPI value or agsize other than the default values might be incompatible with previous versions of this operating system. Specifically, only file system images less than or equal to 2 GB in size and created with the default parameters can be interchanged amongst AIX 4.1 and AIX 4.2. File system images created with fragment size of either 512, 1024, 2048, or 4096, and an NBPI value of either 512, 1024, 2048, 4096, 8192, or 16384, and a agsize of 8 MB can ge interchanges amongst AIX 4.1 and AIX 4.2. Finally, creating a file system with NBPI value greater than 16384 or with an agsize greater than 8 MB results in a JFS file system that is only recognized on AIX 4.2.

The following procedure must be used to migrate incompatible file systems from one version of this operating system to another:

1. Back up the file system by file name on the source system.
2. Create a file system on the destination system.
3. Restore the backed-up files on the destination system.

Backup and Restore

Backup and restore sequences can be performed between file systems with different block sizes, however because of increased disk utilization, restore operations can fail due to a lack of free blocks if the block size of the source file system is smaller than the block size of the target file system. This is of particular interest for full file system backup and restore sequences and can even occur when the total file system size of the target file system is larger than that of the source file system.

Device Driver Limitations

A device driver must provide disk block addressability that is the same or smaller than the file system fragment size. For example, if a JFS file system was made on a user supplied RAM disk device driver, the driver must allow 512 byte blocks to contain a file system that had 512 byte fragments. If the driver only allowed page level addressability, a JFS with a fragment size of 4096 bytes could only be used.

Performance Costs

Although file systems that use fragments smaller than 4096 bytes as their allocation unit require substantially less disk space than those using the default allocation unit of 4096 bytes, the use of smaller fragments might incur performance costs.

Increased Allocation Activity

Because disk space is allocated in smaller units for a file system with a fragment size other than 4096 bytes, allocation activity can occur more often when files or directories are repeatedly extended in size. For example, a write operation that extends the size of a zero-length file by 512 bytes results in the allocation of one fragment to the file, assuming a fragment size of 512 bytes. If the file size is extended further by another write of 512 bytes, an additional fragment must be allocated to the file. Applying this example to a file system with 4096-byte fragments, disk space allocation occurs only once, as part of the first write operation. No additional allocation activity must be performed as part of the second write operation since the initial 4096-byte fragment allocation is large enough to hold the data added by the second write operation.

Allocation activity adds performance overhead to file system operations. However, allocation activity can be minimized for file systems with fragment sizes smaller than 4096 bytes if the files are extended by 4096 bytes at a time.

Free Space Fragmentation

Using fragments smaller than 4096 bytes can cause greater fragmentation of the free space on the disk. For example, consider an area of the disk that is divided into eight fragments of 512 bytes each. Suppose that different files, requiring 512 bytes each, have written to the first, fourth, fifth, and seventh fragments in this area of the disk, leaving the second, third, sixth, and eighth fragments free. Although four fragments representing 2048 bytes of disk space are free, no partial logical block requiring four fragments (or 2048 bytes) is allocated for these free fragments, since the fragments in a single allocation must be contiguous.

Because the fragments allocated for a file or directory logical blocks must be contiguous, free space fragmentation can cause a file system operation that requests new disk space to fail even though the total amount of available free space is large enough to satisfy the operation. For example, a write operation that extends a zero-length file by one logical block requires 4096 bytes of contiguous disk space to be allocated. If the file system free space is fragmented and consists of 32 noncontiguous 512-byte fragments or a total of 16 KB of free disk space, the write operation will fail because eight contiguous fragments (or 4096 bytes of contiguous disk space) are not available to satisfy the write operation.

A file system with an unmanageable amount of fragmented free space can be defragmented with the **defragfs** command. Running the **defrags** command has a positive impact on performance.

Increased Fragment Allocation Map Size

More virtual memory and file system disk space might be required to hold fragment allocation maps for file systems with a fragment size smaller than 4096 bytes. Fragments serve as the basic unit of disk space allocation, and the allocation state of each fragment within a file system is recorded in the file system fragment allocation map.

Understanding Journaled File System Size Limitations

The maximum size for a journaled file system (JFS) is defined when the file system is created. When you create a JFS, there are five significant issues to consider:

- "Number of I-nodes"
- "Allocation Group Size"
- "File System Fragment Addressability"
- "Journaled File System Log Size Issues"
- "Maximum Journaled File System Size" on page 84.

Number of I-nodes

The total number of i-nodes in a file system limits the total number of files and the total size of the file system. The JFS provides the number of bytes per i-node (NBPI) parameter that affects the number of i-nodes in a file system. JFS supports NBPI values of `512`, `1024`, `2048`, `4096`, `8192`, `16384`, `32768`, `65536`, and `131072`.

One i-node is created for every NBPI bytes of allocation group space allocated to the file system. An allocation group can be partially allocated, though the full number of i-nodes per allocation group is still allocated. NBPI is inversely proportional to the total number of i-nodes in a file system.

The JFS restricts all file systems to 16M (2^{24}) i-nodes.

Allocation Group Size

The JFS segregates file system space into groupings of i-nodes and disk blocks for user data. These groupings are called allocation groups. The allocation group size can be specified when the file system is created. The allocation group sizes are 8M, 16M, 32M, and 64M. Each allocation group size has an associated NBPI range. The ranges are defined by the following table:

Allocation Group Size in Megabytes	Allowable NBPI Values
8	512, 1024, 2048, 4096, 8192, 16384
16	1024, 2048, 4096, 8192, 16384, 32768
32	2048, 4096, 8192, 16384, 32768, 65536
64	4096, 8192, 16384, 32768, 65536, 131072

File System Fragment Addressability

The JFS supports four fragment sizes of `512`, `1024`, `2048`, and `4096` byte units of contiguous disk space. The JFS maintains fragment addresses in i-nodes and indirect blocks as 28-bit numbers. Each fragment must be addressable by a number from 0 to (2^{28}).

Journaled File System Log Size Issues

Another size-related issue is the size of the JFS log. In most instances, multiple journaled file systems use a common log configured to be 4 MB in size. For example, after initial installation, all file systems within the root volume group use logical volume hd8 as a common JFS log. The default logical volume partition size is 4 MB, and the default log size is one partition, therefore, the root volume group normally contains a 4 MB JFS log. When file systems exceed 2 GB or when the total amount of file system space using a single log exceeds 2 GB, the default log size might not be sufficient. In either case, the log sizes are scaled upward as the file system size increases. When the size of the log logical volume is changed, the **logform** command must be run to reinitialize the log before the new space can be used. The JFS log is limited to a maximum size of 256 MB.

Maximum Journaled File System Size

The maximum JFS size is defined when the file system is created. For example, selecting an nbpi ratio of 512 will limit the file system to a size of 8 GB ($512 * 2^{24} = 8$ GB). When creating a JFS file system, the factors listed above (NBPI, fragment size, and allocation group size) need to be weighed carefully. The file system size limitation is the minimum of $NBPI * 2^{24}$ or $Fragment\ Size * 2^{28}$

Understanding Large Files

This section contains information about creating large files and file allocation. Large files are only applicable to AIX 4.2 or later.

Note: All JFS2 file systems support large files.

Create File Systems Enabled for Large Files

File systems enabled for large files can be created with the **crfs** and **mkfs** commands. Both commands have a new option (`bf=true`) to specify file systems enabled for large files. The JFS SMIT menus can create these file systems also.

Large File Geometry

Note: Large file geometry is specific to JFS.

In file systems enabled for large files, file data stored before the 4 MB file offset is allocated in 4096 byte blocks. File data stored beyond the 4 MB file offset is allocated with large disk blocks of 128 KB in size. The large disk blocks are actually 32 contiguous 4096 byte blocks. For example, a 132 MB file in a file system enabled for large files has 1024 4 KB disk blocks and 1024 128KB disk blocks. In a regular file system, the 132 MB file requires 33 single indirect blocks (each filled with 1024 4 KB disk addresses). However, the large file geometry requires only two single indirect blocks for the 132 MB file.

Free Space Fragmentation

Note: Free space fragmentation is specific to JFS.

Large disk blocks require 32 contiguous 4KB blocks. If you write to large files beyond the 4MB, file offset will fail with ENOSPC if the file system does not contain 32 unused contiguous 4KB blocks.

Note: The file system may have thousands of free blocks, but if 32 of them are not contiguous, the allocation will fail.

The **defragfs** command reorganizes disk blocks to provide larger contiguous free block areas.

Zeroing kproc for Large Allocations

Note: Zeroing kproc is specific to JFS.

The JFS is required to initialize all new disk allocations. The JFS starts the kernel kproc procedure used to zero initial file allocations when mounting the first large file enabled file system on your system. The kproc procedure remains once the file system enabled for large files has successfully unmounted.

Understanding Sparse Files

A file is a sequence of indexed blocks of arbitrary size. The indexing is accomplished through the use of direct mapping or indirect index blocks from the files inode. Each index within a files address range is not required to map to an actual data block.

A file that has one or more indexes that are not mapped to a data block is referred to as being *sparsely-allocated* or a *sparse file*. A sparse file will have a size associated with it, but it will not have all of the data blocks allocated to fulfill the size requirements. To identify if a file is sparsely-allocated, use the **fileplace** command. It will indicate all blocks in the file that are not currently allocated.

> **NOTE:** In most circumstances, **du** can also be used to determine if the number of data blocks allocated to a file do not match those required to hold a file of its size. A compressed filesystem might show the same behavior for files that are not sparsely-allocated.

A sparse file is created when an application extends a file by seeking to a location outside the currently allocated indexes, but the data that is written does not occupy all of the newly assigned indexes. The new file size reflects the farthest write into the file.

A read to a section of a file that has unallocated data blocks results in a default value of null bytes being returned. A write to a section of a file that has unallocated data blocks causes the neccesary data blocks to be allocated and the data written.

This behavior can affect file manipulation or archival commands. For example, the following commands do not preserve the sparse allocation of a file:

```
cp     mv     tar     cpio
```

> **NOTE:** In the case of **mv**, this only applies to moving a file to another filesystem. If the file is moved within the same filesystem, it will remain sparse.

The result of a file being copied or restored from the preceding commands has each data block allocated, and thus have no sparse characteristics. However, the following archival commands either preserve sparse characteristics or actively sparse a file:

```
backup
restore
pax
```

Because it is possible to overcommit the resources of a file system with sparse files, care should be taken in the use and maintenance of files of this type.

Enhanced Journaled File System

Understanding File System Block Size

The enhanced journaled file system (JFS2) supports multiple file system block sizes of 512, 1024, 2048, and 4096. Smaller block sizes minimize wasted disk space by more efficiently storing the data in a file or directory’s partial logical blocks. Smaller block sizes also result in additional operational overhead.

The block size for a JFS2 is specified during its creation. Different file systems can have diferent block sizes, but only one block size can be used within a single file system.

Variable Number of I-nodes for Enhanced Journaled File System

JFS2 allocates i-nodes as needed. Therefore, the number of i-nodes available is limited by the size of the file system itself.

Specifying File System Block Size

File system block size is specified during the file system's creation with the **crfs** and **mkfs** commands or by using the System Management Interface Tool (SMIT). The decision of file system block size should be based on the projected size of files contained by the file system.

Identifying File System Block Size

The file system block size value can be identified through the **lsfs** command or the System Management Interface Tool (SMIT). For application programs, the **statfs** subroutine can be used to identify the file system block size.

Compatibility and Migration

Note: The enhanced journaled file system (JFS2) is a new file system and is not compatible with any previous version of this operating system.

Device Driver Limitations

A device driver must provide disk block addressability that is the same or smaller than the file system block size.

Performance Costs

Although file systems that use block sizes smaller than 4096 bytes as their allocation unit might require substantially less disk space than those using the default allocation unit of 4096 bytes, the use of smaller block sizes can incur performance degradation.

Increased Allocation Activity

Because disk space is allocated in smaller units for a file system with a block size other than 4096 bytes, allocation activity can occur more often when files or directories are repeatedly extended in size. For example, a write operation that extends the size of a zero-length file by 512 bytes results in the allocation of one block to the file, assuming a block size of 512 bytes. If the file size is extended further by another write of 512 bytes, an additional block must be allocated to the file. Applying this example to a file system with 4096-byte blocks, disk space allocation occurs only once, as part of the first write operation. No additional allocation activity is performed as part of the second write operation since the initial 4096-byte block allocation is large enough to hold the data added by the second write operation.

Increased Block Allocation Map Size

More virtual memory and file system disk space might be required to hold block allocation maps for file systems with a block size smaller than 4096 bytes. Blocks serve as the basic unit of disk space allocation, and the allocation state of each block within a file system is recorded in the file system block allocation map.

Understanding Enhanced Journaled File System Size Limitations

The maximum size for an enhanced journaled file system is architecturally limited to 4 Petabytes. I-nodes are dynamically allocated by JFS2, so you don't need to consider how many i-nodes you may need when creating a JFS2 file system. You need to consider the size of the file system log.

Enhanced Journaled File System Log Size Issues

In most instances, multiple journaled file systems use a common log configured to be 4 MB in size. When file systems exceed 2 GB or when the total amount of filesystem space using a single log exceeds 2 GB, the default log size might not be sufficient. In either case, scale log sizes upward as the file system size increases. The JFS log is limited to a maximum size of 256 MB.

Mounting Overview

Mounting makes file systems, files, directories, devices, and special files available for use at a particular location. It is the only way a file system is made accessible. The **mount** command instructs the operating system to attach a file system at a specified directory.

You can mount a file or directory if you have access to the file or directory being mounted and write permission for the mount point. Members of the system group can also perform device mounts (in which devices or file systems are mounted over directories) and the mounts described in the **/etc/filesystems** file. A user operating with root user authority can mount a file system arbitrarily by naming both the device and the directory on the command line. The **/etc/filesystems** file is used to define mounts to be automatic at system initialization. The **mount** command is used to mount after system startup.

Understanding Mount Points

A *mount point* is a directory or file at which a new file system, directory, or file is made accessible. To mount a file system or a directory, the mount point must be a directory; and to mount a file, the mount point must be a file.

Typically, a file system, directory, or file is mounted over an empty mount point, but that is not required. If the file or directory that serves as the mount point contains any data, that data is not accessible while it is mounted over by another file or directory. In effect, the mounted file or directory covers what was previously in that directory. The original directory or file that has been mounted over is accessible again once the mount over it is undone.

When a file system is mounted over a directory, the permissions of the root directory of the mounted file system take precedence over the permissions of the mount point. The one exception involves the .. (dot dot) parent directory entry in the mounted-over directory. In order for the operating system to access the new file system, the mount point parent directory information must be available.

For example, if the current working directory is **/home/frank**, the command **cd ..** changes the working directory to **/home**. If **/home/frank** directory is the root of a mounted file system, the operating system must find the parent directory information in the **/home/frank** directory in order for the **cd ..** command to succeed.

For any command that requires parent directory information in order to succeed, users must have search permission in the mounted-over directory. Failure of the mounted-over directory to grant search permission can have unpredictable results, especially since the mounted-over directory permissions are not visible. A common problem is failure of the **pwd** command. Without search permission in the mounted-over directory, the **pwd** command returns this message:

```
pwd: Permission denied
```

This problem can be avoided by always setting the permissions of the mounted-over directory to at least 111.

Mounting File Systems, Directories, and Files

There are two types of mounts, a remote mount and a local mount. *Remote mounts* are done on a remote system on which data is transmitted over a telecommunication line. Remote file systems, such as Network File System (NFS), require that the files be exported before they can be mounted. *Local mounts* are mounts done on your local system.

Each file system is associated with a different device (logical volume). Before you can use a file system, it must be connected to the existing directory structure (either the root file system or to another file system that is already connected). The **mount** command makes this connection.

The same file system, directory, or file can be accessed by multiple paths. For example, if you have one database and several users using this database, it can be useful to have several mounts of the same database. Each mount should have its own name and password for tracking and job-separating purposes. This is accomplished by mounting the same file system on different mount points. For example, you can mount from `/home/server/database` to the mount point specified as `/home/user1`, `/home/user2`, and `/home/user3`:

```
/home/server/database        /home/user1
/home/server/database        /home/user2
/home/server/database        /home/user3
```

A file system, directory, or file can be made available to various users through the use of symbolic links. Symbolic links are created with the **ln -s** command. Linking multiple users to a central file ensures that all changes to the file are reflected each time a user accesses the file.

Controlling Automatic Mounts

Mounts can be set to occur automatically during system initialization. There are two types of automatic mounts. The first type consists of those mounts that are required to boot and run the system. These file systems are explicitly mounted by the boot process. The stanzas of such file systems in the **/etc/filesystems** file have `mount = automatic`. The second type of automatic mount is user-controlled. These file systems are mounted by the **/etc/rc** script when it issues the **mount all** command. The stanzas of user-controlled automatic mounts have `mount = true` in **/etc/filesystems**.

The **/etc/filesystems** file controls automatic mounts; they are done hierarchically, one mount point at a time. They can also be placed in a specific order that can be changed and rearranged.

The **/etc/filesystems** file is organized into stanzas, one for each mount. A stanza describes the attributes of the corresponding file system and how it is mounted. The system mounts file systems in the order they appear in the **/etc/filesystems** file. The following is an example of stanzas within the **/etc/filesystems** file:

```
/:
 dev=/dev/hd4
 vol="root"
 mount=automatic
 check=false
 free=true
 vfs=jfs
 log=/dev/hd8
 type-bootfs

/home:
 dev=/dev/hd1
 vfs=jfs
 log=/dev/hd8
 mount=true
 check=true
 vol="/home"
 free=false
```

```
/usr:
 /dev=/dev/hd2
 vfs=jfs
 log=/dev/hd8
 mount=automatic
 check=false
 type=bootfs
 vol="/usr"
 free=true
```

You can edit the **/etc/filesystems** file to control the order in which mounts occur. If a mount is unsuccessful, any of the following mounts defined in the **/etc/filesystems** file continue to mount. For example, if the mount of the **/home** file system is unsuccessful, the mount for the **/usr** file system continues and be mounted. Mounts can be unsuccessful for reasons such as typographical errors, dependency, or a system problem.

Understanding Mount Security for Diskless Workstations

Diskless workstations must have the ability to create and access device-special files on remote machines to have their **/dev** directories mounted from a server. Because servers cannot distinguish device-special files intended for a client from those intended for the server, a user on the server might be able to access the physical devices of the server using the special files of the client device.

For example, the ownership for a **tty** is automatically set to the user using the **tty**. If the user IDs are not the same on both the client and server, a nonprivileged user on the server can access a **tty** that is being used by a different user on the server.

A user who is privileged on a client can create device-special files to match physical devices on the server and have them not require privilege for access. The user can then use an unprivileged account on the server to access the normally protected devices using the new device-special files.

A similar security problem involves the use of **setuid** and **setgid** programs on the client and server. Diskless clients must be able to create and run**setuid** and **setgid** programs on the server for normal operation. Again, the server cannot distinguish between those programs intended for the server and those intended for the client.

In addition, the user IDs and group IDs might not match between the server and client, so users on the server might be able to run programs with capabilities that were not intended for them.

The problem exists because the **setuid** and **setgid** programs and device-special files should only be usable on the machine that created them.

The solution is to use security options to the **mount** command that restrict the ability to use these objects. These options can also be used in stanzas in the **/etc/filesystems** file.

The `nosuid` option in the mount command prevents the execution of **setuid** and **setgid** programs that are accessed via the resulting mounted file system. This option is used for any file system that is being mounted on a particular host for use only by a different host (for example, exported for diskless clients).

The `nodev` option in the **mount** command prevents the opening of devices using device special files that are accessed via the resulting mounted file system. This option is also used for any file system that is being mounted for use only by a different host (for example, exported for diskless clients).

Diskless Mounts

Although the file system of a diskless workstation is mounted from a server **/exports** directory, to the diskless machine, the file system looks just like the file system on a standalone machine.

The following shows the relationship between server exports, and the diskless workstation mount points:

Server Exports	**Diskless Imports**
/export/root/*HostName*	/ (root)
/export/exec/*SPOTName*	**/usr**
/export/home/*HostName*	**/home**
/export/share	/usr/share
/export/dump	Used by diskless client as dump space
/export/swap	Used by diskless clients as remote paging space.

For more information about the **/export** directory, see “Understanding the /export Directory” on page 75.

Securing Diskless Mounts

In general, users on a server do not have any access to the **/export** directory.

Exporting the /export/root Directory

The **/export/root** directory must be exported with read/write permissions, and the root user on the server must have access. However, you might want to mount this directory with the following options of the **mount** command:

nosuid	Prevents a user on the server from running the **setuid** programs of the client
nodev	Prevents a user from accessing the server devices using a device-special file of the client.

An alternative to mounting the **/export/root** directory with these options is to avoid giving users running on the server any access to the **/export/root** directory.

Exporting the /export/exec Directory

The **/export/exec** directory is exported with read-only permissions and must provide root access. However, you might want to mount this directory with the following options of the **mount** command:

nosuid	Prevents a user on the server from running the **setuid** programs of the client . If you are exporting the server **/usr** directory, you cannot use the **nousid** option.
nodev	Prevents a user from accessing the server devices using a device-special file of the client.

Exporting the /export/share Directory

The **/export/share** directory is exported with read-only permissions and must provide root access. Because this directory generally contains only data (no executables or devices), you do not need to use the mount security options.

Exporting the /export/home Directory

There are several ways to mount a user **/home** directory:

- You can mount the **/export/home/***Clienthostname* directory over the client **/home** directory. In this case, the client has read/write permissions and the root user has access. To ensure system security, mount the **/export/home** directory with the following options to the **mount** command:

nosuid	Prevents a user on the server from running the **setuid** programs of the client.
nodev	Prevents a user from accessing the server devices using a device-special file of the client.

- You can mount the **/home** directory on the server over the **/home** directory of the client. In this case, the **/home** directory is exported with read/write permissions and without root access. To ensure system security, mount the **/home** directory on both the server and client with the `nosuid` and `nodev` options of the **mount** command.
- Alternatively, you can mount on the client each **/home/***UserName* directory on the server over the **/home/***Username* directory on the client so users can log in to different machines and still have access to their home directories. In this case, the **/home/***Username* directories on the server and clients are both mounted with the `nousid` and `nodev` options of the **mount** command.

Exporting the /export/dump Directory

Export the **/export/dump/***Clienthostname* directory with read/write permissions and root access. Users on the server do not have any access to the **/export/dump/***Clienthostname* files.

Exporting the /export/swap Directory

Export the **/export/swap/***Clienthostname* file with read/write permissions and root access. No security measures are necessary. Users on the server do not have any access to the **/export/swap/***Clienthostname* files.

Chapter 8. Paging Space and Virtual Memory

This chapter explains the different types of paging space and allocation policies. For performance implications related to paging spaces, see the section on performance considerations of paging spaces in *AIX 5L Version 5.1 Performance Management Guide*.

Topics covered are:
- "Paging Space Overview"
- "Managing Paging Spaces" on page 96
- "Understanding Paging Space Allocation Policies" on page 94
- "Virtual Memory Manager (VMM) Overview" on page 97.

Paging Space Overview

A *paging space* is fixed disk storage for information that is resident in virtual memory, but is not currently being accessed. A paging space, also called a swap space, is a logical volume with the attribute type equal to paging. This type of logical volume is referred to as a paging-space logical volume, or simply paging space. When the amount of free real memory in the system is low, programs or data that have not been used recently are moved from real memory to paging space to release real memory for other activities.

The following articles provide more information about paging spaces:
- "Managing Paging Spaces" on page 96
- "Understanding Paging Space Allocation Policies" on page 94

The following procedures explain various ways of managing paging spaces:
- Adding and Activating a Paging Space
- Changing or Removing a Paging Space
- Resizing or Moving the hd6 Paging Space.

The "Virtual Memory Manager (VMM) Overview" on page 97 explains how virtual memory is related to paging space.

The default paging space size is determined during the system customization phase of installation according to the following standards:
- Paging space can use no less than 16 MB, except for hd6 which can use no less than 32 MB in AIX 4.2.1 and no less than 64 MB in AIX 4.3 and later.
- Paging space can use no more than 20% of total disk space.
- If real memory is less than 256 MB, paging space is two times real memory.
- If real memory is greater than or equal to 256 MB, paging space is 512 MB.

Another type of paging space is available that can be accessed through a device that uses an NFS server for paging-space storage. For an NFS client to access this paging space, the NFS server must have a file created and exported to that client. The file size represents the paging space size for the client.

The logical volume paging space is defined by making a new paging-space logical volume or by increasing the size of existing paging-space logical volumes. To increase the size of an NFS paging space, the file that resides on the server must be increased by the correct actions on the server.

The total space available to the system for paging is the sum of the sizes of all active paging-space logical volumes.

Understanding Paging Space Allocation Policies

The operating system uses two modes for paging space allocation. The setting of the **PSALLOC** environment variable determines the paging space allocation mode. The default mechanism is the late paging space allocation algorithm. The user can switch to an early paging space allocation mode by setting the value of the **PSALLOC** environment variable to `early`.

Paging Space Considerations

The amount of paging space required depends on the type of activities performed on the system. If paging space runs low, processes can be lost, and if paging space runs out, the system can panic. When a paging-space low condition is detected, define additional paging space.

The system monitors the number of free paging space blocks and detects when a paging-space shortage exists. When the number of free paging-space blocks falls below a threshold known as the paging-space warning level, the system informs all processes (except **kprocs**) of this condition by sending the **SIGDANGER** signal. If the shortage continues and falls below a second threshold known as the paging-space kill level, the system sends the **SIGKILL** signal to processes that are the major users of paging space and that do not have a signal handler for the **SIGDANGER** signal (the default action for the **SIGDANGER** signal is to ignore the signal). The system continues sending **SIGKILL** signals until the number of free paging-space blocks is above the paging-space kill level.

Processes that dynamically allocate memory can ensure that sufficient paging space exists by monitoring the paging-space levels with the **psdanger** subroutine or by using special allocation routines. You can use the **disclaim** subroutine to prevent processes from ending when the paging-space kill level is reached. To do this, define a signal handler for the **SIGDANGER** signal and release memory and paging-space resources allocated in their data and stack areas and in shared memory segments.

Comparing Late and Early Paging Space Allocation

The operating system uses the **PSALLOC** environment variable to determine the mechanism used for memory and paging space allocation. If the **PSALLOC** environment variable is not set, is set to `null`, or is set to any value other than `early`, the system uses the default late allocation algorithm.

The default late allocation algorithm for memory and paging space allocation assists in the efficient use of disk resources and supports applications of customers who wish to take advantage of a sparse allocation algorithm for resource management. The late allocation algorithm does not reserve paging space when a memory request is made; it approves the request and assigns paging space when pages are touched. Some programs allocate large amounts of virtual memory and then use only a fraction of the memory. Examples of such programs are technical applications that use sparse vectors or matrices as data structures. The late allocation algorithm is also more efficient for a real-time, demand-paged kernel such as the one in the operating system.

For AIX 4.3.2 and later, the late allocation algorithm is modified to further delay the allocation of paging space. As mentioned above, before AIX 4.3.2, paging space was allocated when a page was touched. However, this paging space might never be used, especially on systems with large real memory where paging is rare. Therefore, the allocation of paging space is delayed until it is necessary to page out the page, which results in no wasted paging space allocation. This does result, however, in additional overcommitment of paging space. On a system where enough virtual memory is accessed that paging is necessary, the amount of paging space required can be as much as was required on previous releases.

It is possible to overcommit resources when using the late allocation algorithm for paging space allocation. In this case, when one process gets the resource before another, a failure results. The operating system attempts to avoid complete system failure by killing processes affected by the resource overcommitment.

The **SIGDANGER** signal is sent to notify processes that the amount of free paging space is low. If the paging space situation reaches an even more critical state, selected processes that did not receive the **SIGDANGER** signal are sent a **SIGKILL** signal.

The user can use the **PSALLOC** environment variable to switch to an early allocation algorithm for memory and paging space allocation. The early allocation mechanism allocates paging space for the executing process at the time the memory is requested. If there is insufficient paging space available at the time of the request, the early allocation mechanism fails the memory request.

If the **PSALLOC** environment variable is set to `early`, then every program started in that environment from that point on, but not including currently running processes, runs in the early allocation environment. In the early allocation environment, interfaces such as the **malloc** subroutine and the **brk** subroutine will fail if sufficient paging space cannot be reserved when the request is made.

Processes run in the early allocation environment mode are not sent the **SIGKILL** signal if a low paging space condition occur.

The following memory allocation interface subroutines are affected by a switch to an early allocation environment:

- **malloc**
- **free**
- **calloc**
- **realloc**
- **brk**
- **sbrk**
- **shmget**
- **shmctl**

Setting the PSALLOC Environment Variable for Early Allocation Mode

The following examples show the different ways a user can set the **PSALLOC** environment variable to `early` so that applications run in early allocation mode. The examples also explain the results of each method.

1. Entering the following command on a shell command line:
   ```
   PSALLOC=early;export PSALLOC
   ```
   causes all subsequent commands run from that shell session to run in early allocation mode.
2. Entering the following command in a **.shrc** file or **.kshrc** file:
   ```
   PSALLOC=early;export PSALLOC
   ```
   causes all processes in the user's login session, with the exception of the login shell, to run under early allocation mode. This command also protects the processes from the **SIGKILL** signal mechanism.
3. Making the following entry in the **/etc/environment** file:
   ```
   PSALLOC=early
   ```
   causes all processes in the system, except the **init** process (process ID 1) to run in the early allocation mode and be protected from the **SIGKILL** signal mechanism.
4. You can use the **putenv** subroutine to set the **PSALLOC** environment variable to `early` from inside a program. The early allocation behavior takes effect at the next call to the **exec** subroutine.

Early Allocation Mode Considerations

The early allocation algorithm guarantees as much paging space as requested by a memory allocation request. Thus, correct paging space allocation on the system disk is important for efficient operations. When available paging space drops below a certain threshold, new processes cannot be started and currently running processes might not be able to get more memory. Any processes running under the default late allocation mode become highly vulnerable to the **SIGKILL** signal mechanism. In addition, because the operating system kernel sometimes requires memory allocation, it is possible to crash the system by using up all available paging space.

Before you use the early allocation mode throughout the system, it is very important to define an adequate amount of paging space for the system. The paging space required for early allocation mode is almost always greater than the paging space required for the default late allocation mode. How much paging space to define depends on how your system is used and what programs you run. A good starting point for determining the right mix for your system is to define a paging space four times greater than the amount of physical memory.

Certain applications can use extreme amounts of paging space if they are run in early allocation mode. The AIXwindows server currently requires more than 250 MB of paging space when the application runs in early allocation mode. The paging space required for any application depends on how the application is written and how it is run.

All commands and subroutines that show paging space and process memory use include paging space allocated under early allocation mode. The **lsps** command uses the **-s** flag to display total paging space allocation, including paging space allocated under early allocation mode.

Programming Interface

The programming interface that controls the paging space allocation mode uses the **PSALLOC** environment variable. To ensure that an application always runs under the desired mode (with or without early paging space allocation), do the following:

1. Use the **getenv** subroutine to examine the current state of the **PSALLOC** environment variable.
2. If the value of the **PSALLOC** environment variable is not the value required by the application, use the **setenv** subroutine to alter the value of the environment variable. Because only the **execve** subroutine examines the state of the **PSALLOC** environment variable, call the **execve** subroutine with the same set of parameters and environment received by the application. When the application reexamines the state of the **PSALLOC** environment variable and finds the correct value, the application continues normally.
3. If the **getenv** subroutine reveals that the current state of the **PSALLOC** environment variable is correct, no modification is needed. The application continues normally.

Managing Paging Spaces

The following commands are used to manage paging space:

chps	Changes the attributes of a paging space
lsps	Displays the characteristics of a paging space
mkps	Adds an additional paging space
rmps	Removes an inactive paging space
swapoff	Deactives one or more paging space
swapon	Activates a paging space.

The **mkps** command uses the **mklv** command with a specific set of options when creating a paging space logical volume. To create NFS paging spaces, the **mkps** command uses the **mkdev** command with another set of options. Some of the following options are required for all logical volume paging spaces:

- Paging type
- No bad-block relocation
- No mirroring.

The following options are used to maximize paging performance with a logical volume:
- Allocate in the middle of the disk to reduce disk arm travel
- Use multiple paging spaces, each allocated from a separate physical volume.

For NFS paging spaces, the **mkps** command needs the host name of the NFS server and the path name of the file that is exported from the server.

The **swapon** command is used during early system initialization to activate the initial paging-space device. During a later phase of initialization, when other devices become available, the **swapon** command is used to activate additional paging spaces so that paging activity occurs across several devices.

The **swapoff** command is used to deactive a paging space without rebooting the system. Information in the paging space is moved to other other active paging space areas. The deactivated paging space can then be removed using the **rmps** command.

The **/etc/swapspaces** file specifies the paging-space devices that are activated by the **swapon -a** command. A paging space is added to this file when it is created by the **mkps -a** command, removed from the file when it is deleted by the **rmps** command, and added or removed by the **chps -a** command.

Virtual Memory Manager (VMM) Overview

The Virtual Memory Manager (VMM) provides the virtual memory facilities that are used by the other parts of the system to implement the following:
- Virtual address space of processes
- Sharing of executables
- Shared memory segments
- Mapped files.

The VMM implements virtual memory, allowing the creation of segments larger than the physical memory available in the system. The segments are divided into fixed-size units called *pages*. Each page in a segment can be in physical memory or stored on disk until it is needed. When a process accesses a page that is not present in physical memory, the VMM reads the page into memory; this is called a *PageIn*. When physical memory is not available, the VMM writes pages to disk; this is called a *PageOut* or *PageSteal*.

The following are some of the segment types:

Working storage	Segments are used to implement the data areas of processes and shared memory segments. The pages for working storage segments are stored in the paging spaces configured in the system.
Persistent storage	Segments are used to manipulate files and directories. When a persistent storage segment is accessed, the pages are read and written from its file system.
Client storage	Segments are used to implement some virtual file systems like Network File System (NFS) and the CD-ROM file system. The storage for client segment pages can be in a local or remote computer.

Chapter 9. Backup and Restore

This chapter contains information about methods of backing up and restoring information as well as backup policies for specific needs.

Topics covered are:
- "Backup Overview"
- "Developing a Backup Strategy" on page 101
- "Backing Up User Files or File Systems" on page 103
- ″Backing Up Your System″.

Backup Overview

After your system is in use, your next consideration is backing up file systems, directories, and files. Files and directories represent a significant investment of time and effort. At the same time, all computer files are potentially easy to change or erase, either intentionally or by accident. If you take a careful and methodical approach to backing up your file systems, you are always able to restore recent versions of files or file systems with little difficulty. When a hard disk crashes, the information contained on that disk is destroyed. The only way to recover the destroyed data is to retrieve the information from your backup copy.

Backup Methods

Several different methods exist for backing up information. One of the most frequently used methods is called backup by name; it is also called file name archive. This method of backup is done when the **i** flag is specified and is used to make a backup copy of individual files and directories. It is a method commonly used by individual users to back up their accounts.

Another frequently used method is called backup by file system, also called backup by i-node or file system archive. This method of backup is done when the **i** flag is *not* specified. It is used to make a backup copy of an entire file system and is the method commonly used by system administrators to back up large groups of files, such as all of the user accounts in **/home**. A file system backup allows incremental backups to be performed easily. An incremental backup backs up all files that have been modified since a specified previous backup.

The **compress** and **pack** commands enable you to compress files for storage, and the **uncompress** and **unpack** commands unpack the files once they have been restored. The process of packing and unpacking files takes time, but once packed, the data uses less space on the backup medium.

Several commands create backups and archives. Because of this, data that has been backed up needs to be labeled as to what command was used when doing the backup, and how the backup was made (by name or by file system). The **backup** command is the most frequently used, but other commands serve specific purposes:

backup	Backs up files by name or by file system.
mksysb	Creates an installable image of the rootvg volume group.
cpio	Copies files into and out of archive storage.
dd	Converts and copies a file. Commonly used to convert and copy data to and from systems running other operating systems, for example, mainframes. **dd** does not group multiple files into one archive; it is used to manipulate and move data.
tar	Manipulates tar format archives.
rdump	A network command that backs up files by file system onto a remote machine's device.
pax	POSIX-conformant archive utility that can read and write **tar** and **cpio** archives.

Deciding on a Backup Policy

No single backup policy can meet the needs of all users. A policy that works well for a system with one user, for example, could be inadequate for a system that serves five or ten different users. Likewise, a policy developed for a system on which many files are changed daily would be inefficient for a system on which data changes infrequently. Whatever the appropriate backup strategy for your site, it is very important that one exist and that backups be done frequently and regularly. It is difficult to recover from data loss if a good backup strategy has not been implemented.

Only you can determine the best backup policy for your system, but the following general guidelines might be helpful:

- Make sure you can recover from major losses.

  Can your system continue to run after any single fixed disk failure? Can you recover your system if all the fixed disks should fail? Could you recover your system if you lost your backup diskettes or tape to fire or theft? If data were lost, how difficult would it be to re-create it? Think through possible, even unlikely, major losses, and design a backup policy that enables you to recover your system after any of them.
- Check your backups periodically.

  Backup media and their hardware can be unreliable. A large library of backup tapes or diskettes is useless if data cannot be read back onto a fixed disk. To make certain that your backups are usable, display the table of contents from the backup tape periodically (using **restore -T** or **tar -t** for archive tapes). If you use diskettes for your backups and have more than one diskette drive, read diskettes from a drive other than the one on which they were created. You might want the security of repeating each level 0 backup with a second set of media. If you use a streaming tape device for backups, you can use the **tapechk** command to perform rudimentary consistency checks on the tape.
- Keep old backups.

  Develop a regular cycle for reusing your backup media; however, do not reuse all of your backup media. Sometimes it is months before you or some other user of your system notices that an important file is damaged or missing. Save old backups for such possibilities. For example, you could have the following three cycles of backup tapes or diskettes:
  - Once per week, recycle all daily diskettes except the Friday diskette.
  - Once per month, recycle all Friday diskettes except for the one from the last Friday of the month. This makes the last four Friday backups always available.
  - Once per quarter, recycle all monthly diskettes except for the last one. Keep the last monthly diskette from each quarter indefinitely, perhaps in a different building.
- Check file systems before backing up.

  A backup made from a damaged file system might be useless. Before making your backups, it is good policy to check the integrity of the file system with the **fsck** command.
- Ensure files are not in use during a backup.

  Do not use the system when you make your backups. If the system is in use, files can change while they are being backed up, and the backup copy will not be accurate.
- Back up your system before major changes are made to the system.

  It is always good policy to back up your entire system before any hardware testing or repair work is performed or before you install any new devices, programs, or other system features.

  Note: For the backup of named pipes (FIFO special files) the pipes can be either closed or open. However, the restoration fails when the backup is done on open named pipes. When restoring a FIFO special file, its i-node is all that is required to recreate it because it contains all its characteristic information. The content of the named pipe is not relevant for restored. Therefore, the file size during backup is zero (all the FIFOs closed) before the backup is made.

Attention: System backup and restoration procedures require that the system be restored on the same type of platform from which the backup was made. In particular, the CPU and I/O planar boards must be of the same type.

Understanding Backup Media

Several types of backup media are available. The types of backup media available to your specific system configuration depend upon your software and hardware. The types most frequently used are 8-mm tape, 9-track tape, and the 3.5-inch diskette.

For backing up individual files and file systems, diskettes are the standard medium. Unless you specify a different device using the **backup -f** command, the **backup** command automatically writes its output to **/dev/rfd0**, which is the diskette drive. To back up to the default tape device, use **/dev/rmt0**.

Attention: Running the **backup** command results in the loss of all material previously stored on the selected output medium.

Restoring Data

After the data has been correctly backed up, there are several different methods of restoring the data based upon the type of backup command you used.

You need to know how your backup or archive was created to restore it correctly. Each backup procedure gives information about restoring data. For example, if you use the **backup** command, you can specify a backup either by file system or by name. That backup must be restored the way it was done, by file system or by name.

Several commands restore backed up data, such as:

restore	Copies files created by the **backup** command.
rrestore	Network command that copies file systems backed up on a remote machine to the local machine.
cpio	Copies files into and out of archive storage.
tar	Manipulates **tar** archives.
pax	POSIX-conformant archive utility that can read and write **tar** and **cpio** archives.

Developing a Backup Strategy

There are two methods of backing up large amounts of data:
- Complete system backup
- Incremental backup

To understand these two types of backups and which one is right for a site or system, it is important to have an understanding of file system structure and data placement. After you have decided on a strategy for data placement, you can develop a backup strategy for that data. See Implementing Scheduled Backups for an example of a backup strategy that includes weekly complete system backups and daily incremental backups.

File System Structure

It is important to understand the difference between a file system and a directory. A file system is a section of hard disk that has been allocated to contain files. This section of hard disk is accessed by mounting the file system over a directory. After the file system is mounted, it looks just like any other directory to the end user. However, because of the structural differences between the file systems and directories, the data within these entities can be managed separately.

When the operating system is installed for the first time, it is loaded into a directory structure, as shown in the following illustration.

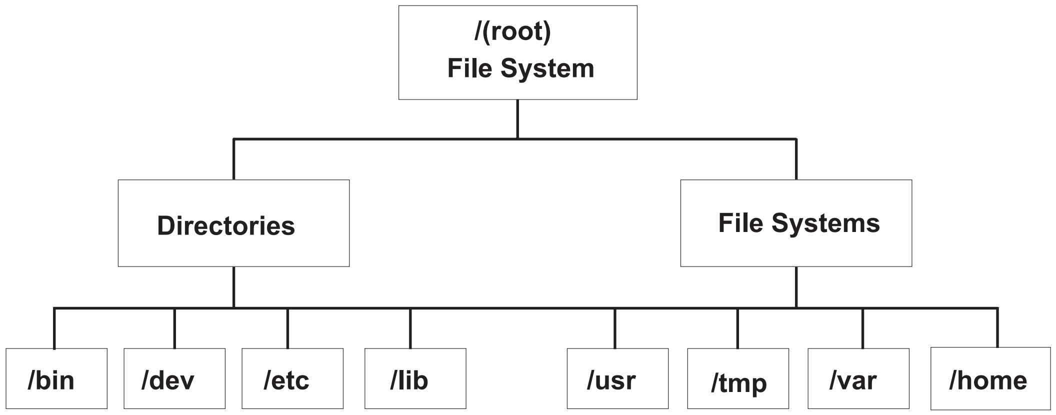

Figure 13. /root File System Tree. . This tree chart shows a directory structure with the **/root** *file system at the top, branching downward to Directories and File Systems. Directories branches* **/bin***,* **/dev***,* **/etc***, and* **/lib***. File Systems branches to* **/usr***,* **/tmp***,* **/var***, and* **/home***.*

The directories on the right (**/usr**, **/tmp**, **/var**, and **/home**) are all file systems so they have separate sections of the hard disk allocated for their use. These file systems are mounted automatically when the system is started, so the end user does not see the difference between these file systems and the directories listed on the left (**/bin**, **/dev**, **/etc**, and **/lib**).

System Data Versus User Data

Data is defined as programs or text and for this discussion is broken down into two classes:

- System data, which makes up the operating system and its extensions. This data is always to be kept in the system file systems, namely **/** (root), **/usr**, **/tmp**, **/var**, and so on.
- User data is typically local data that individuals need to complete their specific tasks. This data is to be kept in the **/home** file system or in file systems that are created specifically for user data.

User programs and text are not to be placed in file systems designed to contain system data. For example, a system manager might create a new file system and mount it over a **/local**.

Backing Up

In general, backups of user and system data are kept in case data is accidentally removed or in case of a disk failure. It is easier to manage backups when user data is kept separate from system data. The following are reasons for keeping system data separate from user data:

- User data tends to change much more often than operating system data. Backup images are much smaller if the system data is not backed up into the same image as the user data. The number of users affects the storage media and frequency required for backup.
- It is quicker and easier to restore user data when it is kept separate. Restoring the operating system along with the user data requires extra time and effort. The reason is that the method used to recover the operating system data involves booting the system from removable media (tape or CD-ROM) and installing the system backup.

To back up the system data, unmount all user file systems, including **/home** with the **umount** command. If these file systems are in use, you are not able to unmount them. Schedule the backups at low usage times so they can be unmounted; if the user data file systems remain mounted, they are backed up along with the operating system data. To ensure that only the operating system file systems are mounted, enter the following command:

```
      mount
```

The only mounted file systems are **/**, **/usr**, **/var**, and **/tmp**, and the output of the **mount** command should be similar to the following:

```
node  mounted  mounted over vfs       date         options

      /dev/hd4     /        jfs   Jun 11 10:36  rw,log=/dev/hd8

      /dev/hd2     /usr     jfs   Jun 11 10:36  rw,log=/dev/hd8

      /dev/hd9var  /var     jfs   Jun 11 10:36  rw,log=/dev/hd8

      /dev/hd      /tmp     jfs   Jun 11 10:36  rw,log=/dev/hd8
```

After you are certain that all user file systems are unmounted, See “Backing Up the System Image and User-Defined Volume Groups” on page 104 for information on backing up the operating system data.

When you finish backing up the operating system, mount the user file system using the **smit mount** command. Next, you can back up files, file systems, or other volume groups, depending on your needs. Procedures on these backups are covered later in the chapter.

Replicating a System (Cloning)

Cloning allows you to save configuration data along with user or system data. For example, you might want to replicate a system or volume group; this is sometimes called cloning. You can then install this image onto another system and can use it just like the first system. The **mksysb** command is used to clone the rootvg volume group, which contains the operating system, while the **savevg** command is used to clone a volume group. Procedures for backing up both your system and user volume groups are discussed later in the chapter.

Backing Up User Files or File Systems

Three procedures can be used to back up files and file systems: the Web-based System Manager fast path **wsm fs**, the SMIT fast paths **smit backfile** or **smit backfilesys**, and the **backup** command.

You can use the SMIT interface for backing up single and small file systems by name, such as **/home** on your local system. Note that SMIT cannot make archives in any other format than that provided by the **backup** command. Also, not every flag of the **backup** command is available through SMIT, as that makes the SMIT dialog convoluted and confusing. SMIT might hang if multiple tapes or disks are needed during the backup. See ″Backing Up by Name″ in the **backup** command.

Use the **backup** command when you want to back up large and multiple file systems. You can sepcify a level number to control how much data is backed up (full, 0; incremental, 1-9). Using the **backup** command is the only way you can specify the level number on backups.

The **backup** command creates copies in one of the two following backup formats:

- Specific files backed up by name using the **-i** flag with the **backup** command.
- Entire file systems backed up by i-node using the *-Level* and *FileSystem* parameters. One advantage of i-node backup is that the backup is defragmented when restored.

  Attention: Backing up by i-node does not work correctly for files that have a user ID (UID) or a group ID (GID) greater than 65535. These files are backed up with UID or GID truncated and will, therefore, have the wrong UID or GID attributes when restored. Backing up by name works correctly for files that have a UID or GID greater than 65535.

Backing Up the System Image and User-Defined Volume Groups

A backup image serves two purposes. One is to restore a corrupted system using the system backup image. The other is to transfer installed and configured software from one system to others. You can backup the system or volume groups using Web-based System Manager, SMIT, or command procedures.

The *rootvg volume group* is a hard disk, or group of disks, containing start up files, the Base Operating System (BOS), configuration information, and any optional software products. A *user-defined volume group* (also called *nonrootvg volume group*) typically contains data files and application software.

The Web-based System Manager and SMIT procedures use the **mksysb** command to create a backup image that can be stored either on tape or in a file. If you choose tape, the backup program writes a *boot image* to the tape, which makes it suitable for installing.

Notes:

1. Startup tapes cannot be made on or used to start a PowerPC Personal Computer.
2. If you choose the SMIT method for backup, you must first install the **sysbr** fileset in the **bos.sysmgt** software package. See Installing Optional Software and Service Updates Using SMIT in the *AIX 5L Version 5.1 Installation Guide* for information on how to install software packages and options.

Configuring before the Backup

Configure the source system before creating a backup image of it. If, however, you plan to use a backup image for installing other, differently configured target systems, create the image *before* configuring the source system.

The *source* system is the system from which you created the backup copy. The *target* system is the system on which you are installing the backup copy.

The installation program automatically installs only the device support required for the hardware configuration of the installed machine. Therefore, if you are using a system backup to install other machines, you might need to install additional devices on the source system before making the backup image and using it to install one or more target systems.

Use Web-based System Manager (type `wsm`, then select `Network`), or the SMIT fast path, **smit devinst**, to install additional device support on the source system.

- If there is sufficient disk space on the source and target systems, install all device support.
- If there is limited disk space on the source and target systems, selectively install all device support.

For information on installing optional software, see Installing Optional Software and Service Updates Using SMIT in the *AIX 5L Version 5.1 Installation Guide*.

A backup transfers the following configurations from the source system to the target system:

- Paging space information
- Logical volume information
- rootvg volume group information
- Placement of logical partitions (if the Create Map Files field is set to **yes** in the Web-based System Manager or SMIT menu).

See the section on customizing the BOS install program Customizing the BOS Install Program in the *AIX 5L Version 5.1 Installation Guide* for information about how to set installation parameters to enable you to bypass menu prompts when you install the target machine from a system backup.

Mounting and Unmounting File Systems

The Backing Up Your System procedure backs up only mounted file systems in the rootvg volume group. You must, therefore, mount all file systems you want to back up before starting. Similarly, you must unmount file systems you do *not* want backed up.

This backup procedure backs up files twice if a local directory is mounted over another local directory in the same file system. For example, if you mount **/tmp** over **/usr/tmp**, the files in the **/tmp** directory are backed up twice. This duplication might exceed the number of files a file system can hold, which can cause a future installation of the backup image to fail.

Security Considerations

If you install the backup image on other systems, you might not, for security reasons, want passwords and network addresses copied to the target systems. Also, copying network addresses to a target system creates duplicate addresses that can disrupt network communications.

Restoring a Backup Image

When installing the backup image, the system checks whether the target system has enough disk space to create all the logical volumes stored on the backup. If there is enough space, the entire backup is recovered. Otherwise, the installation halts and the system prompts you to choose more destination hard disks.

File systems created on the target system are the same size as they were on the source system, unless the **SHRINK** variable was set to `yes` in the **image.data** file before the backup image was made. An exception is the **/tmp** directory, which can be increased to allocate enough space for the **bosboot** command. For information about setting variables, see the **image.data** file in *AIX 5L Version 5.1 Files Reference*.

When it finishes installing the backup image, the installation program reconfigures the Object Data Manager (ODM) on the target system. If the target system does not have exactly the same hardware configuration as the source system, the program might modify device attributes in the following target system files:

- All files in **/etc/objrepos** beginning with ″Cu″
- All files in the **/dev** directory.

For more information about installing (or *restoring*) a backup image, see Installing BOS from a System Backup in the *AIX 5L Version 5.1 Installation Guide*.

Chapter 10. System Environment

The system environment is primarily the set of variables that define or control certain aspects of process execution. They are set or reset each time a shell is started. From the system-management point of view, it is important to ensure the user is set up with the correct values at log in. Most of these variables are set during system initialization. Their definitions are read from the **/etc/profile** file or set by default.

Topics discussed in this chapter are:
- "Profiles Overview"
- "List of Time Data Manipulation Services" on page 108
- "Enabling Dynamic Processor Deallocation" on page 108

Profiles Overview

The shell uses two types of profile files when you log in to the operating system. It evaluates the commands contained in the files and then runs the commands to set up your system environment. The files have similar functions except that the **/etc/profile** file controls profile variables for all users on a system whereas the **.profile** file allows you to customize your own environment.

The following profile and system environment information is provided:
- **/etc/profile** File
- **.profile** File
- Changing the System Date and Time
- Changing the Message of the Day
- "List of Time Data Manipulation Services" on page 108.

/etc/profile File

The first file that the operating system uses at login time is the **/etc/profile** file. This file controls system-wide default variables such as:
- Export variables
- File creation mask (umask)
- Terminal types
- Mail messages to indicate when new mail has arrived.

The system administrator configures the **profile** file for all users on the system. Only the system administrator can change this file.

.profile File

The second file that the operating system uses at login time is the **.profile** file. The **.profile** file is present in your home (**$HOME**) directory and enables you to customize your individual working environment. The **.profile** file also overrides commands and variables set in the **/etc/profile** file. Since the **.profile** file is hidden, use the **ls -a** command to list it. Use the **.profile** file to control the following defaults:
- Shells to open
- Prompt appearance
- Environment variables (for example, search path variables)
- Keyboard sound

The following example shows a typical **.profile** file:

```
PATH=/usr/bin:/etc:/home/bin1:/usr/lpp/tps4.0/user:/home/gsc/bin::
epath=/home/gsc/e3:
export PATH epath
csh
```

This example has defined two paths (`PATH` and `epath`), exported them, and opened a C shell (`csh`).

You can also use the **.profile** file (or if it is not present, the **profile** file) to determine login shell variables. You can also customize other shell environments. For example, use the **.chsrc** and **.kshrc** files to tailor a C shell and a Korn shell, respectively, when each type shell is started.

List of Time Data Manipulation Services

The time functions access and reformat the current system date and time. You do not need to specify any special flag to the compiler to use the time functions.

Include the header file for these functions in the program. To include a header file, use the following statement:

```
#include <time.h>
```

The time services are the following:

adjtime	Corrects the time to allow synchronization of the system clock.
ctime, **localtime**, **gmtime**, **mktime**, **difftime**, **asctime**, **tzset**	Converts date and time to string representation.
getinterval, **incinterval**, **absinterval**, **resinc**, **resabs**, **alarm**, **ualarm**, **getitimer**, **setitimer**	Manipulates the expiration time of interval timers.
gettimer, **settimer**, **restimer**, **stime**, **time**	Gets or sets the current value for the specified systemwide timer.
gettimerid	Allocates a per-process interval timer.
gettimeofday, **settimeofday**, **ftime**	Gets and sets date and time.
nsleep, **usleep**, **sleep**	Suspends a current process from running.
reltimerid	Releases a previously allocated interval timer.

Enabling Dynamic Processor Deallocation

Starting with machine type 7044 model 270, the hardware of all systems with more than two processors is able to detect correctable errors, which are gathered by the firmware. These errors are not fatal and, as long as they remain rare occurrences, can be safely ignored. However, when a pattern of failures seems to be developing on a specific processor, this pattern might indicate that this component is likely to exhibit a fatal failure in the near future. This prediction is made by the firmware based-on-failure rates and threshold analysis.

This operating system, on these systems, implements continuous hardware surveillance and regularly polls the firmware for hardware errors. When the number of processor errors hits a threshold and the firmware recognizes that there is a distinct probability that this system component will fail, the firmware returns an error report. In all cases, the error is logged in the system error log. In addition, on multiprocessor systems, depending on the type of failure, this operating system attempts to stop using the untrustworthy processor and deallocate it. This feature is called *Dynamic Processor Deallocation*.

At this point, the processor is also flagged by the firmware for persistent deallocation for subsequent reboots, until maintenance personnel replaces the processor.

Potential Impact to Applications

This processor decallocation is transparent for the vast majority of applications, including drivers and kernel extensions. However, you can use the published interfaces to determine whether an application or kernel extension is running on a multiprocessor machine, find out how many processors there are, and bind threads to specific processors.

The interface for binding processes or threads to processors uses logical CPU numbers. The logical CPU numbers are in the range [0..*N*-1] where *N* is the total number of CPUs. To avoid breaking applications or kernel extensions that assume no ″holes″ in the CPU numbering, this operating system always makes it appear for applications as if it is the ″last″ (highest numbered) logical CPU to be deallocated. For instance, on an 8-way SMP, the logical CPU numbers are [0..7]. If one processor is deallocated, the total number of available CPUs becomes 7, and they are numbered [0..6]. Externally, it looks like CPU 7 has disappeared, regardless of which physical processor failed. In the rest of this description, the term CPU is used for the logical entity and the term processor for the physical entity.

Applications or kernel extensions using processes/threads binding could potentially be broken if this operating system silently terminated their bound threads or forcefully moved them to another CPU when one of the processors needs to be deallocated. Dynamic Processor Deallocation provides programming interfaces so that those applications and kernel extensions can be notified that a processor deallocation is about to happen. When these applications and kernel extensions get this notification, they are responsiblefor moving their bound threads and associated resources (such as timer request blocks) away form the last logical CPU and adapt themselves to the new CPU configuration.

If, after notification of applications and kernel extensions, some of the threads are still bound to the last logical CPU, the deallocation is aborted. In this case, the fact that the deallocation has been aborted is logged in the error log and continues using the ailing processor. When the processor ultimately fails, it creates a total system failure. Thus, it is important for applications or kernel extensions binding threads to CPUs to get the notification of an impending processor deallocation, and act on this notice.

Even in the rare cases where the deallocation cannot go through, Dynamic Processor Deallocation still gives advanced warning to system administrators. By recording the error in the error log, it gives them a chance to schedule a maintenance operation on the system to replace the ailing component before a global system failure occurs.

Processor Deallocation:

The typical flow of events for processor deallocation is as follows:

1. The firmware detects that a recoverable error threshold has been reached by one of the processors.
2. The firmware error report is logged in the system error log, and, when executing on a machine supporting processor deallocation, start the deallocation process.
3. This operating system notifies non-kernel processes and threads bound to the last logical CPU.
4. This operating system waits for all the bound threads to move away from the last logical CPU. If threads remain bound, the operating system eventually times out (after ten minutes)and aborts the deallocation
5. Otherwise, the previously registered High Availability Event Handlers (HAEHs) is involked. An HAEH might return an error that aborts the deallocation.
6. Otherwise, the deallocation process continues and ultimately stops the failing processor.

In case of failure at any point of the deallocation, the failure is logged with the reason why the deallocation was aborted. The system administrator can look at the error log, take corrective action (when possible) and restart the deallocation. For instance, if the deallocation was aborted because at least one application did not unbind its bound threads, the system administrator could stop the application(s), restart the deallocation (which should go through this time) and restart the application.

System Administration

Turning Processor Deallocation On and Off

Dynamic Processor Deallocation can be enabled or disabled by changing the value of the **cpuguard** attribute of the ODM object ***sys0***. The possible values for the attribute are **enable** and **disable**.

The default, is that the dynamic processor deallocation is disabled (the attribute **cpuguard** has a value of **disable**). System administrators who want to take advantage of this feature must enable it using either the Web-based System Manager system menus, the SMIT System Environments menu, or the **chdev** command.

> **Note:** If processor deallocation is turned off, the errors are still reported in the error log and you will see the error indicating that the operating system was notified of the problem with a CPU (CPU_FAILURE_PREDICTED, see the following format).

Restarting an Aborted Processor Deallocation

Sometimes the processor deallocation fails because an application did not move its bound threads away from the last logical CPU. Once this problem has been fixed, by either unbinding (when it is safe to do so) or stopping the application, the system administrator can restart the processor deallocation process using the **ha_star** command.

The syntax for this command is:

```
ha_star -C
```

where -C is for a CPU predictive failure event.

Processor State Considerations

Physical processors are represented in the ODM data base by objects named **proc*n*** where *n* is the physical processor number (*n* is a decimal number). Like any other ″device″ represented in the ODM database, processor objects have a state (Defined/Available) and attributes.

The state of a **proc** object is always Available as long as the corresponding processor is present, regardless of whether it is usable. The **state** attribute of a **proc** object indicates if the processor is used and, if not, the reason. This attribute can have three values:

enable	The processor is used.
disable	The processor has been dynamically deallocatd.
faulty	The processor was declared defective by the firmware at startup time.

In the case of CPU errors, if a processor for which the firmware reports a predictive failure is successfully deallocated, its state goes from enable to disable. Independently of of this operating system, this processor is also flagged as defective in the firmware. Upon reboot, it is not available and will have its state set to faulty. But the ODM **proc** object is still marked Available. Only if the defective CPU was physically removed from the system board or CPU board (if it were at all possible) is the **proc** object change to Defined.

Examples:

Processor **proc4** is working correctly and used by the operating system:

```
# lsattr -EH -l proc4
attribute value description user_settable

state enable Processor state False
type PowerPC_RS64-III Processor type False
#
```

Processor **proc4** gets a predictive failure and gets deallocated by the operating system:

```
# lsattr -EH -l proc4
attribute value description user_settable

state disable Processor state False
type PowerPC_RS64-III Processor type False
#
```

At the next system restart, processor **proc4** is reported by firmware as defective and not available to the operating system:

```
# lsattr -EH -l proc4
attribute value description user_settable

state faulty Processor state False
type PowerPC_RS64-III Processor type False
#
```

But in all three cases, the status of processor **proc4** is Available:

```
# lsdev -CH -l proc4
name status location description

proc4 Available 00-04 Processor
#
```

Error Log Entries

The following are examples wth descriptions of error logentries:

errpt short format - summary

Three different error log messages are associated with CPU deallocation. The following is an example of entries displayed by the **errpt** command (without options):

```
# errpt
IDENTIFIER      TIMESTAMP       T     C     RESOURCE_NAME     DESCRIPTION
804E987A        1008161399      I     O     proc4             CPU DEALLOCATED
8470267F        1008161299      T     S     proc4             CPU DEALLOCATION ABORTED
1B963892        1008160299      P     H     proc4             CPU FAILURE PREDICTED
#
```

- If processor deallocation is enabled, a `CPU FAILURE PREDICTED` message is always followed by either a `CPU DEALLOCATED` message or a `CPU DEALLOCATION ABORTED` message.
- If processor deallocation is not enabled, only the `CPU FAILURE PREDICTED` message is logged. Enabling processor deallocation any time after one or more `CPU FAILURE PREDICTED` messages have been logged initiates the deallocation process and results in a success or failure error log entry, as described above, for each processor reported failing.

errpt long format - detailed description

The following is the form of output obtained with **errpt -a**:

- **CPU_FAIL_PREDICTED**

  Error description: Predictive Processor Failure

  This error indicates that the hardware detected that a processor has a high probability to fail in a near future. It is always logged whether or not processor deallocation is enabled.

  DETAIL DATA: Physical processor number, location

  Example: error log entry - long form

  ```
  LABEL: CPU_FAIL_PREDICTED
  IDENTIFIER: 1655419A

  Date/Time: Thu Sep 30 13:42:11
  Sequence Number: 53
  Machine Id: 00002F0E4C00
  Node Id: auntbea
  Class: H
  Type: PEND
  ```

```
Resource Name: proc25
Resource Class: processor
Resource Type: proc_rspc
Location: 00-25

Description
CPU FAILURE PREDICTED

Probable Causes
CPU FAILURE

Failure Causes
CPU FAILURE

Recommended Actions
ENSURE CPU GARD MODE IS ENABLED
RUN SYSTEM DIAGNOSTICS.

Detail Data
PROBLEM DATA
0144 1000 0000 003A 8E00 9100 1842 1100 1999 0930 4019
0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 4942 4D00 5531
2E31 2D50 312D 4332 0000
0002 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000
... ... ... ... ...
```

- **CPU_DEALLOC_SUCCESS**

  Error Description: A processor has been successfully deallocated upon detection of a predictive processor failure.

  This message is logged when processor deallocation is enabled, and when the CPU has been successfully deallocated.

  DETAIL DATA: Logical CPU number of deallocated processor.

  Example: error log entry - long form:

```
LABEL: CPU_DEALLOC_SUCCESS
IDENTIFIER: 804E987A

Date/Time: Thu Sep 30 13:44:13
Sequence Number: 63
Machine Id: 00002F0E4C00
Node Id: auntbea
Class: O
Type: INFO
Resource Name: proc24

Description
CPU DEALLOCATED


Recommended Actions
MAINTENANCE IS REQUIRED BECAUSE OF CPU FAILURE

Detail Data
LOGICAL DEALLOCATED CPU NUMBER

0
```

  The preceding example shows that **proc24** was successfully deallocated and was logical CPU **0** when the failure occurred.

- **CPU_DEALLOC_FAIL**

Error Description: A processor deallocation, due to a predictive processor failure, was not successful.

This message is logged when CPU deallocation is enabled, and when the CPU has not been successfully deallocated.

DETAIL DATA: Reason code, logical CPU number, additional information depending of the type of failure.

The reason code is a numeric hexadecimal value. The possible reason codes are:

2	One or more processes/threads remain bound to the last logical CPU. In this case, the detailed data give the PIDs of the offending processes.
3	A registered driver or kernel extension returned an error when notified. In this case, the detailed data field contains the name of the offending driver or kernel extension (ASCII encoded).
4	Deallocating a processor causes the machine to have less than two available CPUs. This operating system does not deallocate more than *N*-2 processors on an *N*-way machine to avoid confusing applications or kernel extensions using the total number of available processors to determine whether they are running on a Uni Processor (UP) system where it is safe to skip the use of multiprocessor locks, or a Symmetric Multi Processor (SMP).
200 (0xC8)	Processor deallocation is disabled (the ODM attribute **cpuguard** has a value of **disable**). You normally do not see this error unless you start **ha_star** manually.

Examples: error log entries - long format

Example 1:

```
LABEL: CPU_DEALLOC_ABORTED
IDENTIFIER: 8470267F
Date/Time: Thu Sep 30 13:41:10
Sequence Number: 50
Machine Id: 00002F0E4C00
Node Id: auntbea
Class: S
Type: TEMP
Resource Name: proc26

Description
CPU DEALLOCATION ABORTED

Probable Causes
SOFTWARE PROGRAM

Failure Causes
SOFTWARE PROGRAM

Recommended Actions
MAINTENANCE IS REQUIRED BECAUSE OF CPU FAILURE
SEE USER DOCUMENTATION FOR CPU GARD

Detail Data
DEALLOCATION ABORTED CAUSE
0000 0003
DEALLOCATION ABORTED DATA
6676 6861 6568 3200
```

The preceding example shows that the deallocation for **proc26** failed. The reason code **3** means that a kernel extension returned an error to the kernel notification routine. The `DEALLOCATION ABORTED DATA` above spells **fvhaeh2**, which is the name the extension used when registering with the kernel.

Example 2:

```
LABEL: CPU_DEALLOC_ABORTED
IDENTIFIER: 8470267F
Date/Time: Thu Sep 30 14:00:22
Sequence Number: 71
Machine Id: 00002F0E4C00
Node Id: auntbea
Class: S
Type: TEMP
Resource Name: proc19

Description
CPU DEALLOCATION ABORTED

Probable Causes
SOFTWARE PROGRAM

Failure Causes
SOFTWARE PROGRAM

Recommended Actions
MAINTENANCE IS REQUIRED BECAUSE OF CPU FAILURE;
SEE USER DOCUMENTATION FOR CPU GARD

Detail Data
DEALLOCATION ABORTED CAUSE
0000 0002
DEALLOCATION ABORTED DATA
0000 0000 0000 4F4A
```

The preceding example shows that the deallocation for **proc19** failed. The reason code **2** indicates thread(s) were bound to the last logical processor and did not unbind upon receiving the `SIGCPUFAIL` signal. The `DEALLOCATION ABORTED DATA` shows that these threads belonged to process **0x4F4A**.

Options of the **ps** command (`-o THREAD`, `-o BND`) allow listings of all threads or processes, with the number of the CPU they are bound to when applicable.

Example 3:

```
LABEL: CPU_DEALLOC_ABORTED
IDENTIFIER: 8470267F

Date/Time: Thu Sep 30 14:37:34
Sequence Number: 106
Machine Id: 00002F0E4C00
Node Id: auntbea
Class: S
Type: TEMP
Resource Name: proc2

Description
CPU DEALLOCATION ABORTED

Probable Causes
SOFTWARE PROGRAM

Failure Causes
SOFTWARE PROGRAM

Recommended Actions
MAINTENANCE IS REQUIRED BECAUSE OF CPU FAILURE
SEE USER DOCUMENTATION FOR CPU GARD

Detail Data
```

```
DEALLOCATION ABORTED CAUSE
0000 0004
DEALLOCATION ABORTED DATA
0000 0000 0000 0000
```

The preceding example shows that the deallocation of **proc2** failed because there were two or fewer active processors at the time of failure (reason code **4**).

Chapter 11. National Language Support

Many system variables are used to establish the language environment of the system. These variables and their supporting commands, files, and other tools, are referred to as National Language Support (NLS).

Topics covered in this chapter are:

- “National Language Support Overview”
- “Locale Overview” on page 118
- “Understanding Locale” on page 119
- “Understanding Locale Categories” on page 123
- “Understanding Locale Environment Variables” on page 123
- “Understanding the Locale Definition Source File” on page 124
- “Understanding the Character Set Description (charmap) Source File” on page 125
- “Changing Your Locale” on page 125
- “Converters Overview” on page 126.

National Language Support Overview

National Language Support (NLS) provides commands and Standard C Library subroutines for a single worldwide system base. An internationalized system has no built-in assumptions or dependencies on language-specific or cultural-specific conventions such as:

- Code sets
- Character classifications
- Character comparison rules
- Character collation order
- Numeric and monetary formatting
- Date and time formatting
- Message-text language.

All information pertaining to cultural conventions and language is obtained at process run time.

The following capabilities are provided by NLS to maintain a system running in an international environment:

- “Localization of Information”
- “Separation of Messages from Programs” on page 118
- “Conversion between Code Sets” on page 118

Localization of Information

An internationalized system processes information correctly for different locations. For example, in the United States, the date format, 9/6/1990, is interpreted to mean the sixth day of the ninth month of the year 1990. The United Kingdom interprets the same date format to mean the ninth day of the sixth month of the year 1990. The formatting of numerical and monetary data is also country-specific, for example, the U.S. dollar and the U.K. pound. A *locale* is defined by these language-specific and cultural-specific conventions for processing information.

All locale information must be accessible to programs at run time so that data is processed and displayed correctly for your cultural conventions and language. This process is called localization. Localization consists of developing a database containing locale-specific rules for formatting data and an interface to obtain the rules. For more information about localization, see “Locale Overview”.

Separation of Messages from Programs

To facilitate translations of messages into various languages and to make the translated messages available to the program based on a user’s locale, it is necessary to keep messages separate from the programs and provide them in the form of message catalogs that a program can access at run time. To aid in this task, commands and subroutines are provided by the message facility. For more information, see ″Message Facility Overview″.

Conversion between Code Sets

A *character* is any symbol used for the organization, control, or representation of data. A group of such symbols used to describe a particular language make up a *character set*. A code set contains the encoding values for a character set. It is the encoding values in a code set that provide the interface between the system and its input and output devices.

Historically, the effort was directed at encoding the English alphabet. It was sufficient to use a 7-bit encoding method for this purpose because the number of English characters is not large. To support larger alphabets, such as the Asian languages (for example, Chinese, Japanese, and Korean), additional code sets were developed that contained multibyte encodings.

The following code sets are supported:

- Industry-standard code sets are provided by means of the ISO8859 family of code sets, which provide a range of single-byte code set support that includes Latin-1, Latin-2, Arabic, Cyrillic, Hebrew, Greek, and Turkish. The IBM-eucJP code set is the industry-standard code set used to support the Japanese locale.
- Personal Computer (PC) based code sets IBM-850 and IBM-943 (and IBM-932) are supported. IBM-850 is a single-byte code set used to support Latin-1 countries (U.S., Canada, and Western Europe). IBM-943 and IBM-932 are multibyte code sets used to support the Japanese locale.
- A Unicode environment based on the UTF-8 codeset is supported for all supported language/territories. UTF-8 provides character support for most of the major languages of the world and can be used in environments where multiple languages must be processed simultaneously.

As more code sets are supported, it becomes important not to clutter programs with the knowledge of any particular code set. This is known as *code set independence.* To aid in code set independence, NLS supplies converters that translate character encoding values found in different code sets. Using these converters, a system can accurately process data generated in different code set environments. For more information, see “Converters Overview” on page 126.

Locale Overview

An internationalized system has no built-in assumptions or dependencies on code set, character classification, character comparison rules, character collation order, monetary formatting, numeric punctuation, date and time formatting, or the text of messages. A *locale* is defined by these language and cultural conventions. All of the information pertaining to language-specific and cultural-specific conventions is obtained at process run time.

All locale information must be accessible to programs at run time so that data is processed and displayed correctly for your language-specific and cultural-specific conventions. The National Language Support (NLS) system provides these localization capabilities. NLS provides a database containing locale-specific rules for formatting data and an interface to obtain these rules.

Understanding Locale

A locale is made up of the language, territory, and code set combination used to identify a set of language conventions. These conventions include information on collation, case conversion, and character classification, the language of message catalogs, date-and-time representation, the monetary symbol, and numeric representation.

Locale information contained in locale definition source files must first be converted into a locale database by the **localedef** command. The **setlocale** subroutine can then access this information and set locale information for applications. To deal with locale data in a logical manner, locale definition source files are divided into six categories (see “Understanding Locale Categories” on page 123). Each category contains a specific aspect of the locale data. The **LC_*** environment variables and the **LANG** environment variable can be used in specifying the desired locale.

Locale Naming Conventions

Each locale is named by its locale definition source file name. These files are named for the language, territory, and code set information they describe. The following format is used for naming a locale definition file:

```
language[_territory][.codeset][@modifier]
```

For example, the locale for the Danish language spoken in Denmark using the ISO8859-1 code set is `da_DK.IS08859-1`. The `da` stands for the Danish language and the `DK` stands for Denmark. The short form of `da_DK` is sufficient to indicate this locale. The same language and territory using the IBM-850 code set is indicated by either `Da_DK.IBM-850` or the short form `Da_DK`.

System-defined locale definition files are provided to show the format of locale categories and their keywords. The **/usr/lib/nls/loc** directory contains the locale definition files for system-defined locales. The C, or POSIX, locale defines the ANSI C-defined standard locale inherited by all processes at startup time. The other system-defined locale definition source files are:

Locale	Language	Country	Code Set
Ar_AA	Arabic	Arabic Countries	IBM-1046
ar_AA	Arabic	Arabic Countries	IS08859-6
ar_AE	Arabic	United Arab Emirates	ISO8859-6
ar_BH	Arabic	Bahrain	ISO8859-6
ar_EG	Arabic	Egypt	ISO8859-6
ar_JO	Arabic	Jordan	ISO8859-6
ar_KW	Arabic	Kuwait	ISO8859-6
ar_LB	Arabic	Lebanon	ISO8859-6
ar_OM	Arabic	Oman	ISO8859-6
ar_QA	Arabic	Qatar	ISO8859-6
ar_SA	Arabic	Saudi Arabia	ISO8859-6
ar_SY	Arabic	Syria	ISO8859-6
ar_TN	Arabic	Tunisia	ISO8859-6
be_BY	Byelorussian	Belarus	ISO8859-5
bg_BG	Bulgarian	Bulgaria	ISO8859-5
ca_ES	Catalan	Spain	IS08859-1
ca_ES	Catalan	Spain	ISO8859-15

Locale	Language	Country	Code Set
cs_CZ	Czech	Czech Republic	ISO8859-2
Da_DK	Danish	Denmark	IBM-850
da_DK	Danish	Denmark	ISO8859-1
da_DK	Danish	Denmark	ISO8859-15
de_AT	German	Austria	ISO8859-15
De_CH	German	Switzerland	IBM-850
de_CH	German	Switzerland	ISO8859-1
de_CH	German	Switzerland	ISO8859-15
De_DE	German	Germany	IBM-850
de_DE	German	Germany	ISO8859-1
de_DE	German	Germany	ISO8859-15
de_LU	German	Luxembourg	ISO8859-15
el_GR	Greek	Greece	ISO8859-7
en_AU	English	Australia	ISO8859-15
en_BE	English	Belgium	ISO8859-15
en_CA	English	Canada	ISO8859-15
En_GB	English	Great Britain	IBM-850
en_GB	English	Great Britain	ISO8859-1
en_GB	English	Great Britain	ISO8859-15
en_IE	English	Ireland	ISO8859-15
en_IN	English	India	ISO8859-15
en_NZ	English	New Zealand	ISO8859-15
En_US	English	United States	IBM-850
en_US	English	United States	ISO8859-1
en_US	English	United States	ISO8859-15
en_ZA	English	South Africa	ISO8859-15
es_AR	Spanish	Argentina	ISO8859-15
es_CL	Spanish	Chile	ISO8859-15
es_CO	Spanish	Columbia	ISO8859-15
Es_ES	Spanish	Spain	IBM-850
es_ES	Spanish	Spain	ISO8859-1
es_ES	Spanish	Spain	ISO8859-15
es_MX	Spanish	Mexico	ISO8859-15
es_PE	Spanish	Peru	ISO8859-15
es_PR	Spanish	Puerto Rico	ISO8859-15
es_UY	Spanish	Uruguay	ISO8859-15
es_VE	Spanish	Venezuela	ISO8859-15
Et_EE	Estonian	Estonia	IBM-922
ET_EE	Estonian	Estonia	UTF-8
Fi_FI	Finnish	Finland	IBM-850
fi_FI	Finnish	Finland	ISO8859-1

Locale	Language	Country	Code Set
fi_FI	Finnish	Finland	ISO8859-15
Fr_BE	French	Belgium	IBM-850
fr_BE	French	Belgium	ISO8859-1
fr_BE	French	Belgium	ISO8859-15
Fr_CA	French	Canada	IBM-850
fr_CA	French	Canada	ISO8859-1
fr_CA	French	Canada	ISO8859-15
Fr_FR	French	France	IBM-850
fr_FR	French	France	ISO8859-1
fr_FR	French	France	ISO8859-15
fr_LU	French	Luxembourg	ISO8859-15
Fr_CH	French	Switzerland	IBM-850
fr_CH	French	Switzerland	ISO8859-1
fr_CH	French	Switzerland	ISO8859-15
HI_IN	Hindi	India	UTF-8
hr_HR	Croatian	Croatia	ISO8859-2
hu_HU	Hungarian	Hungary	ISO8859-2
Is_IS	Icelandic	Iceland	IBM-850
is_IS	Icelandic	Iceland	ISO8859-1
is_IS	Icelandic	Iceland	ISO8859-15
it_CH	Italian	Switzerland	ISO8859-15
It_IT	Italian	Italy	IBM-850
it_IT	Italian	Italy	ISO8859-1
it_IT	Italian	Italy	ISO8859-15
Iw_IL	Hebrew	Israel	IBM-856
iw_IL	Hebrew	Israel	ISO8859-8
Ja_JP	Japanese	Japan	IBM-943
ja_JP	Japanese	Japan	IBM-eucJP
ko_KR	Korean	Korea	IBM-eucKR
Lt_LT	Lithuanian	Lithuania	IBM-921
LT_LT	Lithuanian	Lithuania	UTF-8
Lv_LV	Latvian	Latvia	IBM-921
LV_LV	Latvian	Latvia	UTF-8
mk_MK	Macedonian	Former Yugoslav Republic of Macedonia	ISO8859-5
Nl_BE	Dutch	Belgium	IBM-850
nl_BE	Dutch	Belgium	ISO8859-1
nl_BE	Dutch	Belgium	ISO8859-15
Nl_NL	Dutch	Netherlands	IBM-850
nl_NL	Dutch	Netherlands	ISO8859-1
nl_NL	Dutch	Netherlands	ISO8859-15
No_NO	Norwegian	Norway	IBM-850

Locale	Language	Country	Code Set
no_NO	Norwegian	Norway	ISO8859-1
no_NO	Norwegian	Norway	ISO8859-15
pl_PL	Polish	Poland	ISO8859-2
pt_BR	Portuguese	Brazil	ISO8859-1
pt_BR	Portuguese	Brazil	ISO8859-15
Pt_PT	Portuguese	Portugal	IBM-850
pt_PT	Portuguese	Portugal	ISO8859-1
pt_PT	Portuguese	Portugal	ISO8859-15
ro_RO	Romanian	Romania	ISO8859-2
ru_RU	Russian	Russia	ISO8859-5
sh_SP	Serbian (Latin)	Yugoslavia	ISO8859-2
sh_YU	Serbian (Latin)	Yugoslavia	ISO8859-2
sl_SI	Slovene	Slovenia	ISO8859-2
sk_SK	Slovak	Slovakia	ISO8859-2
sq_AL	Albanian	Albania	ISO8859-1
sq_AL	Albanian	Albania	ISO8859-15
sr_SP	Serbian (Cyrillic)	Yugoslavia	ISO8859-5
sr_YU	Serbian (Cyrillic)	Yugoslavia	ISO8859-5
Sv_SE	Swedish	Sweden	IBM-850
sv_SE	Swedish	Sweden	ISO8859-1
sv_SE	Swedish	Sweden	ISO8859-15
th_TH	Thai	Thailand	TIS-620
TH_TH	Thai	Thailand	UTF-8
tr_TR	Turkish	Turkey	ISO8859-9
Uk_UA	Ukrainian	Ukraine	IBM-1124
Vi_VN	Vietnamese	Vietnam	IBM-1129
VI_VN	Vietnamese	Vietnam	UTF-8
Zh_CN	Simplified Chinese	People's Republic of China	GBK
zh_CN	Simplified Chinese	People's Republic of China	IBM-eucCN
ZH_CN	Chinese	People's Republic of China	UTF-8
zh_TW	Chinese (trad)	Republic of China	IBM-eucTW
Zh_TW	Chinese (trad)	Republic of China	big5

Installation Default Locale

The installation default locale refers to the locale selected at installation. For example, when prompted, a user can specify the French language as spoken in Canada during the installation process. The code set automatically defaults to the ISO8859-1 code set. With this information, the system sets the value of the default locale, specified by the **LANG** environment variable, to `fr_CA` (`fr` for ISO8859-1 French and `CA` for Canada). Every process uses this locale unless the **LC_*** or **LANG** environment variables are modified. The default locale can be changed by using the Manage Language Environment menu in the "System Management Interface Tool (SMIT) Overview" on page 167.

Understanding Locale Categories

A locale *category* is a particular grouping of language-specific and cultural-convention-specific data. For instance, data referring to date-and-time formatting, the names of the days of the week, names of the months, and other time-specific information is grouped into the **LC_TIME** category. Each category uses a set of keywords that describe the particulars of that locale subset.

The following standard categories can be defined in a locale definition source file:

LC_COLLATE	Defines character-collation or string-collation information
LC_CTYPE	Defines character classification, case conversion, and other character attributes
LC_MESSAGES	Defines the format for affirmative and negative responses
LC_MONETARY	Defines rules and symbols for formatting monetary numeric information
LC_NUMERIC	Defines rules and symbols for formatting nonmonetary numeric information
LC_TIME	Defines a list of rules and symbols for formatting time and date information.

Note: Locale categories can only be modified by editing the locale definition source file. Do not confuse them with the environment variables of the same name, which can be set from the command line.

Understanding Locale Environment Variables

National Language Support (NLS) uses several environment variables to influence the selection of locales. You can set the values of these variables to change search paths for locale information:

LANG	Specifies the installation default locale. **Note:** The **LANG** value is established at installation. (This is the locale every process uses unless the **LC_*** environment variables are set). The **LANG** environment variable can be changed by using the “System Management Interface Tool (SMIT) Overview” on page 167. The C and POSIX locales are designed to offer the best performance.
LC_ALL	Overrides the value of the **LANG** environment variable and the values of any other **LC_*** environment variables.
LC_COLLATE	Specifies the locale to use for **LC_COLLATE** category information. The **LC_COLLATE** category determines character-collation or string-collation rules governing the behavior of ranges, equivalence classes, and multicharacter collating elements.
LC_CTYPE	Specifies the locale to use for **LC_CTYPE** category information. The **LC_CTYPE** category determines character handling rules governing the interpretation of sequences of bytes of text data characters (that is, single-byte versus multibyte characters), the classification of characters (for example, alpha, digit, and so on), and the behavior of character classes.
LC__FASTMSG	Specifies that default messages are used for the C and POSIX locales and that **NLSPATH** are ignored when **LC__FASTMSG** is set to `true`; otherwise, POSIX compliant message handling will be performed. The default value will be `LC__FASTMSG=true` in **/etc/environment**.
LC_MESSAGES	Specifies the locale to use for **LC_MESSAGES** category information. The **LC_MESSAGES** category determines rules governing affirmative and negative responses and the locale (language) for messages and menus.
LC_MONETARY	Specifies the locale to use for **LC_MONETARY** category information. The **LC_MONETARY** category determines the rules governing monetary-related formatting.
LC_NUMERIC	Specifies the locale to use for **LC_NUMERIC** category information. The **LC_NUMERIC** category determines the rules governing nonmonetary numeric formatting.
LC_TIME	Specifies the locale to use for **LC_TIME** category information. The **LC_TIME** category determines the rules governing date and time formatting.

LOCPATH	Specifies the search path for localized information, including binary locale files, input methods, and code-set converters. **Note:** All **setuid** and **setgid** programs ignore the **LOCPATH** environment variable.
NLSPATH	Specifies the search path for locating message catalog files. This environment variable is used by the Message Facility component of the NLS subsystem. See the **catopen** subroutine for more information about expected format of the **NLSPATH** variable.

The environment variables that affect locale selection can be grouped into three priority classes:

Locale Environment Variable Hierarchy	
Priority Class	**Environment Variables**
High	**LC_ALL**
	LC_COLLATE
	LC_CTYPE
Medium	**LC_MESSAGES**
	LC_MONETARY
	LC_NUMERIC
	LC_TIME
Low	**LANG**

The behavior of an internationalized program is affected by the locale environment variables in the following manner:

- If the **LC_ALL** environment variable is set, the value of the **LC_ALL** variable is used for all categories. For example, if the **LC_ALL** environment variable is equal to `en_US` and the **LANG** environment variable is equal to `fr_FR`, the locale is set to `en_US`.
- If the **LC_ALL** environment variable is not set, the values specified for medium-priority environment variables are used. For example, if the **LANG** environment variable is set to `en_US` and the **LC_TIME** environment variable is set to `fr_FR`, then the **LC_TIME** category is loaded from the `fr_FR` locale database. The **LC_TIME** environment variable does not affect the behavior of any other category.
- If individual **LC_*** environment variables are not set, the value of the **LANG** environment variable specifies the locale for all remaining categories.
- If the **LANG** variable is not set, the locale for all remaining categories defaults to the C locale.

Understanding the Locale Definition Source File

Unlike environment variables, which can be set from the command line, locales can only be modified by editing and compiling a locale definition source file.

If a desired locale is not part of the library, a binary version of the locale can be compiled by the **localedef** command. Locale behavior of programs is not affected by a locale definition source file unless the file is first converted by the **localedef** command, and the locale object is made available to the program. The **localedef** command converts source files containing definitions of locales into a run-time format and copies the run-time version to the file specified on the command line, which usually is a locale name. Internationalized commands and subroutines can then access the locale information. For information on preparing source files to be converted by the **localedef** command, see ″Locale Definition Source File Format″ in *AIX 5L Version 5.1 Files Reference*.

Understanding the Character Set Description (charmap) Source File

Using the character set description (charmap) source file, you can assign symbolic names to character encodings.

Developers of character set description (charmap) source files can choose their own symbolic names, provided that these names do not conflict with the standardized symbolic names that describe the portable character set.

The charmap file resolves problems with the portability of sources, especially locale definition sources. The standardized portable character set is constant across all locales. The charmap file provides the capability to define a common locale definition for multiple code sets. That is, the same locale definition source can be used for code sets with different encodings of the same extended characters.

A charmap file defines a set of symbols that are used by the locale definition source file to refer to character encodings. The characters in the portable character set can optionally be included in the charmap file, but the encodings for these characters should not differ from their default encodings.

The charmap files are located in the **/usr/lib/nls/charmap** directory.

Changing Your Locale

Changing the NLS Environment

You can change the NLS environment using the Web-based System Manager Users application or the SMIT Manage Language Environment menu to:

- Change the default language environment
- Change the keyboard map for the next system restart
- Manage fonts
- Convert the code set of message catalogs
- Convert the code set of flat text files.

You can also use the **setmaps** command to set the code set map of a terminal.

Changing the Default Language Environment

The setting of the **LANG** environment variable (the ″**LANG = <name>**″ string in the **/etc/environment** file) designates the default locale, which is a language-territory-code-set combination. The default locale provides formats for default collation, character classification, case conversion, numeric and monetary formatting, date-and-time formatting, and affirmative or negative responses. The default locale includes reference to the code set.

Changing the Default Keyboard Mapping for the Next System Restart

If more than one code set is supported for a given language-territory combination, multiple low-function terminal (LFT) keyboard mappings exist. The selected keyboard mapping must match the code set of the selected language environment.

Managing Fonts

Users can perform such tasks as selecting the active font and selecting the font to load for the next system restart. The selected font must support the same code set as the selected language environment and LFT keyboard mapping.

Converting the Code Set of Message Catalogs

Message catalogs are shipped in one code set for each translated language-territory combination. The code set of the message catalog must match the code set of the locale.

Converting the Code Set of Flat Text Files

User-defined flat files of one code set can be converted to another code set when appropriate (IBM-850 to ISO8859-1, for example).

Typical User Scenarios

There are several NLS-related scenarios some users might encounter on the system. This section lists common scenarios with suggested actions to be taken.

- User keeps default code set

  The user might be satisfied with the default code set for language-territory combinations even where more than one code set is supported. The user might keep the default code set if the current user environment uses that code set, or if the user is new and has no code set preference.

  The language-territory selected at system installation time is defaulted to the appropriate locale based on the default code set. The default keyboard mappings, default font, and message catalogs are all established around the default code set. This scenario requires no special action from the user.
- User changes code set from the default code set

  Users of a Latin-1 or Japanese locale might want to migrate their data and NLS environment to a different (nondefault) code set. This can be done in the following fashion:
  - When the user has existing data that requires conversion

    Flat text files that require conversion to the preferred code set can be converted through the Web-based System Manager Users application, the SMIT Manage the Language Environment menu, or the **iconv** utility. User-defined structured files require conversion through user-written conversion tools that use the **iconv** library functions to convert the desired text fields within the structured files.
  - When the user wants to change to the other code set

    Where more than one code set is supported for a language-territory combination, the user may change to a nondefault locale by using:
    - The Web-based System Manager Users application
    - The SMIT Manage Language Environment menu
    - The **chlang**, **chkbd**, and **chfont** commands.

Converters Overview

National Language Support (NLS) provides a base for internationalization to allow data to be changed from one code set to another. You might need to convert text files or message catalogs. There are several standard converters for this purpose.

Understanding iconv Libraries

The iconv facility consists of a set of functions that contain the data and logic to convert from one code set to another. The utility also includes the **iconv** command, which converts data. A single system can have several converters. The **LOCPATH** environment variable determines the converter that the **iconv** subroutines use.

Note: All **setuid** and **setgid** programs ignore the **LOCPATH** environment variable.

Universal UCS Converter

UCS-2 is a universal 16-bit encoding (see the code set overview in *AIX 5L Version 5.1 General Programming Concepts: Writing and Debugging Programs*) that can be used as an interchange medium to provide conversion capability between virtually any code sets. The conversion can be accomplished using the Universal UCS Converter, which converts between any two code sets XXX and YYY as follows:

```
XXX <-> UCS-2 <-> YYY
```

The XXX and YYY conversions must be included in the supported List of UCS-2 Interchange Converters, and must be installed on the system.

The universal converter is installed as the file **/usr/lib/nls/loc/iconv/Universal_UCS_Conv**. A new conversion can be supported by creating a new link with the appropriate name in the **/usr/lib/nls/loc/iconv** directory. For example, to support new converters between IBM-850 and IBM-437, you can execute the following commands:

```
ln -s /usr/lib/nls/loc/iconv/Universal_UCS_Conv
/usr/lib/nls/loc/iconv/IBM-850_IBM-437

ln -s /usr/lib/nls/loc/iconv/Universal_UCS_Conv
/usr/lib/nls/loc/iconv/IBM-437_IBM-850
```

Attention: If a converter link is created for incompatible code sets (for example, ISO8859-1 and IBM-eucJP), and if the source data contains characters that do not exist in the target code set, significant data loss can result.

The conversion between multibyte and wide character code depends on the current locale setting. Do not exchange wide character codes between two processes, unless you have knowledge that each locale that might be used handles wide character codes in a consistent fashion. Most locales for this operating system use the Unicode character value as a wide character code, except locales based on the IBM-850 and IBM-eucTW codesets.

Chapter 12. Process Management

The process is the entity that the operating system uses to control the use of system resources. AIX Version 4 introduces the use of *threads* to control processor-time consumption, but most of the system management tools still require the administrator to refer to the process in which a thread is running, rather than to the thread itself.

See the *AIX 5L Version 5.1 System User's Guide: Operating System and Devices* for basic information on managing your own processes; for example, restarting or stopping a process that you started or scheduling a process for a later time. This guide also defines terms that describe processes, such as daemons and zombies.

For task procedures, see "Chapter 12. Process Management" in the *AIX 5L Version 5.1 System Management Concepts: Operating System and Devices*.

Tools are available to:

- Observe the creation, cancellation, identity, and resource consumption of processes
  - The **ps** command is used to report process IDs, users, CPU-time consumption, and other attributes.
  - The **who -u** command reports the shell process ID of logged-on users.
  - The **svmon** commad is used to report process real-memory consumption. (See *Performance Toolbox Version 2 and 3 for AIX: Guide and Reference* for information on the **svmon** command.)
  - The **acct** command mechanism writes records at process termination summarizing the process's resource use. (See how to set up an accounting system in "Accounting Overview" on page 157 .)
- Control the priority level at which a process contends for the CPU.
  - The **nice** command causes a command to be run with a specified process priority. (See *AIX 5L Version 5.1 System User's Guide: Operating System and Devices.)*
  - The **renice** command changes the priority of a given process.
- Terminate processes that are out of control.
  - The **kill** command sends a termination signal to one or more processes.
- Tune the operating system's process-management mechanisms.
  - The **schedtune** command permits changes to the process scheduler parameters. See *AIX 5L Version 5.1 Performance Management Guide* for information on the **schedtune** command.

Chapter 13. Workload Management

Workload Manager (WLM) is designed to provide the system administrator increased control over how the scheduler and the virtual memory manager (VMM) allocate resources to processes. You can use WLM to prevent different classes of jobs from interfering with each other and to allocate resources based on the requirements of different groups of users.

WLM is primarily intended for use with large systems. Large systems are often used for server consolidation, in which workloads from many different server systems (such as printer, database, general user, and transaction processing systems) are combined into a single large system to reduce the cost of system maintenance. These workloads often interfere with each other and have different goals and service agreements.

WLM also provides isolation between user communities with very different system behaviors. This can prevent effective starvation of workloads with certain behaviors (for example, interactive or low CPU usage jobs) by workloads with other behaviors (for example, batch or high memory usage jobs).

Also, WLM ties into the accounting subsystem (see “Accounting Overview” on page 157) allowing users to do resource usage accounting per WLM class in addition to the standard accounting per user or group.

WLM Concepts

With WLM, you can create different classes of service for jobs, as well as specify attributes for those classes. These attributes specify minimum and maximum amounts of CPU, physical memory, and disk I/O throughput to be allocated to a class. WLM then assigns jobs automatically to classes using class assignment rules provided by a system administrator. These assignment rules are based on the values of a set of attributes for a process. Either the system administrator or a privileged user can also manually assign jobs to classes, overriding the automatic assignment.

Terminology

class	A *class* is a collection of processes and their associated threads. A class has a single set of resource-limitation values and target shares. **class** is used to describe both subclasses and superclasses.
superclass	A *superclass* is a class that has subclasses associated with it. No processes can belong to a superclass without also belonging to a subclass. A superclass has a set of class-assignment rules that determines which processes are assigned to the superclass. A superclass also has a set of resource-limitation values and resource target shares that determines the amount of resources that can be used by processes which belong to the superclass. These resources are divided among the subclasses based on the resources limitation values and resource target shares of the subclasses.

subclasses	A *subclass* is a class associated with exactly one superclass. Every process in a subclass is also a member of its superclass. Subclasses have access only to resources that are available to the superclass. A subclass has a set of class assignment rules that determines which of the processes assigned to the superclass belong to the subclass. A subclass also has a set of resource-limitation values and resource target shares that determines the resources that can be used by processes in the subclass. These resource-limitation values and resource target shares indicate how much of the resources available to the superclass (the target for the superclass) can be used by processes in the subclass. WLM administration can be done using either the Web-based System Manager, SMIT, or the WLM command-line interface.
classification mechanism	A *classification mechanism* is a set of class assignment rules that determines which processes are assigned to which classes (superclasses or subclasses within superclasses).
class assignment rule	A *class assignment rule* indicates which values within a set of process attributes result in a process being assigned to a particular class (superclass or subclass within a superclass).
process attribute value	A *process attribute value* is the value that a process has for a process attribute. The process attributes can include attributes such as user ID, group ID, and application path name.
resource-limitation values	*Resource-limitation values* are a set of values that WLM maintains for a set of resource utilization values. These limits are completely independent of the resource limits specified with the **setrlimit** subroutine.
resource target share	*Resource target shares* are the shares of a resource that are available to a class (subclass or superclass). These shares are used with other class shares (subclass or superclass) at the same level and tier to determine the desired distribution of the resources between classes at that level and tier.
resource-utilization value	A *resource-utilization value* is the amount of a resource that a process or set of processes is currently using in a system. Whether it is one process or a set of processes is determined by the scope of process resource collection.
scope-of-resource collection	The *scope-of-resource collection* is the level at which resource utilization is collected and the level at which resource-limitation values are applied. This might be at the level of each process in a class, the level of the sum across every process in a class owned by each user, or the level of the sum across every process in a class. The only scope currently supported is the latter.
process class properties	The *process class properties* are the set of properties that are given to a process based on the classes (subclass and superclass) to which it is assigned.
class authorizations	The *class authorizations* are a set of rules that indicates which users and groups are allowed to perform operations on a class or processes and threads in a class. This includes the authorization to manually assign processes to a class or to create subclasses of a superclass.
class tier	The *class tier* value is the position of the class within the hierarchy of resource limitation desirability for all classes. The resource limits (including the resource targets) for all classes in a tier are satisfied before any resource is provided to lower tier classes. Tiers are provided at both the superclass and subclass levels. Resources are provided to superclasses based on their tiers. Within a superclass, resources are given to subclasses based on their tier values within the superclass. Thus, superclass tier is the major differentiator in resource distribution; the subclass tier provides an additional smaller differentiator within a superclass.

Classes

WLM allows system administrators to define classes and define for each class a set of attributes and resource limits. The processes are assigned to classes based on criteria provided by the system administrator. The resource entitlements and limits are enforced at the class level. This method of defining classes of service and regulating the resource utilization of each class of applications prevents applications with very different resource use patterns from interfering with each other when they share a single server.

WLM supports a hierarchy of classes with two levels:

- The resources of the system are distributed among superclasses according to the resource entitlements for each superclass. The system administrator defines resource entitlements.
- In turn, each superclass can have subclasses. The resources allocated to the superclass are distributed among the subclasses according to the resource entitlements given to each subclass.
- The system administrator can delegate the administration of the subclasses of each superclass to a *superclass administrator* or to a group of superclass administrators.
- WLM supports 32 superclasses (27 user-defined, plus 5 predefined). Each superclass in turn can have 12 subclasses (10 user-defined and 2 predefined).
- Depending on the needs of the organization, a system administrator can decide to use only superclasses or to use superclasses and subclasses. Another option is to use subclasses only for some of the superclasses.

  Note: Throughout this discussion of WLM, the term *class* applies to both superclasses and subclasses. If discussion applies only to a specific class type, that type is explicitly mentioned.

Process Assignment to Classes

The processes are assigned to a class, using class-assignment rules provided by the system administrator. The classification criteria are based on the value of a set of attributes of the process such as user ID, group ID, name of the application file, type of process, and application tag.

A defined set of rules is used to determine the superclass a process is assigned to. If this superclass has subclasses defined, there is another set of rules for this superclass to determine which subclass is assigned to which process.

This automatic assignment process also takes into account the **inheritance** attributes of both the superclass and the subclass.

The automatic class assignment is done when a process calls the **exec** subroutine. The class assignment is reevaluated when a process uses a subroutine that can alter a process attribute used for classification purposes. Examples are the **setuid**, **setgid**, **setpri**, and **plock** subroutines.

In addition to this automatic class assignment, a user with the proper authority can manually assign processes or groups of processes to a specific superclass or subclass.

Resource Control

WLM manages three types of resources:

- CPU utilization of the threads in a class. This is the sum of all the CPU cycles consumed by every thread in the class.
- Physical memory utilization of the processes in a class. This is the sum of all the memory pages that belong to the processes in the class.
- Disk I/O bandwidth of the class. This is the bandwidth (in 512–byte blocks per second) of all the I/Os started by threads in the class on each disk device accessed by the class.

Resource Entitlements

WLM allows system administrators to specify per-class resource entitlements independently for each resource type. These entitlements can be specified by indicating the following:

- The target for usage of different types of resources. This target is specified with shares. The shares are specified as relative amounts of usage between different classes. For instance, if two classes have respectively 1 and 3 shares of CPU and are the only classes active at this time, their percentage goal used by WLM for its CPU regulation will be 25% and 75%, respectively.
- Minimum and maximum limits. These limits are specified as percentages of the total resource available. WLM supports two kinds of maximum limits:
  – A soft maximum limit indicates the maximum amount of the resource that can be made available when there is contention for the resource. This maximum can be exceeded if there is no contention; that is, if no one else requires the resource.
  – A hard maximum limit indicates the maximum amount of the resource that can be made available even if there is no contention on the resource.

  Generally, a hard maximum is better suited for a non-renewable resource such as memory, and a soft maximum is better suited for a renewable resource such as CPU, unless the system administrator wants to partition the resources of the system.

Setting Up WLM

Class definitions, class attributes, the shares and limits, and the automatic class assignment rules, can be entered using Web-based System Manager, SMIT, or the WLM command-line interface. These definitions and rules are kept in plain text files, that can also be created or modified using a text editor.

These files (called *WLM property files*) are kept in the subdirectories of **/etc/wlm**. A set of files describing superclasses and their associated subclasses define a WLM configuration. The files for the WLM **Config** configuration are in **/etc/wlm/Config**. This directory contains the definitions of the WLM parameters for the superclasses. The files are named **description**, **classes**, **shares**, **limits**, and **rules**. This directory might also contain subdirectories with the name of the superclasses where the subclass definitions are kept. For example, for the superclass *Super* of the WLM configuration **Config**, the directory **/etc/wlm/Config/Super** contains the property files for the subclasses of the superclass *Super*. The files are named **description**, **classes**, **shares**, **limits**, and **rules**.

After a WLM configuration has been defined by the system administrator, it can be made the active configuration using the **wlmcntrl** command.

Dynamic Control

When WLM is active, any parameter of the current configuration can be modified at any time, including the attributes of a class, its resource shares and limits, the assignment rules, and adding new classes or deleting existing classes. This can be done in several ways, such as:

- Modifying the property files for the currently active configuration (directory pointed to by the symbolic link **/etc/wlm/current**) and refreshing WLM by using the **wlmcntrl** command to use the new parameters.
- Creating another configuration with a different set of parameters and updating WLM to load the parameters of the new configuration, thus making it the current configuration.
- Modifying some of the parameters of the currently active configuration using the WLM command line interface (the **mkclass**, **chclass**, and **rmclass** commands).

Because switching configurations and updating parameters of the current configuration can be done using the WLM command-line interface, these commands can be used in scripts initiated by the **cron** facility to use a different set of classes and resource limits for different times of day (or days of the week).

Monitoring Tools

Use the following WLM commands to display WLM statistics and monitor the operation of WLM:

- The **wlmstat** command is text oriented and displays statistics as text (percentage of resource utilization per class for all the resource types managed by WLM).
- The **wlmmon** command gives a graphical view of per-class resource utilization and WLM regulation.
- The **wlmperf** command is an optional tool available with the Performance Toolbox and provides more capabilities, such as long-term record and replay.

WLM API

An Application Programming Interface (API) allows applications to perform any operation that can be done using the WLM command-line interface such as:

- Create, modify, or delete classes
- Change class attributes or resource shares and limits
- Manually assign processes to classes.

In addition, the API allows applications to set an application-defined classification attribute called *tag*. Setting this tag using a set of values provided by the system administrator (through the application user documentation) allows discrimination between several instances of the same application. The different classes can therefore be classified with different resource entitlements.

Per Class Accounting

The AIX accounting system utility lets you collect and report the use of various system resources by user, group, or WLM class. When process accounting is turned on, the operating system records statistics about the process resource usage in an accounting file when the process exits. Beginning with AIX 5.1, this accounting record includes a 64-bit numeric key representing the name of the WLM class that the process belonged to. (See "Accounting Overview" on page 157 for more information about the accounting system utility.)

The accounting subsystem uses a 64-bit key instead of the full 34-character class name to save space (otherwise the change would practically double the size of the accounting record). When the accounting command is run to extract the per-process data, the key is translated back into a class name using the above-mentioned routine. This translation uses the class names currently in the WLM configuration files. So, if a class has been deleted between the time the accounting record was written, when the process terminated, and the time the accounting report is run, the class name corresponding to the key cannot be found, and the class displays as `Unknown`.

To keep accurate records of the resource usage of classes deleted during an accounting period, do one of the following:

- Instead of deleting the class, keep the class name in the classes file and remove the class from the rules file so that no process can be assigned to it. Then you can delete the class after the accounting report has been generated at the end of the accounting period.
- Or, delete the class from the configuration it belongs to, and keep the class name in the classes file in a ″dummy″ configuration (one that is never activated) until after the accounting records for the period have been generated.

Overview of WLM Classes

The central concept of Workload Manager (WLM) is the concept of class. A *class* is a collection of processes (jobs) that has a single set of resource limits applied to it. WLM assigns processes to the various classes and controls the allocation of system resources among the different classes by using class-assignment rules. WLM controls the allocation of system resources among the different classes using per-class resource shares and limits set by the system administrator.

WLM allows system administrators to set up a hierarchy of classes by defining superclasses and then either defining subclasses for some or all of the superclasses themselves, or delegating the administration of the superclasses to superclass administrators who in turn will define the subclasses. The term class is used in reference to both superclasses and subclasses. Otherwise, the appropriate term is used.

The main difference between superclasses and subclasses is resource control (shares and limits). At the superclass level, the determination of resource entitlement based on the resource shares and limits is based on the total amount of each resource managed by WLM available on the machine. At the subclass level, the resource shares and limits are based on the amount of each resource allocated to the parent superclass.

A system administrator has the option to allocate a portion of the system resources to each superclass and then let superclass administrators distribute the allocated resources among the users and applications that they manage.

A process can be assigned to a class either automatically or manually. A process is assigned automatically when it starts using a set of class-assignment rules defined by a WLM administrator (the most common method). A process is assigned manually either by administrators or users with the appropriate level of privilege. Manual assignment overrides automatic assignment.

Class definitions, resource shares and limits, and class-assignment rules can be defined for both the superclass and subclass level by using SMIT and Web-based System Manager, the WLM command-line interface, or by applications using the WLM API. These definitions are kept in a set of text files called the WLM property files. The property files for a set of superclasses and subclasses constitute a WLM configuration.

The WLM administrator creates a name for each class. A class name is up to 16 characters in length and can contain only uppercase and lowercase letters, numbers and underscores ('_'). For a given WLM configuration, the names of all the superclasses must be different from one another, and the names of the subclasses of a given superclass must be different from one another. Subclasses of different superclasses can have the same name. To uniquely identify a subclass, the subclass full name composed of the superclass name and the subclass name separated by a dot must be used. The subclass *Sub* of superclass *Super* will thus be uniquely identified as *Super.Sub*.

Superclasses

The system administrator can define up to 27 superclasses. In addition, the following five superclasses are automatically created:

Default **superclass**
: Is the default superclass and is always defined. All non-root processes that are not automatically assigned to a specific superclass are assigned to the Default superclass. Other processes can also be assigned to the *Default* superclass by providing specific assignment rules.

System **superclass**
: Has all privileged (root) processes assigned to it if those processes are not assigned by rules to a specific class. This superclass also collects the memory pages belonging to kernel memory segments, kernel processes, and kernel threads. Other processes can also be assigned to the System superclass by providing specific assignment rules for this superclass. This superclass has a memory minimum limit of 1% as the default.

Shared **superclass**
: Receives the memory pages that are shared by processes in more than one superclass. This includes pages in shared memory regions and pages in files that are used by processes in more than one superclass (or in subclasses of different superclasses). Shared memory and files that are used by multiple processes that all belong to a single superclass (or subclasses of the same superclass) are associated with that superclass. Only when a process from a different superclass accesses the shared memory region or file are the pages placed in the Shared superclass. This

superclass can have only physical memory shares and limits applied to it. It cannot have shares or limits for the other resource types, subclasses, or assignment rules specified. Whether a memory segment shared by processes in different subclasses of the same superclass is classified into the *Shared* subclass or remains in its original subclass depends on the value of the **localshm** attribute of the original subclass.

Unclassified **superclass**
Is a memory allocation for unclassified processes. The processes in existence at the time that WLM is started are classified according to the assignment rules of the WLM configuration being loaded. During this initial classification, all the memory pages attached to each process are ″charged″ either to the superclass that the process belongs to (when not shared, or shared by processes in the same superclass), or to the *Shared* superclass when shared by processes in different superclasses.

However, a few pages cannot be directly tied to any processes (and thus to any class) at the time of this classification, and this memory is charged to the *Unclassified* superclass. Most of this memory is correctly reclassified over time, when it is either accessed by a process, or released and reallocated to a process after WLM is started. There are no processes in the *Unclassified* superclass. This superclass can have physical memory shares and limits applied to it. It cannot have shares or limits for the other resource types, subclasses, or assignment rules specified.

Unmanaged **superclass**
Is always defined. No processes will be assigned to this class. This class will be used to accumulate the CPU usage for all fixed priority processes and the memory usage for all pinned pages in the system. The CPU utilization for the wait processes is intentionally not accumulated in any class. Otherwise the system would always seem to be at 100% CPU utilization, which could be misleading for users when looking at the WLM statistics.

Subclasses

The system administrator or a superclass administrator can define up to 10 subclasses. In addition, two special subclasses, *Default* and *Shared*, are always defined.

Default **subclass**
Is the default subclass and is always defined. All processes that are not automatically assigned to a specific subclass of the superclass are assigned to the *Default* subclass. You can also assign other processes to the *Default* subclass by providing specific assignment rules.

Shared **subclass**
Receives all the memory pages that are used by processes in more than one subclass of the superclass. Included are pages in shared memory regions and pages in files that are used by processes in more than one subclass of the same superclass. Shared memory and files that are used by multiple processes that all belong to a single subclass are associated with that subclass. Only when a process from a different subclass of the same superclass accesses the shared memory region or file are the pages placed in the *Shared* subclass of the superclass. There are no processes in the *Shared* subclass. This subclass can have only physical memory shares and limits applied to it, and it cannot have shares or limits for the other resource types or assignment rules specified. Whether a memory segment shared by processes in different subclasses of the same superclass is classified into the *Shared* subclass or remains in its original subclass depends on the value of the **localshm** attribute of the original subclass.

Backward Compatibility

System administrators have the option of using either:

- Only superclasses, or
- Superclasses and subclasses.

The system administrator can also choose to create subclasses for some superclasses and not for others. In this case, the superclasses with no user-defined subclasses do not need to have a subdirectory **/etc/wlm/*config*/*superclass_name*** and property files associated to the superclass.

When starting WLM with configurations created with versions earlier to AIX 5.1, only superclasses are used. The default output of the **wlmstat** command lists only the superclasses and is similar to those of previous versions. For example:

```
# wlmstat
               CLASS  CPU  MEM  DKIO
        Unclassified    0    0   0
           Unmanaged    0    0   0
             Default    0    0   0
              Shared    0    2   0
              System    2   12   0
              class1    0    0   0
              class2    0    0   0
#
```

If some of the superclasses have subclasses defined by a WLM administrator, then the subclasses are listed. For example:

```
# wlmstat
               CLASS  CPU  MEM  DKIO
        Unclassified    0    0   0
           Unmanaged    0    0   0
             Default    0    0   0
              Shared    0    2   0
              System    3   11   7
              class1   46    0   0
    class1.Default     28    0   0
     class1.Shared      0    0   0
        class1.sub1    18    0   0
              class2   48    0   0
#
```

The output is the same when you use the **ps** command. For processes in a superclass without any subclasses, the **ps** command lists the superclass name as the process class name.

WLM Class Attributes

The attributes of a WLM class are:

Class Name
Can be up to 16 characters in length and can only contain uppercase and lowercase letters, numbers and underscores ('_').

Tier
A number between 0 and 9 used to prioritize resource allocation between classes.

Inheritance
Specifies whether a child process inherits the class assignment from its parent.

Adminuser, admingroup (superclass only)
Used to delegate the administration of a superclass.

Authuser, authgroup
Used to delegate the right to manually assign a process to a class.

Resource Set
Used to limit the set of resources a given class has access to in terms of CPUs (processor set).

Tier Attribute

Tiers are based on the importance of a class relative to other classes in WLM. There are 10 available tiers, from 0 through 9. Tier value 0 is the most important; the value 9 is the least important. As a result, classes belonging to tier 0 get resource allocation priority over classes in tier 1; classes in tier 1 have priority over classes in tier 2, and so on. The default tier number is 0.

The tier applies at both the superclass and the subclass level. Superclass tiers are used to specify resource allocation priority between superclasses. Subclass tiers are used to specify resource allocation priority between subclasses of the same superclass. There are no relationships between tier numbers of subclasses for different superclasses.

Inheritance Attribute

The inheritance attribute indicates whether a child process should inherit its parent class or be classified according to the automatic assignment rules upon the *exec* subroutine. The possible values are yes or no. The default is *no*.

This attribute can be specified at both superclass and subclass level. For a subclass of a given superclass:

- If the inheritance attribute is set to yes at both the superclass and the subclass level, then, upon exec(), a child of a process in the subclass will remain in the same subclass.
- If the inheritance attribute is yes for the superclass and no (or unspecified) for the subclass, then, upon exec(), a child of a process in the subclass will remain in the same superclass and will be classified in one of its subclasses according to the assignment rules for the superclass.
- If the inheritance attribute is no (or unspecified) for the superclass and yes for the subclass, then upon exec(), a child of a process in the subclass will be submitted to the automatic assignment rules for the superclasses.
  - If the process is classified by the rules in the same superclass, then it will remain in the subclass (it will not be submitted to the subclass's assignment rules).
  - If the process is classified by the superclass's rules in a different superclass, then the subclass assignment rules of the new superclass are applied to determine the subclass of the new superclass the process will be assigned to.
- If both superclass and subclass inheritance attribute are set to no (or unspecified), then, upon *exec*(), a child of a process in the subclass will be submitted to the standard automatic assignment.

localshm Attribute

The localshm attribute can be specified at the superclass and the subclass levels. It is used to prevent memory segments belonging to one class from migrating to the *Shared* superclass or subclass when accessed by processes in other classes. The possible values for the attribute are yes or no. A value of *yes* means that shared memory segments in this class must remain local to the class and not migrate to the appropriate *Shared* class. A value of *no* is the default when the attribute is not specified.

Memory segments are classified on page faults. When a segment is created, it is marked as belonging to the *Unclassified* superclass. On the first page fault on the segment, this segment is classified into the same class as the faulting process. If, later on, a process belonging to a different class than the segment page faults on this segment, WLM considers whether the segment needs to be reclassified into the appropriate *Shared* class (superclass or subclass). If the faulting process and the segment belong to different superclasses, one of the following occurs:

- If the segment's superclass has the localshm attribute set to *yes*, the segment remains in its current superclass. If the segment's subclass has the localshm attribute set to *yes*, the segment remains in its current subclass. If the superclass localshm attribute is set to *yes* but its subclass attribute is set to *no*, it goes into the *Shared* subclass of the current superclass.
- If the segment's superclass has the localshm attribute set to *no*, the segment goes to *Shared* superclass. This is the default action.

If the faulting process and the segment belong to different subclasses of the same superclass, and the segment's subclass has the localshm attribute set to *yes*, the segment remains in the current class (superclass and subclass). Otherwise, the segment goes to the *Shared* subclass of the superclass.

Of course, if the faulting process and the segment belong to the same class (same superclass and same subclass), the segment is not reclassified regardless of the values of the localshm attributes.

adminuser, admingroup Attributes

Note: These attributes are valid only for superclasses.

The adminuser and admingroup attributes are used to delegate the superclass administration to a user or group of users:

- adminuser specifies the name of the user (as listed in **/etc/passwd**) authorized to perform administration tasks on the superclass.
- admingroup specifies the name of the group of users (as listed in **/etc/group**) authorized to perform administration tasks on the superclass.

Only one value (user or group) is allowed for each attribute. Either of them, none, or both can be specified. The user or group will have authority to do the following:

- Create and delete subclasses
- Change the attributes and resource shares and limits for the subclasses
- Define, remove or modify subclass assignment rules
- Refresh (update) the active WLM configuration for the superclass.

Note: The root user always has authority on any superclass.

authuser, authgroup Attributes

The authuser and authgroup attributes are valid for all classes. They are used to specify the user or group authorized to manually assign processes to the class (superclass or subclass). When manually assigning a process (or a group of processes) to a superclass, the assignment rules for the superclass are used to determine to which subclass of the superclass each process will be assigned.

Only one value (user or group) is allowed for each attribute. Either of them, none, or both can be specified. In addition, the root user and the administrators of a superclass specified by adminuser or admingroup can always manually assign processes to a superclass or to a subclass of the superclass.

Resource Set Attribute

The resource set attribute (called *rset*) is valid for all the classes. Its value is the name of a resource set defined by the system administrator. The *rset* attribute represents a subset of the CPU resource available on the system (processor set). The default is ″system,″ which gives access to all the CPU resources available on the system.

Process to Class Assignment in WLM

In WLM, processes can be classified in either of two ways:

- Automatic assignment when a process calls the system call exec, using assignment rules specified by a WLM administrator. This automatic assignment is always in effect (cannot be turned off) when WLM is active. This is the most common method of assigning processes to the different classes.
- Manual assignment of a selected process or group of processes to a class by a user with the required authority on both the processes and the target class. This manual assignment can be done either by a

WLM command, which could be invoked directly or through SMIT or Web-based System Manager, or by an application using a function of the WLM Application Programming Interface. This manual assignment overrides the automatic assignment.

Automatic Assignment

The automatic assignment of processes to classes uses a set of class-assignment rules specified by a WLM administrator. There are two levels of assignment rules:

- A set of assignment rules at the WLM configuration level used to determine which superclass a given process is assigned to.
- Each superclass with subclasses defined, in turn has a set of assignment rules used to determine which subclass of the superclass the process is assigned to.

The assignment rules at both levels have exactly the same format.

Classification Attributes

The automatic assignment is based on the values of a set of process attributes. These attributes are as follows:

- Process user ID
- Process group ID
- Path name of the application (program) executed
- Type of the process (32/64 bit, for instance)
- Process tag.

The tag is a process attribute, defined as a character string, that an application can set by program, using the WLM Application Programming Interface (API).

The classification is done by comparing the value of these process attributes at exec time against lists of possible values given in the class assignment rules file (called **rules**), to determine which rule is a match for the current value of the process attributes.

A class assignment is a text string with different fields separated by one or more blank spaces or tabs. The fields are as follows:

Name	Must contain the name of a class which is defined in the class file corresponding to the level of the **rules** file (superclass or subclass). Class names can contain only uppercase and lowercase letters, numbers, and underscores and can be up to 16 characters in length. No assignment rule can be specified for the system defined classes Unclassified, Unmanaged and Shared.
Reserved	Reserved for future extension. Its value must be a hyphen (-), and it must be present.
User	May contain either a hyphen (-) or a list of valid user names (as defined in the **/etc/passwd** file). The list is composed of one or more names, separated by a comma (,). An exclamation mark (!) can be used before a name to exclude a given user from the class. Patterns can be specified to match a set of user names, using full Korn shell pattern-matching syntax.
Group	May contain either a hyphen (-) or a list of valid group names (as defined in the **/etc/group** file). The list is composed of one or more names, separated by a comma (,). An exclamation mark (!) can be used before a name to exclude a given group from the class. Patterns can be specified to match a set of user names using full Korn shell pattern matching syntax.
Application	May contain either a hyphen (-) or a list of application path names. This is the path name of the applications (programs) executed by processes included in the class. The application names will be either full path names or Korn shell patterns that match path names. The list is composed of one or more path names, separated by a comma (,). An exclamation mark (!) can be used before a name to exclude a given application.

Type	May contain either a hyphen (-) or a list of process attributes. The possible values for these attributes are as follows: • **32bit**: the process is a 32–bit process • **64bit**: the process is a 64–bit process • **plock**: the process has called the **plock** subroutine to pin memory • **fixed**: the process is a fixed priority process (SCHED_FIFO or SCHED_RR) The value of the type field can be a combination of one or more of the above attributes separated by a plus (+). 32bit and 64bit are mutually exclusive.
Tag	May contain either a hyphen (-) or a list of application tags. An application tag is a string of up to 30 alphanumeric characters. The list is composed of one or more application tag values separated by commas.

Example of Assignment Rules

The following is an example of a top-level **rules** file for the configuration *Config* (**/etc/wlm/Config/rules** file):

```
* This file contains the rules used by WLM to
* assign a process to a superclass
*
* class resvd user    group   application     type  tag
db1      -      -       -    /usr/bin/oracle*      _DB1
db2      -      -       -    /usr/bin/oracle*      _DB2
devlt    -      -      dev        -            -    -
VPs      -   bob,ted    -         -            -    -
acctg    -      -     acct*       -            -    -
System   -    root      -         -            -    -
Default  -      -       -         -            -    -
```

The following is an example of the **rules** file for the **devlt** superclass in the **/etc/wlm/Config/devlt/rules** file:

```
* This file contains the rules used by WLM to
* assign a process to a subclass of the
* superclass devlt
*
* class  resvd   user    group  application         type       tag
hackers   -    jim,liz     -       -                 -          -
hogs      -       -        -       -            64bit+plock    -
editors   -     !sue       -   /bin/vi,/bin/emacs    -          -
build     -       -        -   /bin/make,/bin/cc     -          -
Default   -       -        -       -                 -          -
```

Note: The asterisk (*) is the comment character used in the **rules** file.

Automatic Classification

When a process is created (fork), it remains in the same class as its parent. Reclassification happens when the new process issues a system call which can modify one of the attributes of the process used for classification; for example, *exec*, *setuid* (and related calls), *setgid* (and related calls), *setpri* and *plock*.

Note: Special cases in which reclassification does not occur are cited when applicable.

To classify the process, WLM examines the top-level **rules** file for the active configuration to determine which superclass the process belongs. For each rule in the file, WLM checks the current values for the process attributes against the values and lists of values specified in the rule. Rules are checked in the order that they appear in the file. When a match is found, the process is assigned to the superclass named in the first field of the rule. Then the rules file for the superclass is examined in the same way to determine to which subclass the process should be assigned.

For a process to match one of the rules, each of its attributes must match the corresponding field in the rule. The following is a list of the criteria used to determine whether the value of an attribute matches the values in the field of the **rules** file:

- If the field in the rules file has a value of hyphen (-), then any value of the corresponding process attribute is a match.
- For all the attributes except *type*, if the value of the process attribute matches one of the values in the list in the rules file that is not excluded (prefaced by a ″!″), then a match has occurred.
- For the *type* attribute, if one of the values in the rule is comprised of two or more values separated by a plus (+), then a process is a match only if its characteristics match all the values.

At both superclass and subclass levels, WLM goes through the rules in the order in which they appear in the **rules** file, and classifies the process in the class corresponding to the first rule for which the process is a match. The order of the rules in the rules file is therefore extremely important. Use caution when you create or modify the rules file.

The following are examples using the rules files:

- If user joe from group acct3 executes **/bin/vi**, then the process will be put in superclass acctg.
- If user sue from group dev executes **/bin/emacs**, then the process will be in the superclass devlt (group ID match), but will not be classified into the editors subclass, because user sue is excluded from that class. The process will go to devlt. Default.
- When a DB administrator starts **/usr/sbin/oracle** with a user ID of oracle and a group ID of dbm, to serve the data base DB1, the process will be classified in the Default superclass. Only when the process sets its tag to _DB1 will it be assigned to superclass db1.

All of the preceding examples assume that the superclasses and subclasses described do not have the inheritance attribute set to yes. Otherwise, the new processes would simply inherit the superclass or subclass from their father.

Manual Assignment in WLM

A process or a group of processes can be manually assigned to a superclass and/or subclass by using Web-based System Manager, SMIT, or the **wlmassign** command. An application can assign processes through the **wlm_assign** API function.

To manually assign processes to a class or to cancel an existing manual assignment, a user must have the appropriate level of privilege. (See "Security Considerations" on page 144 for further details.) A manual assignment can be made or canceled separately at the superclass level, the subclass level, or both. This assignment is specified by flags for the programming interface and a set of options for the command line interface used by the WLM administration tools. So, a process can be manually assigned to a superclass only, a subclass only, or to a superclass and a subclass of that superclass. In the latter case, the dual assignment can be done simultaneously (with a single command or API call) or at different times, by different users.

Assignment is very flexible, but can be confusing. Following are two examples of the possible cases.

Example 1: First Assignment of Processes

A system administrator manually assigns *Process1* from *superclassA* to *superclassB* (superclass-level-only assignment). The automatic assignment rules for the subclasses of *superclassB* are used by WLM to determine to which subclass the process is ultimately assigned. *Process1* is assigned to *superclassB.subclassA* and is flagged as having a ″superclass only″ assignment.

A user with the appropriate privileges assigns *Process2* from its current class *superclassA.subclassA* to a new subclass of the same superclass, *superclassA.subclassB*. *Process2* is assigned to its new subclass and flagged as having a ″subclass only″ assignment.

A WLM administrator of the subclasses of *superclassB* manually reassigns *Process1* to *subclassC*, which is another subclass of *superclassB*. *Process1* is reclassified into *superclassB.subclassC* and is now flagged as having both superclass and subclass level assignment.

Example 2: Reassignment or Cancellation of Processes

The reassignment and cancellation of a manual assignment at the subclass level is less complex and affects only the subclass level assignment.

Suppose that the system administrator wants *Process2* to be in a superclass with more resources and decides to manually assign *Process2* to *superclassC*. In Example 1, *Process2* was manually assigned to *subclassB* of *superclassA*, with a ″subclass only″ assignment. Because *Process2* is assigned to a different superclass, the previous manual assignment becomes meaningless and is canceled. *Process2* now has a ″superclass only″ manual assignment to *superclassC* and is assigned to a subclass of *superclassC* using the automatic assignment rules.

Now, the system administrator decides to terminate the manual assignment from *Process1* to *superclassB*. The ″superclass level″ manual assignment of *Process1* is canceled, and *Process1* is assigned a superclass using the top level automatic assignment rules.

If the rules have not changed, *Process1* is assigned to *superclassA*, and its subclass level manual assignment to *superclassB.subclassC* becomes meaningless and is canceled.

If for some reason the top level rules assign *Process1* to *superclassB*, then the subclass level assignment to *superclassB.subclassC* is still valid and remains in effect. *Process1* now has a ″subclass only″ manual assignment.

Interactions with Class Inheritance

When a process is manually assigned to a superclass or subclass with the inheritance attribute set to yes, and the process is a process group leader, WLM attempts to reclassify all the processes in the process group.

In some cases, some of the processes in the process group are not reclassified to the new class of the group leader. For instance, when processes have been manually assigned to their current class, they remain in their assigned class. It is also possible that one or more of the processes in a group require a higher level of privileges than the user performing the manual assignment.

Security Considerations

To assign a process to a class or to cancel a prior manual assignment, the user must have authority both on the process and on the target class. These constraints translate into the following rules:

- The root user can assign any process to any class.
- A user with administration privileges on the subclasses of a given superclass (that is, the user or group name matches the user or group names specified in the attributes **adminuser** and **admingroup** of the superclass) can manually reassign any process from one of the subclasses of this superclass to another subclass of the superclass.
- Users can manually assign their own processes (ones associated with the same real or effective user ID) to a subclass for which they have manual assignment privileges (that is, the user or group name matches the user or group names specified in the attributes **authuser** and **authgroup** of the superclass or subclass).

To modify or terminate a manual assignment, users must have at least the same level of privilege as the person who issued the last manual assignment.

Managing Resources with WLM

WLM monitors and regulates the resource utilization, on a per-class basis, of the threads and processes active on the system. You can set minimum or maximum limits per class for each resource type managed by WLM, as well as a target value per class for each resource . This target is representative of the amount of the resource that is optimal for the jobs in the class.

The shares and limits at the superclass level refer to the total amount of each resource available on the system. At the subclass level, shares and limits refer to the amount of each resource made available to the superclass that the subclass is in (superclass target). The hierarchy of classes is a way to divide the system resources between groups of users (superclasses) and delegate the administration of this share of the resources to superclass administrators. Each superclass administrator can then redistribute this amount of resources between the users in the group by creating subclasses and defining resource entitlements for these subclasses.

Resource Types

WLM manages three types of resources:

CPU utilization of the threads in a class	This is the sum of all the CPU cycles consumed by every thread in the class.
Physical memory utilization for the processes in a class	This is the sum of all the memory pages which belong to the processes in the class.
Disk I/O bandwidth for the class	This is the bandwidth (in 512-byte blocks per second) of all the I/Os started by threads in the class on each disk device accessed by the class.

Every second, WLM calculates the per-class utilization for each resource during the last second, as a percentage of the total resource available, as follows:

- For the CPU, the total amount of CPU time available every second is equal to 1 second times the number of CPUs on the system. For instance, on an eight-way SMP, if all the threads of a class combined consumed 2 seconds of CPU time during the last second, this represents a percentage of 2/8 = 25%. The percentage used by WLM for regulation is a decayed average over a few seconds of this ″instantaneous″ per-second resource utilization.
- For physical memory, the total amount of physical memory available for processes at any given time is equal to the total number of memory pages physically present on the system minus the number of pinned pages. Pinned pages are not managed by WLM because these pages cannot be stolen from a class and given to another class to regulate memory utilization. The memory utilization of a class is the ratio of the number of non-pinned memory pages owned by all the processes in the class to the number of pages available on the system, expressed as a percentage.
- For disk I/O, the main problem is determining a meaningful available bandwidth for a device. When a disk is 100% busy, its throughput, in blocks per second, is very different if one application is doing sequential I/Os than if several applications are generating random I/Os. If you only used the maximum throughput measured for the sequential I/O case (as a value of the I/O bandwidth available for the device) to compute the percentage of utilization of the device under random I/Os, you might be misled to think that the device is at 20% utilization, when it is in fact at 100% utilization.

To get more accurate and reliable percentages of per-class disk utilization, WLM uses the statistics provided by the disk drivers (displayed using the **iostat** command) giving for each disk device the percentage of time that the device has been busy during the last second. WLM counts the number of total blocks which have been read or written on a device during the last second by all the classes accessing the device, and how many blocks have been read or written by each class and what was the percentage of utilization of the device. WLM then calculates the percentage of the disk throughput consumed by each class

For instance, if the total number of blocks read or written during the last second was 1000 and the device had been 70% busy, this means that a class reading or writing 100 blocks used 7% of the disk bandwidth. Similarly to the CPU time (another renewable resource), the values used by WLM for its disk I/O regulation are also a decayed average over a few seconds of these per second percentages.

For the disk I/O resource, the shares and limits apply to each disk device individually accessed by the class. The regulation is done independently for each device. This means that a class could be over its entitlement on one device, and the I/Os to this device will be regulated while it is under its entitlement on another disk and the I/Os to this other device will not be constrained.

Target Shares

The target for use of different resource types is specified by shares. The shares are specified as relative amounts of usage between different classes. If unspecified, the default share is 1. One way of thinking about shares is as a self-adapting percentage.

For example, a system has three defined classes, A, B, and C, whose targets are 50, 30 and 20, respectively.

- If all three classes are active, the total number of shares for the active classes is 100. Their targets, expressed as percentages are 50%, 30% and 20%, respectively.
- If A is not active, the total number of shares is 50 (so each share represents 2%). The target percentages for B and C are 60% and 40%.
- If only one class is active, its target is 100%.
- A class is considered active (regardless of its resource consumption) when it has at least one process assigned to it.

In this example, the sum of the shares for the three classes was 100 simply to make the sample calculation easier. Target shares can be any number between 1 and 65535.

The preceding example supposes that:

- A, B, and C are either all superclasses, or all subclasses of the same superclass, and
- A, B, and C are in the same tier.

The relative share numbers of a subclass and a superclass, of two subclasses of different superclasses, or of classes in different tiers do not give any indication of their relative resource entitlements.

The shares are used by WLM to calculate a percentage goal of resource utilization of each resource type in each class. This goal represents a percentage of resources that can vary widely depending on how many classes are active at any given time. However, WLM makes sure that the dynamic value of this percentage goal remains compatible with the minimum and maximum values for the class. If the calculated percentage is below the minimum, WLM uses the minimum as the target. If the percentage is above the maximum range, WLM uses the maximum as the target. If the percentage is between the minimum and maximum, WLM uses the calculated value.

The shares number can be specified as a hyphen ('-') for any resource type to indicate that the class's resource utilization for this resource type is not regulated by WLM. This is the default when no shares value has been specified for a resource type. This default is different from the default value of 1 share in the first version of WLM. Shares numbers can remain at their default value for the System class, but it is safer to set up specific shares values for the Default class.

The shares are specified, per class, for all the resource types. Shares are specified in stanzas of the **shares** file. For example:

```
shares
classname:
   CPU      =   2
   memory   =   4
   diskIO   =   3
```

Minimum and Maximum Resource Limits

The different resources can be limited by the following values:

- The minimum percentage of the resource that must be made available when requested. The possible values are integers from 0 to 100. If unspecified, the default value is 0.
- The maximum percentage of a resource that can be made available, when there is contention for the resource. This is called a ″soft″ maximum, because it is possible for a class to get more resource than this soft maximum value if there is no contention. The possible values are integers from 1 to 100. If unspecified, the default value is 100.
- The maximum percentage of a resource that can be made available, even if there is no contention for the resource. This is called a ″hard″ maximum. A class will never get more resource than its hard maximum limit, even if it is the only class active on the system. The possible values are integers from 1 to 100. If unspecified, the default value is 100.

Resource limit values are specified in the resource limit file by resource type within stanzas for each class. The limits are specified as a minimum to soft maximum range, separated by a hyphen (-) with white space ignored. The hard maximum when specified follows the soft maximum, and is separated by a semi-colon (;). Each limit value is followed by a percent sign (%).

The following are the only constraints that WLM places on resource limit values:

- The minimum limit must be less than or equal to the soft maximum limit.
- The soft maximum limit must be less than or equal to the hard maximum limit.
- The sum of the minimum of all the superclasses within a tier cannot exceed 100.
- The sum of the minimum of all the subclasses of a given superclass within a tier cannot exceed 100.

When a class with a hard memory limit has reached this limit and requests more pages, the VMM page replacement algorithm (LRU) is initiated and ″steals″ pages from the limited class, thereby lowering its number of pages below the hard maximum, before handing out new pages. This behavior is correct, but extra paging activity, which can take place even where there are plenty of free pages available, impacts the general performance of the system. Minimum memory limits for other classes are recommended before imposing a hard memory maximum for any class.

The constraint of the sum of the minimum limits within a tier being less than or equal to 100 means that a class in the highest priority tier is always allowed to get resources up to its minimum limit. WLM does not guarantee that the class will actually reach its minimum limit. This depends on how the processes in the class use their resources and on other limits that are in effect. For example, a class might not reach its minimum CPU entitlement because it cannot get enough memory.

For physical memory, setting a minimum memory limit provides some protection for the memory pages of the class's processes (at least for those in the highest priority tier). A class should not have pages stolen when it is below its minimum limit unless all the active classes are below their minimum limit and one of them requests more pages. A class in the highest tier should never have pages stolen when it is below its minimum. Setting a memory minimum limits for a class of interactive jobs helps make sure that their pages will not all have been stolen between consecutive activations (even when memory is tight) and improves response time.

Limits are specified for all resource types, per class, in stanzas of the **limits** file. For example:

```
shares
classname:
   CPU     =   0%-50%;80%
   memory  =   10%-30%;50%
```

In this example, no limits are set for disk I/O. Using the system defaults, this translates to the following:

```
diskIO   =   0%-100%;100%
```

Backward Compatibility Considerations

Before AIX 5.1, the default value for unspecified WLM resource shares was one. Beginning with AIX 5.1, the default value is hyphen (-), which means that the resource consumption of the class for this resource is not regulated by WLM. System administrators should review their existing configurations to ensure that new default is appropriate for their classes, or if the default should be reset to 1 or to a different explicit value for some of the classes. For more information on setting user-defined defaults, see shares in the in *AIX 5L Version 5.1 Files Reference*.

Earlier versions of WLM only had a maximum limit, which was a ″soft″ limit for CPU and a ″hard″ limit for memory. Limits specified with the `min%-max%` format have the max interpreted as a softmax for CPU and a hardmax for memory. For limits that are set with AIX 5.1 or later, all interfaces except direct file editing either convert all existing data to the new format or create new limits with the new format. Administrators using text editing must manually convert the **limits** file.

The disk I/O resource is available in AIX 5.1 and later, so when activating WLM with configuration files from an earlier version, the values for the shares and the limits are set to default values. The system defaults are:

- `shares = -`
- `min = 0%, softmax = 100%, hardmax = 100%`

For WLM configurations previous to AIX 5.1, the disk I/O resource is not regulated by WLM and the class behaves as it normally does for your earlier version.

Examples of WLM Classification and Limits

Several methods exist for classifying a process, and all operate concurrently. A top-down strictest first matching algorithm is used to provide for maximum configuration flexibility. You can organize process groupings by user with special cases for programs with certain names, by pathname with special cases for certain users, or any other arrangement.

Example with CPU Limits

The following example examines CPU allocation, assuming that each class can consume all the CPU that it is given.

Two classes, A and B, are in the same tier. CPU limits for A are [30% - 100%]. CPU limits for B are [20% - 100%]. When both classes are running and are using sufficient CPU, WLM first makes sure that they both get their minimum percentages of each second (averaged over several seconds). Then WLM distributes the remaining CPU cycles according to any CPU target share values.

If the CPU target shares for A and B are 60% and 40% respectively, then the CPU utilization for A and B stabilize at 60% and 40% respectively.

A third class, C, is added. This class is a group of CPU-bound jobs and should run with about half (or more) of the available CPU. Class C has limits of [20% - 100%] and CPU target shares of 100%. If C is in the same tier as A and B, then when C is starting, A and B see their CPU allocation decrease steeply and the three classes stabilize at 30%, 20% and 50%, respectively. Their targets in this case are also the minimum for A and B.

A system administrator might not want batch jobs to consume 50% of the CPU when other jobs, possibly of higher priority, are also running. In a situation like the previous example, C is placed in a lower priority tier. C then receives whatever CPU remains after A and B receive their needs. In the above example, C receives no CPU time, because A and B are each capable of absorbing 100% of the CPU. In most situations, however, A and B, in a high-priority tier, are composed of interactive or transaction-oriented jobs, which do not use all of the CPU all of the time. C then receives some share of the CPU, for which it competes with other classes in the same or lower tiers.

Example with Memory Limits

The following example examines memory allocation to groups of processes with varying memory targets. Three groups of processes must run: a group of interactive processes that need to run whenever they are used (PEOPLE), a batch job that always runs in the background (BATCH1), and a second, more important batch job, that runs every night (BATCH0).

PEOPLE has a specified memory minimum of 20%, a memory target of 50 shares, and a class tier value of 1. A 20% minimum limit ensures that the desktop applications in this class resume fairly quickly when users touch their keyboards.

BATCH1 has a memory minimum of 50%, a memory target of 50 shares, and a tier value of 3.

BATCH0 has a memory minimum of 80%, a memory target of 50 shares, and a tier value of 2.

Classes PEOPLE and BATCH1 have a total memory minimum limit of 70. Under normal operation (when BATCH0 is not running), both of these classes are allowed to get all of their reserved memory. They share the rest of the memory in the machine about half and half, even though they are in different tiers. At midnight when BATCH0 is started, the memory minimum total reaches 150. WLM ignores the minimum requirements for the lowest tiers until processes in the upper tiers exit. BATCH0 takes memory from the BATCH1 50% reserve, but not from the PEOPLE 20% reserve. After BATCH0 is done, memory reserves for tier 3 processes are honored again and the system returns to its normal memory balance.

Setting Up WLM

This section discusses some of the steps that a system administrator takes to configure WLM.

Step 1: Design Your Classification

Define your superclasses first. To define which classes you need, you must know your users and their computing needs, as well as the applications on your system, their resource needs and the requirements of your business (for example, which tasks are critical and which can be given a lower priority).

Setting priorities is dependent on what function WLM serves in your organization. In the case of server consolidation, you might already know the applications, the users and their resource requirements, and you might be able to skip or shorten some of the steps.

WLM allows you to classify processes by user or group, application, type, tag, or a combination of these attributes. Because WLM regulates the resource utilization among classes, system administrators should group applications and users with the same resource utilization patterns into the same classes. For instance, you might want to separate the interactive jobs that typically consume very little CPU time but require quick response time from batch-type jobs that typically are very CPU- and memory-intensive. This is the same in a database environment in which you need to separate the OLTP-type traffic from the heavy-duty queries of data mining.

Step 2: Create the Superclasses and Assignment Rules

This step is done using Web-based System Manager, SMIT, or command-line interface. For the first few times, it is probably a good idea to use Web-based System Manager or SMIT to take you through the steps of creating your first WLM configuration, including defining the superclasses and setting their attributes. For the first pass, you can set up some of the attributes and leave the others at their default value. This is the same for the resource shares and limits. All these class characteristics can be dynamically modified at a later time.

You can then start WLM in passive mode, check your classification, and start reviewing the resource utilization patterns of your applications.

Step 3: Use WLM to Refine Your Class Definitions

Verify your configuration, using the **wlmcheck** command or the corresponding SMIT or Web-based System Manager menus. Then start WLM in passive mode on the newly defined configuration. WLM will classify all the existing processes (and all processes created from that point on) and start compiling statistics on the CPU, memory, and disk I/O utilization of the various classes. WLM will not try to regulate this resource usage.

Verify that the various processes are classified in the appropriate class as expected by the system administrator (using the **-o** flag of the **ps** command). If some of the processes are not classified as you expect, you can modify your assignment rules or set the inheritance bit for some of the classes (if you want the new processes to remain in the same class as their parent) and update WLM. You can repeat the process until you are satisfied with this first level of classification (superclasses).

Running WLM in passive mode and refreshing WLM (always in passive mode) involves low risk, has low overhead operation, and can be done safely on a production system without disturbing normal system operation. To activate and refresh WLM, use the **wlmcntrl** command, invoked either from the command line or from SMIT or Web-based System Manager.

Step 4: Gather Resource Utilization Data

Run WLM in passive mode to gather statistics by using the **wlmstat** command. The **wlmstat** command can be used at regular time intervals to display the per-class resource utilization as a percentage of the total resource available, for superclasses). This allows you to monitor your system for extended periods of time to review the resource utilization of your main applications.

With this data, and your business goals defined in step 1, decide which tier number will be given to every superclass and what share of each resource should be given to the various classes.

Step 5: Start WLM

You are now ready to start WLM in active mode. Monitor the system using the **wlmstat** command and verify that the regulation done by WLM is in line with your goals and if applications are not unduly deprived of resources while others get more than they should. If this is the case, adjust the shares and refresh WLM.

For specific cases, you might have to use minimum or maximum limits. Adjust the shares and tier numbers to reach your resource-allocation goals. Reserve limits for cases that cannot be solved only with shares.

- Use minimum limits for applications that typically have low resource usage but need a quick response time when activated by an external event. One of the problems faced by interactive jobs in situations where memory becomes tight is that their pages get stolen during the periods of inactivity. A memory minimum limit can be used to protect some of the pages of interactive jobs if the class is in tier 0.
- Use maximum limits to contain some resource-intensive, low-priority jobs. Unless you partition your system resources for other reasons, a hard maximum will make sense mostly for a nonrenewable resource such as memory. This is because of the time it takes to write data out to the paging space if a

higher priority class needs pages that the first class would have used. For CPU usage, you can use tiers or soft maximum to make sure that if a higher priority class is immediately assigned CPU time.

Monitor and adjust the shares, limits, and tier numbers until you are satisfied with the system behavior.

Step 6: Fine-Tune Your Configurations

Decide whether you need to use subclasses and if you want to delegate the administration for the subclasses for some or all of the superclasses. When creating and adjusting the parameters of subclasses, you can refresh WLM only for the subclasses of a given superclass that do not affect users and applications in the other superclasses.

The administrator of each superclass can repeat this process for the subclasses of each superclass. The only difference is that WLM cannot run in passive mode at the subclass level only. The subclass configuration and tuning has to be done with WLM in active mode. One way not to impact users and applications in the superclass is to start the tier number, and the shares and limits for the subclasses at their default value ('-' (hyphen) for shares, 0% for minimum, and 100% for soft and hard maximum). With these settings, WLM will not regulate the resource allocation between the subclasses.

The administrator can then monitor and set up the subclass shares, limits, and tier number.

Step 7: Create Other Configurations as Needed

Repeat steps 1 through 6 to define other configurations with different parameters, according to the needs of the business. When doing so, you can save time by copying and modifying existing configurations.

Step 8: Specifying WLM Properties

You can specify the properties for the WLM subsystem by using Web-based System Manager, SMIT, an ASCII-oriented user interface, or by creating flat ASCII files. The Web-based System Manager and SMIT interfaces record the information in the same flat ASCII files. These files are called the WLM property files and are named **classes**, **description**, **rules**, **limits**, and **shares**. The WLM property files can only be loaded by the root user.

You can define multiple sets of property files, defining different configurations of workload management. These configurations are usually located in subdirectories of **/etc/wlm**. A symbolic link **/etc/wlm/current** points to the directory containing the current configuration files. This link is updated by the **wlmcntrl** command when WLM starts with a specified set of configuration files.

WLM Application Programming Interface

This API is comprised of a set of routines in the **/usr/lib/libwlm.a** library, which enable applications to perform all the tasks that a WLM administrator can perform using the WLM commands, including:

- Creating classes
- Changing classes
- Removing classes
- Manually assigning processes to specific classes
- Retrieving WLM statistics

In addition, the **wlm_set_tag** routine allows an application to set up an application tag and specify whether this tag should be inherited by child processes at **fork** or **exec**. The library provides support for multi-threaded 32-bit or 64-bit applications.

Application Tag

The application tag is a string of characters and is used as one of the classification criteria for the automatic classification of processes (using the **rules** file). This tag basically provides an application-defined classification criteria in addition to the system-defined criteria such as *user*, *group*, *application* and *type*.

When an application process sets its tag, it is immediately reclassified using the superclass and subclass rules in effect for the currently active WLM configuration. WLM then reviews the assignment rules looking for a match, using all the process attributes, including the new tag.

To be effective, this tag must appear in one or more of the assignment rules. The format and the use of the various tags by each application must be clearly specified in the application's administration documentation and well-known to the WLM administrators so that they can use the various values of the tags in their assignment rules to distinguish between different instances of the same application.

Because different users might have different requirements regarding what set of characteristics of the application processes they want to use to classify those processes, it is recommended that the application provide a set of configuration or run time attributes that can be used to build the tag. The application administrator can specify the format of this tag to the application. The attributes that can be used for the tag and the syntax to specify the format of the WLM tag are application-dependent and are the responsibility of the application provider.

For example, an instance of a database server is able to determine which database it is working on (*db_name*) and through which TCP port a given user is connected (*port_num*). Administrators might have any of the following priorities:

- To create different classes for processes accessing different databases to give each class different resource entitlement
- To separate the processes serving remote requests from different origins and to use the port number as a classification attribute
- To create one superclass for each database and subclass per port number in each superclass.

One way to accommodate these different needs is to specify the content and format of the tag. In this example, assume that the tag can be passed to the application in a configuration file or run time parameter such as `WLM_TAG=`*`$db_name`* or `WLM_TAG=`*`$db_name_$port_num`*.

When setting its tag, an application can specify whether this tag will be inherited by its children so that all the processes spawned by a specific instance of an application can be classified in the same class. Setting the tag inheritance is how the application tag is most commonly used.

The following is an example of how application tags could be used. In this example, the tag of the database is the same as the database name. Then two instances of the server working on two different databases would set up two different tags, for instance *db1* and *db2*.

A system administrator could create two different classes *dbserv1* and *dbserv2* and classify the two database servers (and all their children if tag inheritance is used) in these classes using the tags. It would then be possible to give each class different resource entitlement according to specific business goals.

The corresponding assignment rules could look similar to the following:

```
* class   resvd  user  group     application     type  tag
*
dbserv1     -     -    dbadm     /usr/sbin/dbserv  -   db1
dbserv2     -     -    dbadm     /usr/sbin/dbserv  -   db2
```

Class Management APIs

The WLM API provides applications with the ability to:

- Query the names and characteristics of the existing classes of a given WLM configuration (**wlm_read_classes**).
- Create a new class for a given WLM configuration, define the values of the various attributes of the class (such as tier and inheritance) and the shares and limits for the resources managed by WLM, such as CPU, physical memory, and block I/O (**wlm_create_class**).
- Change the characteristics of an existing class of a given WLM configuration, including the class attributes and resource shares and limits (**wlm_change_class**).
- Delete an existing class of a given configuration (**wlm_delete_class**).

The changes will be applied only to the property files of the specified WLM configuration. Optionally, by specifying an empty string as the configuration name, it is possible to apply the change only to the in-core classes, resulting in an immediate update of the active configuration state.

The API calls require from the caller the same level of privilege that would be required for the command line or for the SMIT or Web-based System Manager interfaces, as follows:
- Any user can read the class names and characteristics
- Only the root user can create, modify, or delete superclasses
- Only the root user or designated superclass administrators (superclass attributes *adminuser* or *admingroup*) can create, modify, or delete subclasses of a given superclass.

In cases where WLM administration is accomplished, both through the command line and administration tools by WLM administrators, and by application(s) through the API, some caution must be applied. Both interfaces share the same name space for the superclass and subclass names, and the total number of superclasses and subclasses.

In addition, when the API directly modifies the in-core WLM data (create new classes, for example), WLM administrators are not aware of this until classes that they did not create appear on the output of commands such as the **wlmstat** command. To avoid conflicts that would confuse the applications using this API when the system administrator updates WLM, the classes created through the API that are not defined in the WLM property files are not automatically removed from the in-core data. They remain in effect until explicitly removed through the **wlm_delete_class** routine or through an invocation of the **rmclass** command (invoked directly or through SMIT or Web-based System Manager by the system administrator).

The WLM API also provides applications with the ability to:
- Query or change the mode of operation of WLM using the **wlm_set** function
- Query the current status of WLM
- Stop WLM
- Toggle between active to passive mode
- Turn the **rset** binding on and off
- Start or update WLM with the current or an alternate configuration by using the **wlm_load** routine
- Assign a process or a group of processes to a class by using the **wlm_assign** routine.

The API requires the same levels of privilege as the corresponding **wlmcntrl** and **wlmassign** commands:
- Any user can query the state of WLM
- Only the root user can change the mode of operation of WLM
- Only the root user can update or refresh a whole configuration
- root or an authorized superclass administrator (*adminuser* or *admingroup*) can update WLM for the subclasses of a given superclass

- root, an authorized user (specified by *authuser* or **authgroup**), or an authorized superclass administrator (*adminuser* or *admingroup*) can assign processes to a superclass or subclass. See the **wlmassign** command for details.

WLM Statistics API

The WLM API routines and **wlm_get_bio_stats** provide application access to the WLM statistics displayed by the **wlmstat** commands.

WLM classification API

The **wlm_check** routine allows the user to verify the class definitions and the assignment rules for a given WLM configuration. The API routine **wlm_classify** (see “Process to Class Assignment in WLM” on page 140) allows an application to determine to which class a process with a specified set of attributes would be classified.

Binary Compatibility

To provide binary compatibility if there are future changes to the data structures, each API call is passed a version number as a parameter, which will allow the library to determine with which version of the data structures the application has been built.

Chapter 14. System Resource Controller and Subsystems

This chapter provides an overview of the System Resource Controller (SRC) and the various subsystems it controls. For task procedures, see System Resource Controller and Subsystemsin *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

System Resource Controller Overview

The System Resource Controller (SRC) provides a set of commands and subroutines to make it easier for the system manager and programmer to create and control subsystems. A *subsystem* is any program or process or set of programs or processes that is usually capable of operating independently or with a controlling system. A subsystem is designed as a unit to provide a designated function.

The SRC was designed to minimize the need for operator intervention. It provides a mechanism to control subsystem processes using a common command line and the C interface. This mechanism includes the following:

- Consistent user interface for start, stop, and status inquiries
- Logging of the abnormal termination of subsystems
- Notification program called at the abnormal system termination of related processes
- Tracing of a subsystem, a group of subsystems, or a subserver
- Support for control of operations on a remote system
- Refreshing of a subsystem (such as after a configuration data change).

The SRC is useful if you want a common way to start, stop, and collect status information on processes.

Subsystem Components

A subsystem can have one or more of the following properties:

- Is known to the system by name
- Requires a more complex execution environment than a subroutine or nonprivileged program
- Includes application programs and libraries as well as subsystem code
- Controls resources that can be started and stopped by name
- Requires notification if a related process is unsuccessful to perform cleanup or to recover resources
- Requires more operational control than a simple daemon process
- Needs to be controlled by a remote operator
- Implements subservers to manage specific resources
- Does not put itself in the background.

A few subsystem examples are ypserv, ntsd, qdaemon, inetd, syslogd, and sendmail.

Note: See each specific subsystem for details of its SRC capabilities.

Use the **lssrc -a** command to list active and inactive subsystems on your system.

Subsystem Group

A *subsystem group* is a group of any specified subsystems. Grouping subsystems together allows the control of several subsystems at one time. A few subsystem group examples are TCP/IP, SNA Services, Network Information System (NIS), and Network File Systems (NFS).

Subserver

A *subserver* is a program or process that belongs to a subsystem. A subsystem can have multiple subservers and is responsible for starting, stopping, and providing status of subservers. Subservers can be defined only for a subsystem with a communication type of IPC message queues and sockets. Subsystems using signal communications do not support subservers.

Subservers are started when their parent subsystems are started. If you try to start a subserver and its parent subsystem is not active, the **startsrc** command starts the subsystem as well.

SRC Hierarchy

The System Resource Controller hierarchy begins with the operating system followed by a subsystem group (such as **tcpip**), which contains a subsystem (such as the **inetd** daemon), which in turn can own several subservers (such as the **ftp** daemon and the **finger** command).

List of SRC Administration Commands

The following is a list of SRC Administration commands:

srcmstr daemon	Starts the System Resource Controller
startsrc command	Starts a subsystem, subsystem group, or subserver
stopsrc command	Stops a subsystem, subsystem group, or subserver
refresh command	Refreshes a subsystem
traceson command	Turns on tracing of a subsystem, a group of subsystems, or a subserver
tracesoff command	Turns off tracing of a subsystem, a group of subsystems, or a subserver
lssrc command	Gets status on a subsystem.

Chapter 15. System Accounting

This chapter provides an overview of the system accounting utility, which allows you to collect and report on individual and group use of various system resources.

The topics covered are:
- “Accounting Overview”
- “Collecting and Reporting System Data”
- “Collecting Accounting Data” on page 158
- “Reporting Accounting Data” on page 159
- “Accounting Commands” on page 161
- “Accounting Files” on page 162.

Accounting Overview

This accounting information can be used to bill users for the system resources they utilize, and to monitor selected aspects of the system operation. To assist with billing, the accounting system provides the resource-usage totals defined by members of the adm group, and, if the **chargefee** command is included, factors in the billing fee.

The accounting system also provides data to assess the adequacy of current resource assignments, set resource limits and quotas, forecast future needs, and order supplies for printers and other devices.

The following information should help you understand how to implement the accounting utility in your system:
- “Collecting and Reporting System Data”
- “Collecting Accounting Data” on page 158
- “Reporting Accounting Data” on page 159
- “Accounting Commands” on page 161
- “Accounting Files” on page 162.

Collecting and Reporting System Data

For data to be collected automatically, a member of the adm group needs to follow the procedures described in ″Setting Up an Accounting System″. These procedures enable the **cron** daemon to run the commands that generate data on:
- The amount of time each user spends logged in to the system
- Usage of the processing unit, memory, and I/O resources
- The amount of disk space occupied by each user’s files
- Usage of printers and plotters
- The number of times a specific command is given.

The system writes a record of each session and process after they are completed. These records are converted into total accounting (tacct) records arranged by user and merged into a daily report. Periodically, the daily reports are combined to produce totals for the defined fiscal period. Methods for collecting and reporting the data and the various accounting commands and files are discussed in the following sections.

Although most of the accounting data is collected and processed automatically, a member of the adm group can enter certain commands from the keyboard to obtain specific information. These commands are discussed in “Keyboard Commands” on page 162.

Collecting Accounting Data

There are several types of accounting data:
- “Connect-Time Accounting”
- “Process Accounting”
- “Disk-Usage Accounting” on page 159
- “Printer-Usage Accounting” on page 159
- “Fee Accounting” on page 159

They are described in the following sections.

Connect-Time Accounting

Connect-time data is collected by the **init** command and the **login** command. When you log in, the **login** program writes a record in the **/etc/utmp** file. This record includes your user name, the date and time of the login, and the login port. Commands, such as **who**, use this file to find out which users are logged into the various display stations. If the **/var/adm/wtmp** connect-time accounting file exists, the **login** command adds a copy of this login record to it.

When your login program ends (normally when you log out), the **init** command records the end of the session by writing another record in the **/var/adm/wtmp** file. Logout records differ from login records in that they have a blank user name. Both the login and logout records have the form described in the **utmp.h** file.

The **acctwtmp** command also writes special entries in the **/var/adm/wtmp** file concerning system shutdowns and startups.

For more information, see “Connect-Time Reports” on page 159.

Process Accounting

The system collects data on resource usage for each process as it runs, including:
- The user and group numbers under which the process runs
- The first eight characters of the name of the command
- A 64-bit numeric key representing the Workload Manager class that the process belongs to
- The elapsed time and processor time used by the process
- Memory use
- The number of characters transferred
- The number of disk blocks read or written on behalf of the process

The **accton** command records these data in a specified file, usually the **/var/adm/pacct** file.

Related commands are the **startup** command, the **shutacct** command, the **dodisk** command, the **ckpacct** command, and the **turnacct** command.

For more information, see “Reporting Accounting Data” on page 159.

Disk-Usage Accounting

Much accounting information is collected as the resources are consumed. The **dodisk** command, run as specified by the **cron** daemon, periodically writes disk-usage records for each user to the **/var/adm/acct/nite/dacct** file. To accomplish this, the **dodisk** command calls other commands. Depending upon the thoroughness of the accounting search, the **diskusg** command or the **acctdusg** command can be used to collect data. The **acctdisk** command is used to write a total accounting record. The total accounting record, in turn, is used by the **acctmerg** command to prepare the daily accounting report.

The **dodisk** command charges a user for the links to files found in the user's login directory and evenly divides the charge for each file between the links. This distributes the cost of using a file over all who use it and removes the charges from users when they relinquish access to a file.

For more information, see "Disk-Usage Accounting Report" on page 160.

Printer-Usage Accounting

The collection of printer-usage data is a cooperative effort between the **enq** command and the queuing daemon. The **enq** command enqueues the user name, job number, and the name of the file to be printed. After the file is printed, the **qdaemon** command writes an ASCII record to a file, usually the **/var/adm/qacct** file, containing the user name, user number, and the number of pages printed. You can sort these records and convert them to total accounting records.

For more information, see "Printer-Usage Accounting Report" on page 160.

Fee Accounting

You can enter the **chargefee** command to produce an ASCII total accounting record in the **/var/adm/fee** file. This file will be added to daily reports by the **acctmerg** command.

For more information, see "Fee Accounting Report" on page 160.

Reporting Accounting Data

After the various types of accounting data are collected, the records are processed and converted into reports.

Accounting commands automatically convert records into scientific notation when numbers become large. A number is represented in scientific notation in the following format:

*Base***e+***Exp*

*Base***e-***Exp*

which is the number equal to the *Base* number multiplied by 10 to the **+***Exp* or **-***Exp* power. For example, the scientific notation 1.345e+9 is equal to 1.345×10^{9}, or 1,345,000,000. And the scientific notation 1.345e-9 is equal to 1.345×10^{-9} or, 0.000000001345.

Connect-Time Reports

The **runacct** command calls two commands, **acctcon1** and **acctcon2**, to process the login, logout, and system-shutdown records that collect in the**/var/adm/wtmp** file. The **acctcon1** command converts these records into session records and writes them to the **/var/adm/acct/nite/lineuse** file. The **acctcon2** command then converts the session records into a total accounting record, **/var/adm/logacct**, that the **acctmerg** command adds to daily reports.

If you run the **acctcon1** command from the command line, you must include the **-l** flag to produce the line-use report, **/var/adm/acct/nite/lineuse**. To produce an overall session report for the accounting period, **/var/adm/acct/nite/reboots**, use the **acctcon1** command with the **-o** flag.

The **lastlogin** command produces a report that gives the last date on which each user logged in.

Process Accounting Reports

Two commands process the billing-related data that was collected in the **/var/adm/pacct** or other specified file. The **acctprc1** command translates the user ID into a user name and writes ASCII records containing the chargeable items (prime and non-prime CPU time, mean memory size, and I/O data). The **acctprc2** command transforms these records into total accounting records that are added to daily reports by the **acctmerg** command.

Process accounting data also provides information that you can use to monitor system resource usage. The **acctcms** command summarizes resource use by command name. This provides information on how many times each command was run, how much processor time and memory was used, and how intensely the resources were used (also known as the *hog factor*). The **acctcms** command produces long-term statistics on system utilization, providing information on total system usage and the frequency with which commands are used.

The **acctcom** command handles the same data as the **acctcms** command, but provides detailed information about each process. You can display all process accounting records or select records of particular interest. Selection criteria include the load imposed by the process, the time period when the process ended, the name of the command, the user or group that invoked the process, the name of the WLM class the proces belonged to, and the port at which the process ran. Unlike other accounting commands, **acctcom** can be run by all users.

Disk-Usage Accounting Report

The disk-usage records collected in the **/var/adm/acct/nite/dacct** file are merged into the daily accounting reports by the **acctmerg** command.

Printer-Usage Accounting Report

The ASCII record in the **/var/adm/qacct** file can be converted to a total accounting record to be added to the daily report by the **acctmerg** command.

Fee Accounting Report

If you used the **chargefee** command to charge users for services such as file restores, consulting, or materials, an ASCII total accounting record is written in the **/var/adm/fee** file. This file is added to the daily reports by the **acctmerg** command.

Daily Reports

Raw accounting data on connect-time, processes, disk usage, printer usage, and fees to charge are merged into daily reports by the **acctmerg** command. Called by the **runacct** command as part of its daily operation, the **acctmerg** command produces the following:

/var/adm/acct/nite/dacct	An intermediate report that is produced when one of the input files is full.
/var/adm/acct/sum/tacct	A cumulative total report in **tacct** format. This file is used by the **monacct** command to produce the ASCII monthly summary.

The **acctmerg** command can convert records between ASCII and binary formats and merge records from different sources into a single record for each user.

Monthly Report

Called by the **cron** daemon, the **monacct** command produces the following:

/var/adm/acct/fiscal	A periodic summary report produced from the **/var/adm/acct/sum/tacct** report by the **monacct** command. The **monacct** command can be configured to run monthly or at the end of a fiscal period.

Accounting Commands

The accounting commands function several different ways. Some commands:

- Collect data or produce reports for a specific type of accounting: connect-time, process, disk usage, printer usage, or command usage.
- Call other commands. For example, the **runacct** command, which is usually run automatically by the **cron** daemon, calls many of the commands that collect and process accounting data and prepare reports. To obtain automatic accounting, you must first configure the **cron** daemon to run the **runacct** command. See the **crontab** command for more information about how to configure the **cron** daemon to submit commands at regularly scheduled intervals.
- Perform maintenance functions and ensure the integrity of active data files.
- Enable members of the adm group to perform occasional tasks, such as displaying specific records, by entering a command at the keyboard.
- Enable a user to display specific information. There is only one user command, the **acctcom** command, which displays process accounting summaries.

Commands That Run Automatically

Several commands usually run by the **cron** daemon automatically collect accounting data.

runacct	Handles the main daily accounting procedure. Normally initiated by the **cron** daemon during non-prime hours, the **runacct** command calls several other accounting commands to process the active data files and produce command and resource usage summaries, sorted by user name. It also calls the **acctmerg** command to produce daily summary report files, and the **ckpacct** command to maintain the integrity of the active data files.
ckpacct	Handles **pacct** file size. It is advantageous to have several smaller **pacct** files if you must restart the **runacct** procedure after a failure in processing these records. The **ckpacct** command checks the size of the **/var/adm/pacct** active data file, and if the file is larger than 500 blocks, the command invokes the **turnacct switch** command to turn off process accounting temporarily. The data is transferred to a new **pacct** file, **/var/adm/pacct** *x*. (*x* is an integer that increases each time a new **pacct** file is created.) If the number of free disk blocks falls below 500, the **ckpacct** command calls the **turnacct off** command to turn off process accounting.
dodisk	Calls the **acctdisk** command and either the **diskusg** command or the **acctdusg** command to write disk-usage records to the **/var/adm/acct/nite/dacct** file. This data is later merged into the daily reports.
monacct	Produces a periodic summary from daily reports.
sa1	Collects and stores binary data in the **/var/adm/sa/sa** *dd* file, where *dd* is the day of the month.
sa2	Writes a daily report in the **/var/adm/sa/sa***dd* file, where *dd* is the day of the month. The command removes reports from the **/var/adm/sa/sa***dd* file that have been there longer than one week.

Other commands are run automatically by procedures other than the **cron** daemon:

startup	When added to the **/etc/rc** file, the **startup** command initiates startup procedures for the accounting system.
shutacct	Records the time accounting was turned off by calling the **acctwtmp** command to write a line to **/var/adm/wtmp** file. It then calls the **turnacct off** command to turn off process accounting.

Keyboard Commands

A member of the adm group can enter the following commands from the keyboard:

ac	Prints connect-time records. This command is provided for compatibility with Berkeley Software Distribution (BSD) systems.
acctcom	Displays process accounting summaries. This command is also available to users.
acctcon1	Displays connect-time summaries. Either the **-l** flag or the **-o** flag must be used.
accton	Turns process accounting on and off.
chargefee	Charges the user a predetermined fee for units of work performed. The charges are added to the daily report by the **acctmerg** command.
fwtmp	Converts files between binary and ASCII formats.
last	Displays information about previous logins. This command is provided for compatibility with BSD systems.
lastcomm	Displays information about the last commands that were executed. This command is provided for compatibility with BSD systems.
lastlogin	Displays the time each user last logged in.
pac	Prepares printer/plotter accounting records. This command is provided for compatibility with Berkeley Software Distribution (BSD) systems.
prctmp	Displays a session record.
prtacct	Displays total accounting files.
sa	Summarizes raw accounting information to help manage large volumes of accounting information. This command is provided for compatibility with BSD systems.
sadc	Reports on various local system actions, such as buffer usage, disk and tape I/O activity, TTY device activity counters, and file access counters.
sar	Writes to standard output the contents of selected cumulative activity counters in the operating system. The **sar** command reports only on local activities.
time	Prints real time, user time, and system time required to run a command.
timex	Reports in seconds the elapsed time, user time, and run time.

Accounting Files

The two main accounting directories are the **/usr/sbin/acct** directory, where all the C language programs and shell procedures needed to run the accounting system are stored, and the **/var/adm** directory, which contains the data, report and summary files.

The accounting data files belong to members of the adm group, and all active data files (such as **wtmp** and **pacct**) reside in the adm home directory **/var/adm.**

Data Files

Files in the **/var/adm** directory are:

/var/adm/diskdiag	Diagnostic output during the running of disk accounting programs
/var/adm/dtmp	Output from the **acctdusg** command
/var/adm/fee	Output from the **chargefee** command, in ASCII **tacct** records
/var/adm/pacct	Active process accounting file
/var/adm/wtmp	Active process accounting file
/var/adm/Spacct *.mmdd*	Process accounting files for *mmdd* during the execution of the **runacct** command.

Report and Summary Files

Report and summary files reside in a **/var/adm/acct** subdirectory. You must create the following subdirectories before the accounting system is enabled. See ″Setting Up an Accounting System″ for more information.

/var/adm/acct/nite	Contains files that the **runacct** command reuses daily
/var/adm/acct/sum	Contains the cumulative summary files that the **runacct** command updates daily
/var/adm/acct/fiscal	Contains the monthly summary files that the **monacct** command creates.

runacct Command Files

The following report and summary files, produced by the **runacct** command, are of particular interest:

/var/adm/acct/nite/lineuse	Contains usage statistics for each terminal line on the system. This report is especially useful for detecting bad lines. If the ratio between the number of logouts and logins exceeds about 3 to 1, there is a good possibility that a line is failing.
/var/adm/acct/nite/daytacct	Contains the total accounting file for the previous day.
/var/adm/acct/sum/tacct	Contains the accumulation of each day's **nite/daytacct** file and can be used for billing purposes. The **monacct** command restarts the file each month or fiscal period.
/var/adm/acct/sum/cms	Contains the accumulation of each day's command summaries. The **monacct** command reads this binary version of the file and purges it. The ASCII version is **nite/cms**.
/var/adm/acct/sum/daycms	Contains the daily command summary. An ASCII version is stored in **nite/daycms**.
/var/adm/acct/sum/loginlog	Contains a record of the last time each user ID was used.
/var/adm/acct/sum/rprt *mmdd*	This file contains a copy of the daily report saved by the **runacct** command.

Files in the /var/adm/acct/nite Directory

active	Used by the **runacct** command to record progress and print warning and error messages. The file **active.** *mmdd* is a copy of the **active** file made by the **runacct** program after it detects an error.
cms	ASCII total command summary used by the **prdaily** command.
ctacct.*mmdd*	Connect total accounting records.
ctmp	Connect session records.
daycms	ASCII daily command summary used by the **prdaily** command.
daytacct	Total accounting records for one day.
dacct	Disk total accounting records, created by the **dodisk** command.
accterr	Diagnostic output produced during the execution of the **runacct** command.
lastdate	Last day the **runacct** executed, in **date +%m%d** format.
lock1	Used to control serial use of the **runacct** command.
lineuse	tty line usage report used by the **prdaily** command.
log	Diagnostic output from the **acctcon1** command.
log*mmdd*	Same as **log** after the **runacct** command detects an error.
reboots	Contains beginning and ending dates from **wtmp**, and a listing of system restarts.
statefile	Used to record the current state during execution of the **runacct** command.
tmpwtmp	**wtmp** file corrected by the **wtmpfix** command.
wtmperror	Contains **wtmpfix** error messages.
wtmperr*mmdd*	Same as **wtmperror** after the **runacct** command detects an error.
wtmp.*mmdd*	Contains previous day's **wtmp** file. Removed during the cleanup of **runacct** command.

Files in the /var/adm/acct/sum Directory

cms	Total command summary file for the current fiscal period, in binary format.
cmsprev	Command summary file without the latest update.
daycms	Command summary file for the previous day, in binary format.

lastlogin	File created by the **lastlogin** command.
pacct.*mmdd*	Concatenated version of all **pacct** files for *mmdd*. This file is removed after system startup by the **remove** command.
rprt*mmdd*	Saved output of the **prdaily** command.
tacct	Cumulative total accounting file for the current fiscal period.
tacctprev	Same as **tacct** without the latest update.
tacct*mmdd*	Total accounting file for *mmdd*.

Files in the /var/adm/acct/fiscal Directory

cms*?*	Total command summary file for the fiscal period, specified by *?*, in binary format
fiscrpt*?*	A report similar to that of the **prdaily** command for fiscal period, specified by *?*, in binary format
tacct*?*	Total accounting file for fiscal period, specified by *?*, in binary format.

Accounting File Formats

Accounting file output and formats are summarized in the following.

wtmp	Produces the active process accounting file. The format of the **wtmp** file is defined in the **utmp.h** file.
ctmp	Produces connect session records. The format is described in the **ctmp.h** file.
pacct*	Produces active process accounting records. The format of the output is defined in the **/usr/include/sys/acct.h** file.
Spacct*	Produces process accounting files for *mmdd* during the running of the **runacct** command. The format of these files is defined in the **sys/acct.h** file.
daytacct	Produces total accounting records for one day. The format of the file is defined in the **tacct** file format.
sum/tacct	Produces binary file that accumulates each day's command summaries. The format of this file is defined in the **/usr/include/sys/acct.h** header file.
ptacct	Produces concatenated versions of **pacct** files. The format of these files are defined in the **tacct** file.
ctacct	Produces connect total accounting records. The output of this file is defined in the **tacct** file.
cms	Produces total accounting command summary used by the **prdaily** command, in binary format. The ASCII version is **nite/cms**.
daycms	Daily command summary used by the **prdaily** command, in binary format. The ASCII version is **nite/daycms**.

Chapter 16. Web-based System Manager

Web-based System Manager is a graphical user interface (GUI) application for performing system administration tasks such as viewing users and groups, installed software, and printers and devices; managing logical volumes, users and groups, and resources; mounting and unmounting file systems; configuring the network; and many other tasks. You can manage systems from a locally attached display or remotely from another system or personal computer equipped with a web browser.

The Web-based System Manager GUI provides point-and-click control of objects, which provides an alternative to learning and using commands or SMIT.

For a detailed explanation and procedures for using Web-based System Manager, *AIX 5L Version 5.1 Web-based System Manager Administration Guide*.

Chapter 17. System Management Interface Tool

This chapter provides a brief overview of the System Management Interface Tool (SMIT) panels.

System Management Interface Tool (SMIT) Overview

Although “Chapter 16. Web-based System Manager” on page 165 is the primary interface for system management, the System Management Interface Tool (SMIT) provides an alternative, natural-language, task-oriented interface. The SMIT facility runs in two interfaces, ASCII (nongraphical) or AIXwindows (graphical).

SMIT steps you through the desired task with the use of menus, selectors, and dialogs, thereby freeing you from the details of complex command syntax, valid parameter values, and system command spelling. In addition, SMIT creates log files that you can use to duplicate system configuration or to learn how to use specific commands.

Note: In a networked environment, you can use Distributed SMIT as discussed in the *Distributed SMIT 2.2 for AIX: Guide and Reference*.

In the SMIT interface, main menu selections lead to submenus, helping to narrow the scope of choice to a particular task. To skip the main menu and directly access a submenu or dialog, you can use the **smit** command with a *Fast Path* parameter.

To learn more about SMIT, you can:

- Start SMIT, then select **Using SMIT (Information Only)** from the SMIT Main Menu.
- In the SMIT dialogs, select **On Context (Ctrl+F1)** from the Help menu and move the cursor over the particular menu item or field about which you want more information.

The following table lists some basic SMIT tasks:

Basic SMIT Tasks			
Task	*SMIT Fast Path*	*Selection (ASCII)*	*Selection (AIXwindows)*
Enter SMIT	**smit**		
Exit SMIT		F12	F12 or Exit SMIT option from Exit menu
Show command		F6	F6 or Command option from Show menu
Show fast path		F8	F8 or FastPath option from Show menu

Chapter 18. The Common Desktop Environment

With the Common Desktop Environment (CDE), you can access networked devices and tools without having to be aware of their location. You can exchange data across applications by simply dragging and dropping objects.

System administrators will find many tasks that previously required complex command line syntax can now be done more easily and similarly from platform to platform. They can also maximize their investment in existing hardware and software by configuring centrally and distributing applications to users. They can centrally manage the security, availability, and interoperability of applications for the users they support.

For additional information and procedures for using CDE, see "Chapter 18. The Common Desktop Environment" in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

Chapter 19. Documentation Library Service

The Documentation Library Service allows you to read, search, and print online HTML documents. It provides a library application that displays in your Web browser. Within the library application, you can click on links to open documents for reading. You can also type words into the search form in the library application. The library service searches for the words and presents a search results page that contains links that lead to the documents that contain the target words. Starting with AIX 5.1 you can also dowload printable versions of books by using the ″Print″ button in the Library Service GUI.

To launch the library application, type the **docsearch** command or select the CDE help icon, click on the Front Panel Help icon, then click on the Documentation Library icon.

The documentation search service allows you to access only the documents on your documentation server that are registered with the library and that have been indexed. You cannot read or search the Internet or all the documents on your computer. Indexing creates a specially compressed copy of a document or collection of documents. It is this index that is searched rather than the original documents. This technique provides significant performance benefits. When a phrase you are searching for is found in the index, the documentation search service presents a results page that contains links to select and open the document that contains the search phrase.

You can register the HTML documents of your company into the library so that all users can access and search the documents using the library application. Before your documents can be searched, you must create indexes of the documents. For more information on adding your own documents to the library, see Documents and Indexes.

With the exception of the search engine, the library components are installed with the base operating system. To use the library service, it must be configured. You can configure a computer to be a documentation server and install documents on that computer; or you can configure a computer to be a client that gets all of its documents from a documentation server. If the computer is to be a documentation server, the search engine and documentation must also be manually installed.

The library service should be fully configured because it is the library service for the operating system manuals and the Web-based System Manager documentation. Even if you do not need the operating system manuals, is is best to configure the documentation library service because it is expected that other applications might use it as the library function for their own online documentation. For instructions on how to install and configure the documentation library service, see Installing and Configuring the Documentation Library Service and Online Documentation in *AIX 5L Version 5.1 Installation Guide*.

For additional information, see the following topics in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*:

- Changing the Configuration
- Working with Documents and Indexes
- Advanced Topics
- Problem Determination.

Chapter 20. Power Management

Note: The information in this section is specific to POWER-based.

Power Management is a technique that enables hardware and software to minimize system power consumption. It is especially important for products that operate with batteries and desktop products.

Power Management tasks can be performed by using:
- The System Management Interface Tool (SMIT)
- Commands
- The Power Management application.

For a list of the Power Management tasks and procedures that you can perform, see Power Management in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

The following section contains important information for all users of Power Management.

Power Management Limitation Warnings

Note: The information in this section is specific to POWER-based.

Users of Power Management need to be aware of the following limitations:

Changing configuration during suspend/hibernation	Altering the system configuration, such as memory size, devices, and so on, while the system is in the suspend or hibernation state can cause unpredictable results. This could cause loss of data, file system corruption, system crashes, or a failure to resume from the suspend or hibernation state.

Non-PM-aware device drivers	If a device driver is installed that is not Power Management-aware, unpredictable results could occur when resuming from suspend or hibernation. If a non-PM-aware device driver is to be installed, the suspend and hibernation states must never be used. The following command can be run with root authority to disable the suspend and hibernation states effective on the next system startup.
	
	The following command frees the hibernation logical volume and disallows future selections of the suspend or hibernation states:
	`/usr/lib/boot/disable_hibernation`
	
	If you want to reenable these functions, the following command enables the suspend and hibernation states effective on the next system boot, provided the hardware platform supports such states:
	`/usr/lib/boot/enable_hibernation`

Booting from CD-ROM or other media after hibernation	Accessing the rootvg from maintenance mode such as CD-ROM startup when a valid hibernation image exists can result in loss of data and file system corruption. Maintenance modes should only be used after normal system shutdown or power-off, not after a hibernation power-off.

Network connections during suspend/hibernation	Network connections are disconnected during the suspend and hibernation states. These connections might have to be re-established by the user after resuming. Because locally cached data is not available to other nodes on the network during this time and network activity cannot be monitored by the local node during this time, do not use the suspend and hibernation states when using network interfaces such as TCP/IP, NFS, AFS, DCE, SNA, OSI, NetWare, NetBIOS, and so on.
	
	The following command frees the hibernation logical volume and disallows future selections of the suspend or hibernation states:
	`/usr/lib/boot/disable_hibernation`
	
	If you want to reenable these functions, the following command enables the suspend and hibernation states effective on the next system startup, provided the hardware platform supports such states:
	`/usr/lib/boot/enable_hibernation`

Power Button Behavior	When Power Management is enabled, the power button is software controlled. If there is some sort of system problem, the software necessary to make the requested Power Management state transition using the power switch might not be able to run. In such a situation, or whenever necessary, it is always possible to turn off the power immediately by pressing the power button three times quickly (within a two second period). This overrides whatever state transition has been selected for the power switch, and requires a full restart. In addition, if the Power Management daemon (**/usr/bin/pmd**) is never started (by an entry in **/etc/inittab by default**), the power switch acts as if there were no power management. A single button press turns off the system. If **/usr/bin/pmd** is started and then stopped, the first two button presses are ignored, and the third turns off the system. These button presses can be over any period of time as long as **/usr/bin/pmd** is not restarted.

Chapter 21. Devices

Devices include hardware components such as, printers, drives, adapters, buses, and enclosures, as well as pseudo-devices, such as the error special file and null special file. This chapter provides an overview of the following methods used by the operating system to manage these devices:

- “Device Nodes”
- “Location Codes” on page 176
- “PCI Hot Plug Management” on page 180.

Device Nodes

Devices are organized into clusters known as *nodes*. Each node is a logical subsystem of devices, where lower-level devices have a dependency on upper-level devices in child-parent relationships. For example, the system node is the highest of all nodes and consists of all the physical devices in the system. The system device is the top of the node and below that are the bus and adapters that have a dependency on the higher-level system device. At the bottom of the hierarchy are devices to which no other devices are connected. These devices have dependencies on all devices above them in the hierarchy.

At startup time, parent-child dependencies are used to configure all devices that make up a node. Configuration occurs from the top node down and any device having a dependency on a higher-level device is not configured until the higher-level device is configured.

Device Classes

Managing devices requires the operating system to comprehend what device connections are allowed. The operating system classifies devices hierarchically into three groups:

- Functional classes
- Functional subclasses
- Device types

Functional classes consist of devices that perform the same function. Printers, for example, comprise a functional class. Functional classes are grouped into subclasses according to certain device similarities. For example, printers have a serial or parallel interface. Serial printers are one subclass and parallel printers are another. Device types are classified according to their model and manufacturer.

Device classes define valid parent-child connections for the operating system. The hierarchy defines the possible subclasses that can be connected for each of the possible child connection locations. For example, the term RS-232 8-port adapter specifies that only devices belonging to the RS-232 subclass can be connected to any of the eight ports of the adapter.

Device classes and their hierarchical dependencies are maintained in an Object Data Manager (ODM) Device Configuration database.

Device Configuration Database

Device information is contained in a predefined database or a customized database that makes up the device configuration database.

The predefined database contains configuration data for all possible devices supported by the system. The hierarchical device class information is contained in this database.

The customized database contains configuration data for all currently defined and configured devices in the system. A record is kept of each device currently connected to your system.

The Configuration Manager is a program that automatically configures devices on your system during system startup and run time. The Configuration Manger uses the information from the predefined and customized databases during this process, and updates the customized database afterwards.

Device States

Devices that are connected to the system can be in one of four states:

Undefined	The device is unknown to the system.
Defined	Specific information about the device is recorded in the customized database, but it is unavailable to the system.
Available	A defined device is coupled to the operating system, or the defined device is configured.
Stopped	The device is unavailable but remains known by its device driver.

If a tty device and a printer alternately use the same tty connector, both a tty device and a printer are defined on the same parent and port in the device configuration database. Only one of these devices can be configured at a time. When the tty connector is configured, the printer specific setup information is retained until it is configured again. The device is not removed; it is in the defined state. Maintaining a device in defined state retains customized information for a device that is not currently in use, either before it is first made available or while it is temporarily removed from the system.

If a device driver exists for a device, the device can be made available through the device driver.

Some devices, in particular TCP/IP pseudo-devices, need the stopped state.

Device Management

You can use the Web-based System Manager Devices application, SMIT, or operating system commands to perform device management tasks such as deleting or adding a device.

Location Codes

The *location code* is a path from the CPU drawer or system unit through the adapter, signal cables, and the asynchronous distribution box (if there is one) to the device or workstation. This code is another way of identifying physical devices.

The location code consists of up to four fields of information depending on the type of device. These fields represent drawer, slot, connector, and port. Each of these fields consists of two characters.

The location code of a drawer consists of only the drawer field and is simply a two-character code. The location code of an adapter consists of the drawer and slot fields and has the format `AA-BB`, where `AA` corresponds to the drawer location and `BB` indicates the bus and slot that contains the adapter. Other devices have location codes of formats `AA-BB-CC` or `AA-BB-CC-DD`, where `AA-BB` is the location code of the adapter to which the device is connected, `CC` corresponds to the connecter on the adapter to which the device is connected, and `DD` corresponds to a port number or SCSI device address.

For information on finding the labels with the location codes on the hardware, see your operator guide.

Adapter Location Codes

The location code for an adapter consists of two pairs of digits with the format `AA-BB`, where `AA` identifies the location code of the drawer containing the adapter and `BB` identifies the I/O bus and slot containing the card.

A value of `00` for the `AA` field means that the adapter is located in the CPU drawer or system unit, depending on the type of system. Any other value for the `AA` field indicates that the card is located in an I/O expansion drawer. In this case, the `AA` value identifies the I/O bus and slot number in the CPU drawer that contains the asynchronous expansion adapter. The first digit identifies the I/O bus with `0` corresponding to the standard I/O bus and `1` corresponding to the optional I/O bus. The second digit identifies the slot number on the indicated I/O bus.

The first digit of the `BB` field identifies the I/O board containing the adapter card. If the card is in the CPU drawer or system unit, this digit is `0` for the standard I/O bus and `1` for the optional I/O bus. If the card is in an I/O expansion drawer, this digit is `0`. The second digit identifies the slot number on the indicated I/O bus (or slot number in the I/O expansion drawer) that contains the card.

A location code of `00-00` is used to identify the standard I/O board.

Examples:

`00-05`	Identifies an adapter card in slot 5 of the standard I/O board and is located in either the CPU drawer or the system unit, depending on the type of system.
`00-12`	Identifies an adapter in slot 2 of the optional I/O bus and is located in the CPU drawer.
`18-05`	Identifies an adapter card located in slot 5 of an I/O expansion drawer. The drawer is connected to the asynchronous expansion adapter located in slot 8 of the optional I/O bus in the CPU drawer.

Printer and Plotter Location Codes

Location codes of `00-00-S1-00` or `00-00-S2-00` indicate the printer, plotter, or tty device is connected to the standard I/O board serial ports `s1` or `s2`. A location code of `00-00-0P-00` indicates the parallel printer is connected to the standard I/O board parallel port.

Any other location code indicates that the printer, plotter, or tty device is connected to an adapter card other than the standard I/O board. For these printers, plotters, and tty devices, the location code format is `AA-BB-CC-DD`, where `AA-BB` indicates the location code of the controlling adapter.

`AA`	A value of `00` for the `AA` field indicates the adapter card is located in the CPU drawer or system unit depending on the type of system. Any other value for the `AA` field indicates the card is located in an I/O expansion drawer; in which case, the first digit identifies the I/O bus and the second digit identifies the slot number on the bus, in the CPU drawer, that contains the asynchronous expansion adapter to which the I/O expansion drawer is connected..
`BB`	The first digit of the `BB` field identifies the I/O bus containing the adapter card. If the card is in the CPU drawer or system unit, this digit is `0` for the standard I/O bus and `1` for the optional I/O bus. If the card is in an I/O expansion drawer, this digit is `0`. The second digit identifies the slot number on the I/O bus (or slot number in the I/O expansion drawer) that contains the card.
`CC`	The `CC` field identifies the connector on the adapter card where the asynchronous distribution box is connected. Possible values are `01`, `02`, `03`, and `04`.
`DD`	The `DD` field identifies the port number on the asynchronous distribution box where the printer, plotter, or tty device is attached.

tty Location Codes

Location codes of `00-00-S1-00` or `00-00-S2-00` indicate the tty device is connected to the standard I/O serial ports `s1` or `s2`.

Any other location code indicates the tty device is connected to an adapter card other than the standard I/O board. For these devices, the location code format is `AA-BB-CC-DD`, where `AA-BB` indicates the location code of the controlling adapter card.

AA	A value of 00 for the AA field indicates the adapter card is located in the CPU drawer or system unit depending on the type of system. Any other value for the AA field indicates the card is located in an I/O expansion drawer. In this case, the first digit identifies the I/O bus, and the second digit identifies the slot number on the bus in the CPU drawer that contains the asynchronous expansion adapter where the I/O expansion drawer is connected.
BB	The first digit of the BB field identifies the I/O bus containing the adapter card. If the card is in the CPU drawer or system unit, this digit will be 0 for the standard I/O bus and 1 for the optional I/O bus. If the card is in an I/O expansion drawer this digit is 0. The second digit identifies the slot number on the I/O bus (or slot number in the I/O expansion drawer) that contains the card.
CC	The CC field identifies the connector on the adapter card where the asynchronous distribution box is connected. Possible values are 01, 02, 03, and 04.
DD	The DD field identifies the port number on the asynchronous distribution box where the tty device is attached.

SCSI Device Location Codes

These location codes apply to all SCSI devices including:
- CD-ROMs
- Disks
- Initiator devices
- Read/write optical drives
- Tapes
- Target mode

The location code format is AA-BB-CC-S,L. The AA-BB fields identify the location code of the SCSI adapter controlling the SCSI device.

AA	A value of 00 for the AA field indicates the controlling adapter card is located in the CPU drawer or system unit, depending on the type of system.
BB	The BB field identifies the I/O bus and slot containing the card. The first digit identifies the I/O bus. It is 0 for the standard I/O bus and 1 for the optional I/O bus. The second digit is the slot on the indicated I/O bus containing the card. A value of 00 for the BB field indicates the standard SCSI controller.
CC	The CC field identifies the SCSI bus of the card that the device is attached to. For a card that provides only a single SCSI bus, this field is set to 00. Otherwise, a value of 00 indicates a device attached to the internal SCSI bus of the card, and a value of 01 indicates a device attached to the external SCSI bus of the card.
S,L	The S,L field identifies the SCSI ID and logical unit number (LUN) of the SCSI device. The S value indicates the SCSI ID and the L value indicates the LUN.

Direct-Bus-Attached Disk Location Codes

For a direct-attached disk device, the location code format is AA-BB. The AA field is a value of 00, that indicates the disk is located in the system unit. The BB field indicates the I/O bus and slot number where the disk is attached. The first digit is always 0, which indicates the disk is attached to the standard I/O bus. The second digit identifies the slot number on the standard I/O bus to which the disk is attached.

Serial-Linked Disk Location Codes

The location code for serial-linked disk drives is of the format AA-BB-CC-DD, where AA-BB indicates the location code of the controlling adapter card.

The individual fields are interpreted as follows:

AA	A value of 00 for the AA field indicates the controlling adapter card is located in the CPU drawer or system unit, depending on the type of system.

`BB`	The `BB` field identifies the I/O bus and slot containing the card. The first digit identifies the I/O bus. It is `0` for the standard I/O bus and `1` for the optional I/O bus. The second digit is the slot on the indicated I/O bus that contains the card.
`CC`	The `CC` field identifies the connector on the adapter card where the controller drawer is attached. Possible values are `00`, `01`, `02`, and `03`.
`DD`	The `DD` field identifies the logical unit number (LUN) of the disk. This corresponds to the slot in the drawer where the disk resides.

Diskette Drive Location Codes

Diskette drives have location codes of either `00-00-0D-01` or `00-00-0D-02`, indicating that they are attached to the standard I/O planar diskette ports `0` or `1`.

Dials/LPFKeys Location Codes

For a Dials/LPFKeys device attached to a graphics input adapter, the location code format is `AA-BB-CC.`

The individual fields are interpreted as follows:

`AA`	A value of `00` for the `AA` field indicates the controlling adapter card is located in the CPU drawer or system unit, depending on the type of system.
`BB`	The `BB` field identifies the I/O bus and slot containing the card. The first digit identifies the I/O bus. It is `0` for the standard I/O bus and `1` for the optional I/O bus. The second digit is the slot on the indicated I/O bus that contains the card.
`CC`	The `CC` field indicates the card connector where the device is attached. The value is either `01` or `02`, depending on whether the attached device is port `1` or port `2` on the card.

Note: Serially attached Dials/LPFKeys devices do not indicate location codes. This is because these devices are considered to be attached to a tty device. The tty device is specified by the user during Dials/LPFKeys definition.

Multiprotocol Port Location Codes

The location code for a multiprotocol port is of the format `AA-BB-CC-DD` where `AA-BB` indicates the location code of the multiprotocol adapter card.

The individual fields are interpreted as follows:

`AA`	A value of `00` for the `AA` field indicates the multiprotocol adapter card is located in the CPU drawer or system unit, depending on the type of system.
`BB`	The `BB` field identifies the I/O bus and slot containing the card. The first digit identifies the I/O bus. It is `0` for the standard I/O bus and `1` for the optional I/O bus. The second digit is the slot on the indicated I/O bus that contains the card.
`CC`	The `CC` field identifies the connector on the adapter card to which the multiprotocol distribution box is connected. The value is always `01`.
`DD`	The `DD` field identifies the physical port number on the multiprotocol distribution box. Possible values are `00`, `01`, `02`, and `03`.

PCI Hot Plug Management

This section provides an overview of hot plug management, specifically PCI Hot Plug Support for PCI adapters. For information about using the PCI hot plug features and procedures for unconfiguring, adding, removing, and replacing adapters, see Managing Hot Plug Connectors in *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

For information about the commands you can use to display information about PCI hot plug slots and to add, replace, or remove PCI hot plug adapters, see:

- The lsslot command, in the *AIX 5L Version 5.1 Commands Reference, Volume 3*. This command displays a list of all the hot plug slots and their characteristics.
- The drslot command, in the *AIX 5L Version 5.1 Commands Reference, Volume 2*. This command prepares a hot plug slot for adding or removing a hot plug adapter.

Overview

PCI Hot Plug Management consists of user interfaces that allow you to manage hot plug connectors, also known as *dynamic reconfiguration* connectors, or slots. A connector defines the type of slot, for example, PCI. A slot is a unique identifier. Dynamic reconfiguration is the ability of the system to adapt to changes in the hardware and firmware configuration while it is still running.

PCI Hot Plug Support for PCI Adapters is a specific subset of the dynamic reconfiguration function that provides the capability of adding, removing, and replacing PCI adapter cards while the host system is running and without interrupting other adapters in the system. You can also display information about PCI hot plug slots.

Different hot plug connector types require different operations to perform various hot plug management functions. For example, adding a SCSI device requires different operations than when you add a PCI adapter card.

> **Note:** Although PCI hot plug management provides the capability of adding, removing, and replacing PCI adapters without powering off the system or restarting the operating system, not all devices in hot plug slots can be managed in this fashion. For example, the hard disk that makes up the rootvg volume group or the I/O controller to which it is attached cannot be removed or replaced without powering off the system because it is necessary for running the operating system.

Some adapters cannot be hot plugged and should not be removed while the system unit is running. To determine whether an adapter can be hot plugged, refer to the list of supported PCI adapters in the *PCI Adapter Placement Reference*, which is shipped with system units that support PCI hot plug.

Adding a PCI Hot Plug Adapter

You can insert a new PCI adapter into an available PCI slot while the operating system is running. This can be another adapter of the same type currently installed or a different type of PCI adapter. New resources are made available to the operating system and applications without having to restart the operating system. Some reasons for adding an adapter might include:

- Adding additional function or capacity to your existing hardware and firmware.
- Migrating PCI adapters from a system that no longer requires the function provided by those adapters.
- Installing a new system for which adapter cards become available after the initial configuration of optional hardware subsystems, including PCI adapters, and the installation and start of the operating system.

For steps on how to add a PCI hot plug adapter, see Adding a PCI Hot Plug Adapter in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

Removing a PCI Hot Plug Adapter

You can remove a PCI hot plug adapter from its I/O drawer or enclosure without shutting down the operating system or turning off the system power. Removing an adapter makes the resources provided by the adapter unavailable to the operating system and applications. Before you remove the adapter, you must ensure that any resources using the adapter are not in use when you remove it. Some reasons for removing an adapter might include:

- Removing existing I/O subsystems.
- Removing an adapter that is no longer required or is failing and a replacement card is not available.
- Migrating an adapter to another system when the function is no longer required on the system from which it is being removed.

For steps on how to remove a PCI hot plug adapter, see Removing or Replacing a PCI Hot Plug Adapter in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

Replacing a PCI Hot Plug Adapter

You can exchange a defective or failing PCI hot plug adapter with another of the same type without shutting down the operating system or turning off the system power. Because the adapter is of the same type, the existing device driver is able to support the replacement adapter. The replace function retains the configuration information about the replaced adapter and compares this information to the card that replaced it. Device configuration and configuration information about devices below the adapter are utilized in configuring the replacement device.

Some reasons for replacing an adapter might include:

- Temporarily replacing the card to aid in determining a problem or to isolate a failing FRU.
- Replacing a flawed, failing, or intermittently failing adapter with a functional card.
- Replacing a failing redundant adapter in an HACMP or multi-path to storage configuration.

For steps on how to replace a PCI hot plug adapter, see Removing or Replacing a PCI Hot Plug Adapter in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

Resource Utilization

Before you can remove or replace a hot plug device, it must be unconfigured. The associated device driver must free any system resources that it has allocated for the device. This includes unpinning and freeing memory, undefining interrupt and EPOW handlers, releasing DMA and timer resources, and any other required steps. The driver must also ensure that interrupts, bus memory, and bus I/O are disabled on the device.

It is expected that the system administrator will perform the following tasks before and after the removal process:

- Terminate and restore applications, daemons, or processes using the device.
- Unmount and remount filesystems.
- Remove and recreate device definitions and perform other operations necessary to free up a device in use.
- Put the system into a safe state to be serviced.
- Obtain and install any required device drivers.

> **Note:** If you add an adapter using a PCI hot plug replace or add operation, it and its child devices may not be available for specification as a boot device using the **bootlist** command. You may have to restart the machine to make all potential boot devices known to the operating system.

In some cases, the system administrator may also perform the following tasks:

- Prepare a PCI hot plug adapter to be inserted, removed, or replaced.

- Identify slots or PCI adapters that are involved in the hot plug operation.
- Remove or insert PCI hot plug adapters.

Attention: Before you attempt to remove or insert PCI hot plug adapters, refer to the *PCI Adapter Placement Reference*, (shipped with system units that support hot plug), to determine whether your adapter can be hot-swapped. Refer to your system unit documentation for instructions for installing or removing adapters.

Unconfiguring a Device from the System

The remove and replace operations fail unless the device connected to the identified slot has been unconfigured and is in the defined state. You can do this with the rmdev command. Before placing the adapter in the defined state, close all applications that are using the adapter, otherwise, the command will be unsuccessful.

Unconfiguring Communications Adapters

This section provides an overview of the process for unconfiguring PCI communications adapters. This includes Ethernet, Token-ring, FDDI, and ATM adapters. For procedural information, see Unconfiguring Communications Adapters in the *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

If your application is using TCP/IP protocol, you must remove the TCP/IP interface for the adapter from the network interface list before you can place the adapter in the defined state. Use the netstat command to determine whether your adapter is configured for TCP/IP and to check the active network interfaces on your adapter.

An Ethernet adapter can have two interfaces: Standard Ethernet (en*X*) or IEEE 802.3 (et*X*). *X* is the same number in the **entX** adapter name. Only one of these interfaces can be using TCP/IP at a time. For example, Ethernet adapter **ent0** can have en0 and et0 interfaces.

A Token ring adapter can have only one interface: Token-ring (trX). *X* is the same number in the **tokX** adapter name. For example, Token-ring adapter **tok0** has a tr0 interface.

An ATM adapter can have only one atm interface: ATM (atX). *X* is the same number in the **atmX** adapter name. For example, ATM adapter **atm0** has an at0 interface. However, ATM adapters can have multiple emulated clients running over a single adapter.

The ifconfig command removes an interface from the network. The rmdev command unconfigures the PCI device while retaining its device definition in the Customized Devices Object Class. Once the adapter is in the defined state, you can use the drslot command to remove the adapter.

Chapter 22. Tape Drives

Topics included in this chapter are:
- "Tape Drive Attributes"
- "Special Files for Tape Drives" on page 193

Basic tasks for Tape Drives are listed in "Chapter 22. Tape Drives" in *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*.

Tape Drive Attributes

The following describes tape drive attributes you can adjust to meet the needs of your system. The attributes can be displayed or changed using the Web-based System Manager Devices application, SMIT, or commands (in particular, the **lsattr** and the **chdev** commands).

Each type of tape drive only uses a subset of all the attributes.

General Information about Each Attribute

Block Size
The block size attribute indicates the block size to use when reading or writing the tape. Data is written to tape in blocks of data, with inter-record gaps between blocks. Larger records are useful when writing to unformatted tape, because the number of inter-record gaps is reduced across the tape, allowing more data to be written. A value of **0** indicates variable length blocks. The allowable values and default values vary depending on the tape drive.

Device Buffers
Setting the Device Buffers attribute (using the **chdev** command) to `mode=yes` indicates an application is notified of write completion after the data has been transferred to the data buffer of the tape drive, but not necessarily after the data is actually written to the tape. If you specify the `mode=no`, an application is notified of write completion only after the data is actually written to the tape. Streaming mode cannot be maintained for reading or writing if this attribute is set to the `mode=no` value. The default value is `mode=yes`.

With the `mode=no` value, the tape drive is slower but has more complete data in the event of a power outage or system failure and allows better handling of end-of-media conditions.

Extended File Marks
Setting the Extended File Marks attribute (for **chdev**command, the **extfm** attribute) to the `no` value writes a regular file mark to tape whenever a file mark is written. Setting this attribute to the `yes` value writes an extended file mark. For tape drives, this attribute can be set on. The default value is `no`. For example, extended filemarks on 8 mm tape drives use 2.2 MB of tape and can take up to 8.5 seconds to write. Regular file marks use 184 KB and take approximately 1.5 seconds to write.

To reduce errors when you use an 8 mm tape in append mode, use extended file marks for better positioning after reverse operations at file marks.

Retension
Setting the Retension attribute (for the **chdev** command, the **ret** attribute) to `ret=yes` instructs the tape drive to re-tension a tape automatically whenever a tape is inserted or the drive is reset. *Retensioning* a tape means to wind to the end of the tape and then rewind to the beginning of the tape to even the tension throughout the tape. Retensioning the tape can reduce errors, but this action can take several minutes. If you specify the `ret=no` value, the tape drive does not automatically retension the tape. The default value is `yes`.

Density Setting #1, and Density Setting #2

Density Setting #1 (for the **chdev** command, the **density_set_1** attribute) sets the density value that the tape drive writes when using special files **/dev/rmt***, **/dev/rmt*.1**, **/dev/rmt*.2**, and **/dev/rmt*.3**. Density Setting #2 (for **chdev**, the **density_set_2** attribute) sets the density value that the tape drive writes when using special files **/dev/rmt*.4**, **/dev/rmt*.5**, **/dev/rmt*.6**, and **/dev/rmt*.7**. See "Special Files for Tape Drives" on page 193 for more information.

The density settings are represented as decimal numbers in the range **0** to **255**. A zero (**0**) setting selects the default density for the tape drive, which is usually the high density setting of the drive. Specific permitted values and their meanings vary with different types of tape drives. These attributes do not affect the ability of the tape drive to read tapes written in all densities supported by the tape drive. It is customary to set Density Setting #1 to the highest density possible on the tape drive and Density Setting #2 to the second highest density possible on the tape drive.

Reserve Support

For tape drives that use the Reserve attribute (for the **chdev** command, the **res_support** attribute), specifying the value `res_support=yes` causes the tape drive to be reserved on the SCSI bus while it is open. If more than one SCSI adapter shares the tape device, this ensures access by a single adapter while the device is open. Some SCSI tape drives do not support the reserve or release commands. Some SCSI tape drives have a predefined value for this attribute so that reserve or release commands are always supported.

Variable Length Block Size

The Variable Length Block Size attribute (for the **chdev** command, the **var_block_size** attribute) specifies the block size required by the tape drive when writing variable length records. Some SCSI tape drives require that a nonzero block size be specified in their Mode Select data even when writing variable length records. The **Block Size** attribute is set to **0** to indicate variable length records. See the specific tape drive SCSI specification to determine whether or not this is required.

Data Compression

Setting the Data Compression attribute (for the **chdev** command, the **compress** attribute) to `compress=yes` causes the tape drive to be in compress mode, if the drive is capable of compressing data. If so, then the drive writes data to the tape in compressed format so that more data fits on a single tape. Setting this attribute to `no` forces the tape drive to write in native mode (noncompressed). Read operations are not affected by the setting of this attribute. The default setting is `yes`.

Autoloader

Setting the Autoloader attribute (for the **chdev** command, the **autoload** attribute) to `autoload=yes` causes Autoloader to be active, if the drive is so equipped. If so, and another tape is available in the loader, any read or write operation that advances the tape to the end is automatically continued on the next tape. Tape drive commands that are restricted to a single tape cartridge are unaffected. The default setting is `yes`.

Retry Delay

The Retry Delay attribute sets the number of seconds that the system waits after a command has failed before reissuing the command. The system may reissue a failed command up to four times. This attribute applies only to type OST tape drives. The default setting is `45`.

Read/Write Timeout

The Read/Write Timeout or Maximum Delay for a **READ/WRITE** attribute sets the maximum number of seconds that the system allows for a read or write command to complete. This attribute applies only to type OST tape drives. The default setting is `144`.

Return Error on Tape Change

The Return Error on Tape Change or Reset attribute, when set, causes an error to be returned on open when the tape drive has been reset or the tape has been changed. A previous operation to the tape drive must have taken place that left the tape positioned beyond beginning of tape upon closing. The error returned is a `-1` and `errno` is set to `EIO`. Once presented to the application, the error condition is cleared. Also, reconfiguring the tape drive itself will clear the error condition.

Attributes for 2.0 GB 4 mm Tape Drives (Type 4mm2gb)

Block Size
The default value is 1024.

Device Buffers
The general information for this attribute applies to this tape drive type.

Attributes with Fixed Values
If a tape drive is configured as a 2.0 GB 4 mm tape drive, the attributes for Retension, Reserve Support, Variable Length Block Size, Density Setting #1, and Density Setting #2 have predefined values that cannot be changed. The density settings are predefined because the tape drive always writes in 2.0 GB mode.

Attributes for 4.0 GB 4 mm Tape Drives (Type 4mm4gb)

Block Size
The default value is 1024.

Device Buffers
The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2
The user cannot change the density setting of this drive; the device reconfigures itself automatically depending on the Digital Data Storage (DDS) media type installed, as follows:

Media Type	Device Configuration
DDS	Read-only.
DDS ||||	Read/write in 2.0 GB mode only.
DDS2	Read in either density; write in 4.0 GB mode only.
non-DDS	Not supported; cartridge will eject.

Data Compression
The general information for this attribute applies to this tape drive type.

Attributes with Fixed Values
If a tape drive is configured as a 4.0 GB 4 mm tape drive, the Retension, Reserve Support, Variable Length Block Size, Density Setting #1, and Density Setting #2 attributes have predefined values that cannot be changed.

Attributes for 2.3 GB 8 mm Tape Drives (Type 8mm)

Block Size
The default value is 1024. A smaller value reduces the amount of data stored on a tape.

Device Buffers
The general information for this attribute applies to this tape drive type.

Extended File Marks
The general information for this attribute applies to this tape drive type.

Attributes with Fixed Values
If a tape drive is configured as a 2.3 GB 8mm tape drive, the Retension, Reserve Support, Variable Length Block Size, Data Compression, Density Setting #1, and Density Setting #2 attributes have predefined values which cannot be changed. The density settings are predefined because the tape drive always writes in 2.3 GB mode.

Attributes for 5.0GB 8mm Tape Drives (Type 8mm5gb)

Block Size
The default value is 1024. If a tape is being written in 2.3 GB mode, a smaller value reduces the amount of data stored on a tape.

Device Buffers
The general information for this attribute applies to this tape drive type.

Extended File Marks
The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2
The following settings apply:

Setting	Meaning
140	5 GB mode (compression capable)
21	5 GB mode noncompressed tape
20	2.3 GB mode
0	Default (5.0 GB mode)

The default values are 140 for Density Setting #1, and 20 for Density Setting #2. A value of 21 for Density Setting #1 or #2 permits the user to read or write a noncompressed tape in 5 GB mode.

Data Compression
The general information for this attribute applies to this tape drive type.

Attributes with Fixed Values
If a tape drive is configured as a 5.0 GB 8 mm tape drive, the Retension, Reserve Support, and Variable Length Block Size attributes have predefined values which cannot be changed.

Attributes for 20000 MB 8 mm Tape Drives (Self Configuring)

Block Size
The default value is 1024.

Device Buffers
The general information for this attribute applies to this tape drive type.

Extended File Marks
The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2
The drive can read and write data cartridges in 20.0 GB format. During a Read command, the drive automatically determines which format is written on tape. During a Write, the Density Setting determines which data format is written to tape.

The following settings apply:

Setting	Meaning
39	20 GB mode (compression capable)
0	Default (20.0 GB mode)

The default value is 39 for Density Setting #1 and Density Setting #2.

Data Compression
The general information for this attribute applies to this tape drive type.

Attributes with Fixed Values

If a tape drive is configured as a 20.0 GB 8 mm tape drive, the Retension, Reserve Support, and Variable Length Block Size attributes have predefined values which cannot be changed.

Attributes for 35 GB Tape Drives (Type 35gb)

Block Size

The IBM 7205 Model 311 throughput is sensitive to block size. The minimum recommended block size for this drive is 32 K Bytes. Any block size less than 32 K Bytes restricts the data rate (backup/restore time). The following table lists recommended block sizes by command:

Supported Command	Default Block Size (Bytes)	RECOMMENDATION
BACKUP	32 K or 51.2 K (default)	Uses either 32 K or 51.2 K depending on if ″Backup″ is by name or not. No change is required.
TAR	10 K	There is an error in the manual that states a 512 K byte block size. Set the Blocking Parameter to **-N64**.
MKSYSB	See BACKUP	MKSYSB uses the BACKUP Command. No change is required.
DD	n/a	Set the Blocking Parameter to **bs=32K**.
CPIO	n/a	Set the Blocking Parameter to **-C64**.

Note: You should be aware of the capacity and throughput when you select a blocksize. Small blocksizes have a significant impact on performance and a minimal impact on capacity. The capacities of the 2.6 GB format (density) and 6.0 GB format (density) are significantly impacted when you use smaller than the recommended blocksizes. As an example: using a blocksize of 1024 bytes to backup 32 GB of data takes approximately 22 hours. Backing up the same 32 GB of data using a blocksize of 32 K Bytes takes approximately 2 hours.

Device Buffers

The general information for this attribute applies to this tape drive type.

Extended File Marks

The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2

The following chart shows the Supported Data Cartridge type and Density Settings (in decimal and hex) for the IBM 7205-311 Tape Drive. When you perform a Restore (Read) Operation, the tape drive automatically sets the density to match the written density. When you perform a Backup Operation (Write), you must set the Density Setting to match the Data Cartridge that you are using.

Supported Data Cartridges	Native Capacity	Compressed Data Capacity	Web-based System Manager or SMIT Density Setting	HEX Density Setting
DLTtape III	2.6 GB	2.6 GB (No Compression)	23	17h
	6.0 GB	6.0 GB (No Compression)	24	18h
	10.0 GB	20.0 GB (Default for drive)	25	19h
DLTtapeIIIxt	15.0 GB	30.6 GB (Default for drive)	25	19h
DLTtapeIV	20.0 GB	40.0 GB	26	1Ah
	35.0 GB	70.0 GB (Default for drive)	27	1Bh

Note: If you request an unsupported Native Capacity for the Data Cartridge, the drive defaults to the highest supported capacity for the Data Cartridge that is loaded into the drive.

Data Compression

The actual compression depends on the type of data being that is being written. (see above table) A Compression Ratio of 2:1 is assumed for this Compressed Data Capacity.

Attributes with Fixed Values

The general information for this attribute applies to this tape drive type.

Attributes for 150 MB 1/4-Inch Tape Drives (Type 150mb)

Block Size

The default block size is 512. The only other valid block size is **0** for variable length blocks.

Device Buffers

The general information for this attribute applies to this tape drive type.

Extended File Marks

Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you wish to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension

The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2

The following settings apply:

Setting	Meaning
16	QIC-150
15	QIC-120
0	Default (QIC-150), or whatever was the last density setting by a using system.

The default values are 16 for Density Setting #1, and 15 for Density Setting #2.

Attributes with Fixed Values

If a tape drive is configured as a 150 MB 1/4-inch tape drive, attributes for Extended File Marks, Reserve Support, Variable Length Block Size, and Data Compression have predefined values which cannot be changed.

Attributes for 525 MB 1/4-Inch Tape Drives (Type 525mb)

Block Size

The default block size is 512. The other valid block sizes are 0 for variable length blocks, and 1024.

Device Buffers

The general information for this attribute applies to this tape drive type.

Extended File Marks

Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you want to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension

The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2

The following settings apply:

Setting	Meaning
17	QIC-525*
16	QIC-150
15	QIC-120
0	Default (QIC-525), or whatever was the last density setting by a using system.

* QIC-525 is the only mode that supports the 1024 block size.

The default values are 17 for Density Setting #1, and 16 for Density Setting #2.

Attributes with Fixed Values

If a tape drive is configured as a 525 MB 1/4-inch tape drive, the Extended File Marks, Reserve Support, Variable Length Block Size, and Data Compression attributes have predefined values which cannot be changed.

Attributes for 1200 MB 1/4-Inch Tape Drives (Type 1200mb-c)

Block Size

The default block size is 512. The other valid block sizes are 0 for variable length blocks, and 1024.

Device Buffers

The general information for this attribute applies to this tape drive type.

Extended File Marks

Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you wish to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension

The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2

The following settings apply:

Setting	Meaning
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120
0	Default (QIC-1000), or whatever was the last density setting by a using system.

* QIC-525 and QIC-1000 are the only modes that support the 1024 block size.

The default values are 21 for Density Setting #1, and 17 for Density Setting #2.

Attributes with Fixed Values

If a tape drive is configured as a 1200 MB 1/4-inch tape drive, the Extended File Marks, Reserve Support, Variable Length Block Size, and Data Compression attributes have predefined values which cannot be changed.

Attributes for 12000 MB 4 mm Tape Drives (Self Configuring)

Block Size

The IBM 12000 MB 4 mm Tape Drive's throughput is sensitive to blocksize. The minimum recommended blocksize for this drive is 32 K Bytes. Any block size less than 32 K Bytes restricts the data rate (backup/restore time). The following table lists recommended block sizes by command:

Supported Command	Default Block Size (Bytes)	RECOMMENDATION
BACKUP	32 K or 51.2 K (default)	Will use either 32 K or 51.2 K depending on if ″Backup″ is by name or not. No change is required.
TAR	10 K	There is an error in the manual that states a 512 K byte block size. Set the Blocking Parameter to **-N64**.
MKSYSB	See BACKUP	MKSYSB uses the BACKUP Command. No change is required.
DD	n/a	Set the Blocking Parameter to **bs=32K**.
CPIO	n/a	Set the Blocking Parameter to **-C64**.

Note: You should be aware of the capacity and throughput when you select a blocksize. Small blocksizes have a significant impact on performance and a minimal impact on capacity.

Device Buffers

The general information for this attribute applies to this tape drive type.

Extended File Marks

The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2

The following chart shows the Supported Data Cartridge type and Density Settings (in decimal and hex) for the IBM 12000 MB 4 mm Tape Drive. When you perform a Restore (Read) Operation, the tape drive automatically sets the density to match the written density. When you perform a Backup Operation (Write), you must set the Density Setting to match the Data Cartridge you are using.

Supported Data Cartridges	Native Capacity	Compressed Data Capacity	Web-based System Manager or SMIT Density Setting	HEX Density Setting
DDS III	2.0 GB	4.0 GB	19	13h
DDS2	4.0 GB	8.0 GB	36	24h
DDS3	12.0 GB	24.0 GB	37	25h

Note: If you request an unsupported Native Capacity for the Data Cartridge, the drive defaults to the highest supported capacity for the Data Cartridge that is loaded into the drive.

Data Compression

The actual compression depends on the type of data being that is being written. (see above table) A Compression Ratio of 2:1 is assumed for this Compressed Data Capacity.

Attributes with Fixed Values

The general information for this attribute applies to this tape drive type.

Attributes for 13000 MB 1/4-Inch Tape Drives (Self configuring)

Block Size

The default block size is 512. The other valid block sizes are 0 for variable length blocks, and 1024.

Device Buffers
The general information for this attribute applies to this tape drive type.

Extended File Marks
Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you wish to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension
The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2
The following settings apply:

Setting	Meaning
33	QIC-5010-DC*
34	QIC-2GB*
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120
0	Default (QIC-5010-DC)*

* QIC-525, QIC-1000, QIC-5010-DC, and QIC-2GB are the only modes that support the 1024 block size.

The default values are 33 for Density Setting #1, and 34 for Density Setting #2.

Attributes with Fixed Values
If a tape drive is configured as a 13000 MB 1/4-inch tape drive, the Extended File Marks, Reserve Support, and Variable Length Block Size attributes have predefined values which cannot be changed.

Attributes for 1/2-Inch 9-Track Tape Drives (Type 9trk)

Block Size
The default block size is 1024.

Device Buffers
The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2
The following settings apply:

Setting	Meaning
3	6250 bits per inch (bpi)
2	1600 bpi
0	Whichever writing density was used previously.

The default values are 3 for Density Setting #1, and 2 for Density Setting #2.

Attributes with Fixed Values
If a tape drive is configured as a 1/2-inch 9-track tape drive, the Extended File Marks, Retension, Reserve Support, Variable Length Block Size, and Data Compression attributes have predefined values which cannot be changed.

Attributes for 3490e 1/2-Inch Cartridge (Type 3490e)

Block Size

The default block size is 1024. This drive features a high data transfer rate, and block size can be critical to efficient operation. Larger block sizes can greatly improve operational speeds, and in general, the largest possible block size should be used.

Note: Increasing the block value can cause incompatibilities with other programs on your system. If this occurs, you receive the following error message while running those programs:

```
A system call received a parameter that is not valid.
```

Device Buffers

The general information for this attribute applies to this tape drive type.

Compression

The general information for this attribute applies to this tape drive type.

Autoloader

This drive features a tape sequencer, an autoloader that sequentially loads and ejects a series of tape cartridges from the cartridge loader. For this function to operate correctly, the front panel switch should be in the AUTO position and the Autoloader attribute must be set to `yes`.

Attributes for Other SCSI Tapes (Type ost)

Block Size

The system default is 512, but this should be adjusted to the default block size for your tape drive. Typical values are 512 and 1024. 8 mm and 4 mm tape drives usually use 1024 and waste space on the tape if the block size attribute is left at 512. A value of 0 indicates variable block size on some drives.

Device Buffers

The general information for this attribute applies to this tape drive type.

Extended File Marks

The general information for this attribute applies to this tape drive type.

Density Setting #1 and Density Setting #2

The default value is 0 for both of these settings. Other values and their meanings vary for different tape drives.

Reserve Support

The default value is no. This can be set to yes, if the drive supports reserve/release commands. If you are unsure, no is a safer value.

Variable Length Block Size

The default variable length block size value is 0. Nonzero values are used primarily on quarter inch cartridge (QIC) drives. See the SCSI specification for the particular tape drive for advice.

Retry Delay

This attribute applies exclusively to type ost tape drives

Read/Write Timeout

This attribute applies exclusively to type ost tape drives

Attributes with Fixed Values

If a tape drive is configured as an Other SCSI tape drive, the attributes for Extended File Marks, Retension, and Data Compression have predefined values which cannot be changed.

Special Files for Tape Drives

Writing to and reading from files on tapes is done by using **rmt** special files. There are several special files associated with each tape drive known to the operating system. These special files are **/dev/rmt***, **/dev/rmt*.1**, **/dev/rmt*.2**, through **/dev/rmt*.7**. The **rmt*** is the logical name of a tape drive, such as **rmt0**, **rmt1**, and so on.

By selecting one of the special files associated with a tape drive, you make choices about how the I/O operations related to the tape drive will be performed.

Density	You can select whether to write with the tape drive Density Setting #1 or with the tape drive Density Setting #2. The values for these density settings are part of the attributes of the tape drive. Because it is customary to set Density Setting #1 to the highest possible density for the tape drive and Density Setting #2 to the next highest possible density for the tape drive, special files that use Density Setting #1 are sometimes referred to as high density and special files that use Density Setting #2 sometimes are referred to as low density, but this view is not always correct. When reading from a tape, the density setting is ignored.
Rewind-on-Close	You can select whether the tape is rewound when the special file referring to the tape drive is closed. If rewind-on-close is selected, the tape is positioned at the beginning of the tape when the file is closed.
Retension-on-Open	You can select whether the tape is retensioned when the file is opened. Retensioning means winding to the end of the tape and then rewinding to the beginning of the tape to reduce errors. If retension-on-open is selected, the tape is positioned at the beginning of the tape as part of the open process.

The following table shows the names of the rmt special files and their characteristics.

Special File	**Rewind on Close**	**Retension on Open**	**Density Setting**
/dev/rmt*	Yes	No	#1
/dev/rmt*.1	No	No	#1
/dev/rmt*.2	Yes	Yes	#1
/dev/rmt*.3	No	Yes	#1
/dev/rmt*.4	Yes	No	#2
/dev/rmt*.5	No	No	#2
/dev/rmt*.6	Yes	Yes	#2
/dev/rmt*.7	No	Yes	#2

Suppose you want to write three files on the tape in tape drive rmt2. The first file is to be at the beginning of the tape, the second file after the first file, and the third file after the second file. Further, suppose you want Density Setting #1 for the tape drive. The following list of special files, in the order given, could be used for writing the tape.

1. `/dev/rmt2.3`
2. `/dev/rmt2.1`
3. `/dev/rmt2`

These particular special files are chosen because:

- **`/dev/rmt2.3`** is chosen as the first file because this file has Retension-on-Open, which ensures that the first file is at the beginning of the tape. Rewind-on-Close is not chosen because the next I/O operation is to begin where this file ends. If the tape is already at the beginning when the first file is opened, using the **`/dev/rmt2.1`** file as the first file would be faster since time for retensioning the tape is eliminated.

- **`/dev/rmt2.1`** is chosen for the second file because this file has neither Retension-on-Open nor Rewind-on-Close chosen. There is no reason to go to the beginning of the tape either when the file is opened or when it is closed.
- **`/dev/rmt2`** is chosen for the third and final file because Retension-on-Open is not wanted since the third file is to follow the second file. Rewind-on-Close is selected because there are no plans to do any more writing after the third file on the tape. The next use of the tape will begin at the beginning of the tape.

Besides controlling tape operations by choosing a particular rmt special file, you can use the **tctl** command to control tape operations.

Appendix A. Comparisons for BSD System Managers

This appendix is for system administrators who are familiar with 4.3 BSD UNIX or System V operating systems. This information explains the differences and the similarities between those systems and AIX.

Topics discussed in this appendix are:

Comparisions Between AIX and BSD for System Managers

The following articles provide information for 4.3 BSD administrators:

  - “NFS and NIS (formerly Yellow Pages) for BSD 4.3 System Managers” on page 209
  - “Passwords for BSD 4.3 System Managers” on page 209
  - “Performance Measurement and Tuning for BSD 4.3 System Managers” on page 212
  - “Printers for BSD 4.3 System Managers” on page 213
  - “Terminals for BSD 4.3 System Managers” on page 214
  - “UUCP for BSD 4.3 System Managers” on page 215.

Introduction to AIX for BSD System Managers

The following tips can help you get started managing the system:

- Start by logging in as root at the graphics console.
- Perform system management from the system console until you become experienced with the system. It is easier to work from the system console than a remote terminal. Once you are experienced with the system, you can work remotely from an xterm or an ASCII terminal.
- Take advantage of the several AIX facilities for system management tasks. They include:
  - System Management Interface Tool (SMIT). SMIT provides an interface between system managers and configuration and management commands. SMIT can help system managers perform most system administration tasks. For more information, see the “System Management Interface Tool (SMIT) Overview” on page 167.
  - The Object Data Manager (ODM). The ODM provides routines that access objects from the ODM databases. The ODM databases contain device configuration information. For more information about how the ODM databases store device information, see “Chapter 21. Devices” on page 175.
  - The System Resource Controller (SRC). The SRC provides access and control of daemons and other system resources through a single interface. For more information, see the “System Resource Controller Overview” on page 155.

Major Differences between 4.3 BSD and this Operating System

This article summarizes the major differences between this operating system and 4.3 BSD systems. For more detailed discussions of these topics, see the list of articles in “Comparisions Between AIX and BSD for System Managers” on page 195.

Configuration Data Storage

4.3 BSD usually stores configuration data in ASCII files. Related pieces of information are kept on the same line and record processing (sorting and searching) can be done on the ASCII file itself. Records can vary in length and are terminated by a line feed. 4.3 BSD provides tools to convert some potentially large ASCII files to a database (dbm) format. Relevant library functions search the pair of dbm files if they exist, but search the original ASCII file if the dbm files are not found.

Some configuration data for this operating system is stored in ASCII files, but often in a *stanza* format. A stanza is a set of related pieces of information stored in a group of several lines. Each piece of information has a label to make the contents of the file more understandable.

This operating system also supports dbm versions of password and user information. Furthermore, the **/etc/passwd**, **/etc/group**, and **/etc/inittab** files are examples of files for this operating system where the information is stored in traditional form rather than in stanza form.

Other configuration data for this operating system are stored in files maintained by the Object Data Manager (ODM). Web-based System Manager or the System Management Interface Tool (SMIT) can manipulate and display information in ODM files. Alternatively, you can use the ODM commands directly to view these files. To query the ODM files, use the following commands:

- **odmget**
- **odmshow**.

The following ODM commands alter ODM files:
- **odmadd**
- **odmcreate**
- **odmdrop**
- **odmchange**
- **odmdelete**.

> **Attention:** Altering ODM files incorrectly can cause the system to fail, and might prevent you from successfully restarting the system. Only use ODM commands directly on ODM files when task-specific commands, such as those generated by Web-based System Manager or SMIT, are unsuccessful.

Configuration Management

When a system running this operating system starts up, a set of configuration-specific commands are invoked by the Configuration Manager. These configuration-specific commands are called *methods*. Methods identify the devices on the system and update the appropriate ODM files in the **/etc/objrepos** directory.

Device special files in the **/dev** directly are not preinstalled. Some special files, such as those for hard disks, are created automatically during the startup configuration process. Other special files, such as those for ASCII terminals, must be created by the system administrator by using the Web-based System Manager Devices application or the SMIT Devices menu. This information is retained in the ODM for later use by the system.

Disk Management

In this operating system, disk drives are referred to as *physical volumes*. Partitions are referred to as *logical volumes*. As in 4.3 BSD, a single physical volume can have multiple logical volumes. However, unlike 4.3 BSD, a single volume in this operating system can span multiple physical volumes. To do this, you must make several physical volumes into a *volume group* and create logical volumes on the volume group.

Commands in this operating system used for file system and volume management include:
- **crfs**
- **varyonvg**
- **varyoffvg**
- **lsvg**
- **importvg**
- **exportvg**.

The following 4.3 BSD commands are also available:
- **mkfs**
- **fsck**
- **fsdb**
- **mount**
- **umount**.

Differences between these commands for 4.3 BSD and for this operating system of are discussed in “File Systems for BSD 4.3 System Managers” on page 205.

4.3 BSD maintains a list of file systems in the **/etc/fstab** file. This operating system maintains a stanza for each file system in the **/etc/filesystems** file.

New Commands

To handle new configuration and disk management systems, this operating system has about 150 commands that are new to 4.3 BSD administrators. For more information, see “Commands for System Administration for BSD 4.3 System Managers” on page 201.

Startup

This operating system supports automatic identification and configuration of devices. Consequently, the startup process is very different from 4.3 BSD systems. In addition to the kernel, an image of a boot file system and the previous base device configuration information is loaded to a RAM disk. In the first phase of startup, sufficient configuration information is loaded and checked to permit accessing logical volumes. The paging space device is identified to the kernel and the hard disk root file system is checked. At this time, the operating system changes the root file system from the RAM disk to the hard disk and completes the startup procedure, including configuring other devices.

User Authorization

4.3 BSD, and versions of AT&T UNIX operating systems before SVR4, store all user authentication information, including encrypted passwords, in the **/etc/passwd** file. Traditionally, the **/etc/passwd** file could be read by all.

On SVR4 systems, encrypted passwords are removed from the **/etc/passwd** file and stored in the **/etc/shadow** file. Only users with root authority and trusted programs (such as the **/bin/login** program) can read the **/etc/shadow** file.

This operating system stores encrypted passwords in the **/etc/security/passwd** file. Other files in the **/etc/security** directory are the **user** and **limits** files. These three files define the way a user is allowed to access the system (such as using the **rlogin** or **telnet** commands) and the user’s resource limits (such as file size and address space).

Printing

Most 4.3 BSD printing commands are supported with minor differences. One difference is that the **/etc/qconfig** file is the configuration file in this operating system.

The line printing system for this operating system can interoperate with the 4.3 BSD line printing system, both for submitting print jobs to 4.3 BSD systems and for printing jobs submitted from a 4.3 BSD system.

Shells

This operating system supports the Bourne shell, C shell and Korn shell. The full path name for the Bourne shell program is **/bin/bsh**. The **/bin/sh** file is a hard link to the **/bin/ksh** file. This file can be changed by the administrator.

AIX does not support **setuid** or **setgid** for shell scripts in any shell.

Notes:

1. This operating system has no shell scripts that rely on the **/bin/sh**. However, many shell scripts from other systems rely on **/bin/sh** being the Bourne shell.
2. Although the Bourne shell and Korn shell are similar, the Korn shell is not a perfect superset of the Bourne shell.

Accounting for BSD 4.3 System Managers

The accounting files in the **/usr/lib/acct** directory and the system activity reporting tools in the **/usr/lib/sa** directory for this operating system are identical to those available with AT&T System V Release 4 (SVR4) with the addition of 4.3 BSD accounting utilities.

Many of the accounting commands are in the **/usr/lib/acct** directory. To begin system accounting, use the**/usr/lib/acct/startup** command. If accounting is not started, commands such as **lastcomm**(1) cannot return information.

This operating system provides these 4.3 BSD accounting facilities:

last(1)	Indicates last logins of users and terminals
lastcomm (1)	Shows in reverse order the last commands executed
acct(3)	Enables and disables process accounting
ac(8)	Login accounting
accton (8)	Turns system accounting on or off
sa(8)	Generally maintains system accounting files.

This operating system also provides these System V Interface Definition (SVID) Issue II accounting commands and library functions:

acctcms (1)	Produces command usage summaries from accounting records
acctcom (1)	Displays selected process-accounting record summaries
acctcon1 (1)	Converts login/logoff records to session records
acctcon2 (1)	Converts login/logoff records to total accounting records
acctdisk (1)	Generates total accounting records from **diskusg**(1) command output
acctmerg (1)	Merges total accounting files into an intermediary file
accton (1)	Turns on accounting
acctprc1 (1)	Processes accounting information from **acct**(3) command
acctprc2 (1)	Processes output of **acctprc1**(1) command into total accounting records
acctwtmp (1)	Manipulates connect-time accounting records
chargefee(1)	Charges to login name
ckpacct(1)	Checks size of **/usr/adm/pacct** file
diskusg (1)	Generates disk accounting information
dodisk (1)	Performs disk accounting
fwtmp(1)	Converts binary records (**wtmp** file) to formatted ASCII. **Note:** The **wtmp** file is in the **/var/adm** directory
lastlogin(1)	Updates last date on which each person logged in
monacct (1)	Creates monthly summary files
prctmp (1)	Prints session record file produced by **acctcon1**(1) command
prdaily (1)	Formats a report of yesterday's accounting information
prtacct (1)	Formats and prints any total accounting file
runacct (1)	Runs daily accounting
shutacct (1)	Called by system shutdown to stop accounting and log the reason
startup (1)	Called by system initialization to start accounting
turnacct (1)	Turns process accounting on or off
wtmpfix(1)	Corrects time/date stamps in a file using **wtmp** format.

Backup for BSD 4.3 System Managers

The **tar** and **cpio** commands can move data between systems. The **tar** command for this operating system is not fully compatible with the 4.3 BSD **tar** command. The **tar** command for this operating system requires the **-B** flag (blocking input) if it is reading from a pipe. The AT&T **cpio** command is compatible with this version.

This operating system can read and write in **dump** and **restore** command format. For example, the **backup** command for this operating system with the syntax:

```
backup -0uf Device Filesystemname
```

is the same as the 4.3 BSD **dump** command with the syntax:

```
dump 0uf Device Filesystemname
```

Similarly, the **restore** command for this operating sytem with the syntax:

```
restore -mivf Device
```

is the same as the 4.3 BSD **restore** command with the syntax:

```
restore ivf Device
```

This operating system also has the 4.3 BSD **rdump** and **rrestore** commands. The only difference in the two versions is that for this operating system each argument must be preceded by a **-** (dash). For example, the following command:

```
rdump -0 -f orca:/dev/rmt0 /dev/hd2
```

is equivalent to the 4.3 BSD command:

```
rdump 0f orca:/dev/rmt0 /dev/hd2
```

The **backup** command for this operating system with the following syntax:

```
backup -0f /dev/rmt0 /dev/hd2
```

is equivalent to the 4.3 BSD **dump** command with this syntax:

```
dump 0f /dev/rmt0 /dev/hd2
```

Non-IBM SCSI Tape Support

This operating system does not directly support non-IBM SCSI tape drives. However, you can add your own header and interface that use the IBM SCSI driver. For more information, see the information on adding an unsupported device to the system in *AIX 5L Version 5.1 Kernel Extensions and Device Support Programming Concepts* and “Backup Overview” on page 99.

Startup for BSD 4.3 System Managers

On 4.3 BSD systems, the **init** program is the last step in the startup procedure. The main role of the **init** program is to create processes for each available terminal port. The available terminal ports are found by reading the **/etc/ttys** file.

On System V, the **init** program is started at system initialization. The **init** process starts processes according to entries in the **/etc/inittab** file.

This operating system follows the System V initialization procedure. You can edit the **/etc/inittab** file by directly editing the file, using the **telinit** command, or by using the following commands:

chitab(1)	Changes records in the **/etc/inittab** file

lsitab(1)	Lists records in the **/etc/inittab** file
mkitab(1)	Makes records in the **/etc/inittab** file
rmitab(1)	Removes records in the **/etc/inittab** file.

Changes made to the **/etc/inittab** file take effect the next time the system is rebooted, or when the **telinit q** command is run.

Commands for System Administration for BSD 4.3 System Managers

This list contains commands that are specifically for administering the environment for this operating system.

bosboot (1)	Initializes a boot device.
bootlist (1)	Alters the list of boot devices (or the ordering of these devices in the list) available to the system.
cfgmgr(1)	Configures devices by running the programs in **/etc/methods** directory.
chcons (1)	Redirects the system console to device or file, effective next startup
chdev(1)	Changes the characteristics of a device
chdisp (1)	Changes the display used by the low-function terminal (LFT) subsystem.
checkcw(1)	Prepares constant-width text for the **troff** command.
checkeq(1)	Checks documents formatted with memorandum macros.
checkmm(1)	Checks documents formatted with memorandum macros.
checknr (1)	Checks **nroff** and **troff** files.
chfont(1)	Changes the default font selected at boot time.
chfs(1)	Changes attributes of a file system.
chgroup (1)	Changes attributes for groups.
chgrpmem (1)	Changes the administrators or members of a group.
chhwkbd (1)	Changes the low function terminal (LFT) keyboard attributes stored in the Object Data Manager (ODM) database.
chitab(1)	Changes records in the **/etc/inittab** file.
chkbd(1)	Changes the default keyboard map used by the low-function terminal (LFT) at system startup.
chkey(1)	Changes your encryption key.
chlang	Sets **LANG** environment variable in the **/etc/environment** file for the next login.
chlicense (1)	There are two types of user licensing, fixed and floating. Fixed licensing is always enabled, and the number of licenses can be changed through the **-u** flag. Floating licensing can be enabled or disabled (on or off) through the **-f** flag
chlv(1)	Changes the characteristics of a logical volume
chnamsv (1)	Changes TCP/IP-based name service configuration on a host
chprtsv (1)	Changes a print service configuration on a client or server machine
chps(1)	Changes attributes of a paging space.
chpv(1)	Changes the characteristics of a physical volume in a volume group.
chque(1)	Changes the queue name.
chquedev (1)	Changes the printer or plotter queue device names.
chssys (1)	Changes a subsystem definition in the subsystem object class.
chtcb(1)	Changes or queries the trusted computing base attribute of a file.
chtz	Changes the system time zone information.
chuser(1)	Changes attributes for the specified user.
chvfs(1)	Changes entries in the **/etc/vfs** file.
chvg(1)	Sets the characteristics of a volume group.
chvirprt (1)	Changes the attribute values of a virtual printer.
crfs (1)	Adds a file system.
crvfs(1)	Creates entries in the **/etc/vfs** file.
exportvg (1)	Exports the definition of a volume group from a set of physical volumes.
extendvg (1)	Adds physical volumes to a volume group.

grpck(1)	Verifies the correctness of a group definition.
importvg(1)	Imports a new volume group definition from a set of physical volumes.
lsallq (1)	Lists the names of all configured queues.
lsallqdev (1)	Lists all configured printer and plotter queue device names within a specified queue.
lsdisp(1)	Lists the displays currently available on the system.
lsfont(1)	Lists the fonts available for use by the display.
lsfs(1)	Displays the characteristics of file systems.
lsgroup(1)	Displays the attributes of groups.
lsitab(1)	Lists the records in the **/etc/inittab** file.
lskbd(1)	Lists the keyboard maps currently available to the low-function terminal (LFT) subsystem.
lslicense (1)	Displays the number of fixed licenses and the status of floating licensing.
lslpp (1)	Lists optional program products.
lsnamsv (1)	Shows name service information stored in the database.
lsprtsv (1)	Shows print service information stored in the database.
lsps	Lists paging space and attributes.
lsque(1)	Displays the queue stanza name.
lsquedev (1)	Displays the device stanza name.
lssrc(1)	Gets the status of a subsystem, a group of subsystems, or a subserver.
lsuser(1)	Displays attributes of user accounts.
lsvfs(1)	Lists entries in the **/etc/vfs** file.
mkcatdefs (1)	Preprocesses a message source file.
runcat(1)	Pipes the output data from the **mkcatdefs** command to the **gencat** command.
mkdev(1)	Adds a device to the system.
mkfont (1)	Adds the font code associated with a display to the system.
mkfontdir (1)	Creates a **fonts.dir** file from a directory of font files.
mkgroup (1)	Creates a new group.
mkitab(1)	Makes records in the **/etc/inittab** file.
mklv(1)	Creates a logical volume.
mklvcopy (1)	Adds copies to a logical volume.
mknamsv (1)	Configures TCP/IP-based name service on a host for a client.
mknotify (1)	Adds a notify method definition to the notify object class.
mkprtsv (1)	Configures TCP/IP-based print service on a host.
mkps(1)	Add an additional paging space to the system.
mkque(1)	Adds a printer queue to the system.
mkquedev (1)	Adds a printer queue device to the system.
mkserver (1)	Adds a subserver definition to the subserver object class.
mkssys(1)	Adds a subsystem definition to the subsystem object class.
mksysb	Backs up mounted file systems in the rootvg volume group for subsequent reinstallation.
mkszfile	Records size of mounted file systems in the rootvg volume group for reinstallation.
mktcpip (1)	Sets the required values for starting TCP/IP on a host.
mkuser (1)	Creates a new user account.
mkuser.sys (1)	Customizes a new user account.
mkvg(1)	Creates a volume group.
mkvirprt (1)	Makes a virtual printer.
odmadd(1)	Adds objects to created object classes.
odmchange (1)	Changes the contents of a selected object in the specified object class.
odmcreate (1)	Produces the **.c** (source) and **.h** (include) files necessary for ODM application development and creates empty object classes.
odmdelete (1)	Deletes selected objects from a specified object class.
odmdrop (1)	Removes an object class.
odmget(1)	Retrieves objects from the specified object classes and places them into an **odmadd** input file.
odmshow (1)	Displays an object class definition on the screen.
pwdck(1)	Verifies the correctness of local authentication information.

redefinevg	Redefines the set of physical volumes of the given volume group in the device configuration database.
reducevg (1)	Removes physical volumes from a volume group. When all physical volumes are removed from the volume group, the volume group is deleted.
reorgvg (1)	Reorganizes the physical partition allocation for a volume group.
restbase (1)	Restores customized information from the boot image.
rmdel(1)	Removes a delta from a Source Code Control System (SCCS) file.
rmdev(1)	Removes a device from the system.
rmf(1)	Removes folders and the messages they contain.
rmfs(1)	Removes a file system.
rmgroup(1)	Removes a group.
rmitab(1)	Removes records in the **/etc/inittab** file.
rmlv(1)	Removes logical volumes from a volume group.
rmlvcopy (1)	Removes copies from a logical volume.
rmm(1)	Removes messages.
rmnamsv (1)	Unconfigures TCP/IP-based name service on a host.
rmnotify (1)	Removes a notify method definition from the notify object class.
rmprtsv (1)	Unconfigures a print service on a client or server machine.
rmps(1)	Removes a paging space from the system.
rmque(1)	Removes a printer queue from the system.
rmquedev (1)	Removes a printer or plotter queue device from the system.
rmserver (1)	Removes a subserver definition from the subserver object class.
rmssys(1)	Removes a subsystem definition from the subsystem object class.
rmuser(1)	Removes a user account.
rmvfs(1)	Removes entries in the **/etc/vfs** file.
rmvirprt (1)	Removes a virtual printer.
savebase (1)	Saves base customized device data in the ODM onto the boot device.
swapoff(1)	Deactives one or more paging space.
swapon(1)	Specifies additional devices for paging and swapping.
syncvg (1)	Synchronizes logical volume copies that are not current.
usrck(1)	Verifies the correctness of a user definition.
varyoffvg(1)	Deactivates a volume group.
varyonvg (1)	Activates a volume group.

Cron for BSD 4.3 System Managers

The **cron** daemon for this operating system is similar to the System V Release 2 **cron** daemon. An entry in the **/etc/inittab** file starts the **cron** daemon.

Devices for BSD 4.3 System Managers

A device on a 4.3 BSD system is accessible to an application only when:
- The device is physically installed and functioning.
- The driver for the device is in the kernel.
- The device special files for the device exist in the **/dev** directory.

A device on this operating system is accessible to an application only when:
- The device is physically installed and functioning.
- The driver for the device is in the kernel or in a loaded kernel extension.
- The device special files for the device exist in the **/dev** directory.
- The object database in the **/etc/objrepos** directory contains entries for the device that match the physical configuration.

The device specific programs called *methods*, found in the **/etc/methods** directory, maintain the object database. The methods are invoked by the Configuration Manager (accessed through the **cfgmgr** command) and other commands.

If a device can no longer be accessed by an application program, it can mean that the hardware is faulty or it can mean that the configuration database in the **/etc/objrepos** directory is damaged.

The **cfgmgr** command processes the configuration database in the **/etc/objrepos** directory and is processed at startup time by the **cfgmgr** command (the Configuration Manager).

The pseudocode below shows the Configuration Manager logic:

```
/* Main */
While there are rules in the Config_Rules database
        {
        Get the next rule and execute it
        Capture stdout from the last execution
        Parse_Output(stdout)
        }
/* Parse Output Routine */
/* stdout will contain a list of devices found */
Parse_OutPut(stdout)
        {
        While there are devices left in the list
                {
                Lookup the device in the database
                if (!defined)
                        Get define method from database and execute
                if (! configured)
                        {
                        Get config method from database and execute
                        Parse_Output(stdout)
                         }
                }
        }
```

File Comparison Table for 4.3 BSD, SVR4, and this Operating System

The following table compares file names and functions between 4.3 BSD, SVR4, and this operating system.

File Comparison Table				
4.3 BSD File	**SVR4 File**	**File for this operating system**	**Database**	**Type (odm/dbm)**
L-Devices	Devices	Devices	no	
L-dialcodes	Dialcodes	Dialcodes	no	
L.cmds	Permissions	Permissions	no	
L.sys	Systems	Systems	no	
USERFILE	Permissions	Permissions	no	
aliases	mail/namefiles	aliases	aliasesDB/DB	dbm
fstab	vfstab	filesystems	no	
ftpusers	ftpusers	ftpusers	no	
gettytab		N/A		
group	group	group	no	
hosts	hosts	hosts	no	

hosts.equiv	hosts.equiv	hosts.equiv	no	
inetd.conf	inetd.conf	inetd.conf	no	
map3270	N/A	map3270	no	
motd	motd	motd	no	
mtab	mnttab	N/A	no	
named.boot	named.boot	named.boot	no	
named.ca		named.ca	no	
named.hosts		named.data (See note)	no	
named.local		named.local	no	
named.pid	named.pid	named.pid	no	
named.rev		named.rev	no	
networks	networks	networks	no	
passwd	passwd	passwd	no	
printcap	qconfig	qconfig		
protocols		protocols	no	
remote	remote	remote	no	
resolv.conf	resolv.conf	resolv.conf	no	
sendmail.cf	sendmail.cf	sendmail.cf	sendmail.cfDB	neither
services		services	no	
shells	shells	N/A		
stab		N/A		
syslog.conf		syslog.conf	no	
syslog.pid		syslog.pid	no	
termcap	terminfo	terminfo		
ttys	ttys	N/A	yes	odm
types		N/A		
utmp	utmp	utmp		
vfont		N/A		
vgrindefs		vgrindefs		
wtmp	wtmp	wtmp		

Note: The file names **named.ca**, **named.hosts**, **named.local**, and **named.rev** are user definable in the **named.boot** file. However, these are the names used for these files in the documentation for this operating system.

File Systems for BSD 4.3 System Managers

This information offers a brief comparison of file systems for this operating system to other system file systems, and an outline of the supported file system types on this operating system.

This operating system uses the **/etc/filesystem** file to list file system device information, and has similar commands for mounting and unmounting file systems.

/etc/filesystems File and /etc/fstab File

4.3 BSD systems store lists of block devices and mount points in the **/etc/fstab** file.

SVR4 systems stores block devices and mount point information in **/etc/vfstab** file.

This operating system stores block device and mount points information in **/etc/filesystems** file. The **crfs**, **chfs**, and **rmfs** commands update the **/etc/filesystems** file.

4.3 BSD system administrators might be interested in the *check* variable in the **/etc/filesystems** file. The *check* variable can be set to the value `True`, `False` or to a number. For example, you can specify `check=2` in the **/etc/filesystems** file. The number specifies the pass of the **fsck** command that will check this file system. The **check** parameter corresponds to the fifth field in an **/etc/fstab** file record.

There is no dump frequency parameter in the **/etc/filesystems** file.

File System Support on this operating system

This operating system supports disk quotas.

This operating system does not allow mounting of diskettes as file systems.

The syntax of the**mount** and **umount** commands for this operating system differs from 4.3 BSD and from SVR4 versions of these commands. The commands to mount and unmount all file systems at once are shown for all three systems in the following table:

mount and unmount Commands			
Function	**Syntax for this operating system**	**4.3 BSD Syntax**	**SVR4 Syntax**
mount all file systems	**mount all**	**mount -a**	**mountall**
unmount all file systems	**umount all**	**umount -a**	**umountall**

See the “File Systems Overview” on page 67 for more information.

Finding and Examining Files for BSD 4.3 System Managers

This operating system supports the following 4.3 BSD file commands:

- **which**
- **whereis**
- **what**
- **file**.

This operating system does not support the 4.3 BSD **fast find** syntax of the **find** command. At this time, there is no replacement function. The following **ffind** shell script can be used to simulate the functionality:

```
#!/bin/bsh
PATH=/bin
for dir in /bin /etc /lib /usr
do
find $dir -print | egrep $1
done
```

The syntax for the **ffind** script is:

```
ffind Filename
```

Paging Space for BSD 4.3 System Managers

The following commands assist in managing paging space (also known as swap space):

chps(1)	Changes attributes of a paging space
lsps(1)	List attributes of a paging space
mkps(1)	Add an additional paging space to the system
rmps(1)	Removes a paging space from the system
swapoff(1)	Deactives one or more paging space
swapon(1)	Specifies additional devices for paging and swapping.

If a large paging space is required, place one paging logical volume for each hard disk. This allows scheduling of paging across multiple disk drives.

Networking for BSD 4.3 System Managers

This article describes how to use 4.3 BSD ASCII network configurations, additional commands and command options, and name and address resolution for this operating system, as well as the differences between 4.3 BSD network management and network management for this operating system.

How to Change Default Startup to Permit 4.3 BSD ASCII Configuration

You can administer network interfaces for this operating system through the SMIT and ODM files, or through 4.3 BSD ASCII configuration files.

To administer network interfaces through 4.3 BSD ASCII configuration files, uncomment the commands in the **/etc/rc.net** file below the heading:

```
# Part II - Traditional
Configuration
```

Then if you want flat file configuration and SRC support, edit the **/etc/rc.net** file and uncomment the **hostname**, **ifconfig**, and **route** commands with the appropriate parameters.

If you want flat file configuration without SRC support, use the **smit setbootup_option** fast path to change the system to BSD-style **rc** configuration. This option configures the system to use the **/etc/rc.bsdnet** file at startup. You also have to edit the **/etc/rc.bsdnet** file and uncomment the **hostname**, **ifconfig**, and **route** commands with the appropriate parameters.

Additional Options for ifconfig and netstat Commands

The **ifconfig** command for this operaitng system has the following additional options:

mtu	The *mtu* variable specifies the maximum transmission unit (MTU) used on the local network (and local subnets) and the MTU used for remote networks. To maximize compatibility with Ethernet and other networks, set both the Token-Ring and Ethernet default *mtu* value to 1500.
allcast	The **allcast** flag sets the Token-Ring broadcast strategy. Setting the **allcast** flag optimizes connectivity through Token-Ring bridges. Clearing the **allcast** flag (by specifying **-allcast**) minimizes excess traffic on the ring.

The **netstat** command for this operating system has the **-v** flag. The **netstat -v** command prints driver statistics such as transmit byte count, transmit error count, receive byte count, and receive error count.

Additional Network Management Commands

The following additional commands are supported on this operating system:

securetcpip	The **securetcpip** shell script enables controlled access mode, which provides enhanced network security. It disallows the running of several unsecured TCP/IP programs, such as the **tftp**, **rcp**, **rlogin**, and **rsh** programs. It also restricts the use of the **.netrc** file
gated	The **gated** command provides MIB support for SNMP
no	The **no** command sets network options that include:

dogticks	Sets timer granularity for **ifwatchdog** routines
subnetsarelocal	Determines if packet address is on the local network
ipsendredirects	Specifies whether the kernel should send redirect signals
ipforwarding	Specifies whether the kernel should forward packets
tcp_ttl	Specifies the time-to-live for Transmission Control Protocol (TCP) packets
udp_ttl	Specifies the time-to-live for User Datagram Protocol (UDP) packets
maxttl	Specifies the time-to-live for Routing Information Protocol (RIP) packets
ipfragttl	Specifies the time-to-live for Internet Protocol (IP) fragments
lowclust	Specifies a low water mark for cluster **mbuf** pool
lowmbuf	Specifies a low water mark for the **mbuf** pool
thewall	Specifies the maximum amount of memory that is allocated to the **mbuf** and cluster **mbuf** pool
arpt_killc	Specifies the time in minutes before an inactive complete Address Resolution Protocol (ARP) entry is deleted

iptrace	The **iptrace** command provides interface-level packet tracing for Internet protocols.
ipreport	The **ipreport** command formats the trace into human-readable form. An example of using this command is the following:

```
iptrace -i en0 /tmp/iptrace.log
# kill iptrace daemon
kill ′ps ax | grep iptrace | awk '{ print $1 }'′
ipreport /tmp/iptrace.log | more
```

Name and Address Resolution

The **gethostbyname** and **gethostbyaddr** subroutines in the **libc** library provide support for Domain Name Service, Network Information Services (NIS, formerly called Yellow Pages), and the **/etc/hosts** database. If the **/etc/resolv.conf** file exists, the name server is always checked first. If the name is not resolved and NIS is running, NIS is checked. If NIS is not running, the **/etc/hosts** file is checked.

Differences between this operating system and 4.3 BSD

On this operating system, the network daemons are started from the **/etc/rc.tcpip** file, not the**/etc/rc.local** file. The **/etc/rc.tcpip** shell script is invoked from the **/etc/inittab** file, not the **/etc/rc** file.

If the System Resource Controller (SRC) is running, the TCP/IP daemons run under SRC control. If you do not want the TCP/IP daemons running under SRC control, use the **smit setbootup_option** fast path to change the system to BSD-style **rc** configuration.

These network management functions available on 4.3 BSD are supported by this operating system:

- Kernel-level SYSLOG logging facilities
- Xerox Network Systems (XNS) support
- Access rights for UNIX domain sockets.

The tn3270 Command

The **tn3270** command is a link to the **telnet** command, but it uses the **/etc/map3270** file and the current **TERM** environment variable value to provide 3270 keyboard mappings. Thus, the **tn3270** command operates exactly like the BSD version.

If you want to change the escape sequences from the defaults used by the **tn3270**, **telnet**, or **tn** commands, set the **TNESC** environment variable before starting these commands.

Online Documentation and man Command for BSD 4.3 System Managers

This operating system supports the **man-k**, **apropos**, and **whatis** commands, but the database used by these commands must first be created with the **catman-w** command.

The **man** command first searches for flat text pages in the **/usr/man/cat?** files. Next, it searches **nroff**-formatted pages in **/usr/man/man?** files. New man pages can be added in flat text or **nroff** form.

Notes:

1. The **man** command text pages are not provided with the system. The **catman** command creates the database from these text pages. These pages can be either flat text pages stored in the **/usr/man/cat**? files or **nroff** formatted pages stored the in **/usr/man/man**? files.
2. The Text Formatting licensed program must be installed for the **nroff** command to be available for the **man** command to read **nroff**-formatted man pages.

NFS and NIS (formerly Yellow Pages) for BSD 4.3 System Managers

Network File System (NFS) and Network Information Services (NIS) daemons are started from the **/etc/rc.nfs** file. However, before the NFS and NIS daemons can be started, the **portmap** daemon must be started in the **/etc/rc.tcpip** file. By default, the **/etc/rc.nfs** file is not invoked by the **/etc/inittab** file. If you add a line in the **/etc/inittab** file to invoke the **/etc/rc.nfs** script, it should be invoked after the **/etc/rc.tcpip** script.

If NIS is active, include a root entry prior to the **+**:: (plus sign, colon, colon) entry in the **/etc/passwd** file and a system entry prior to the **+**:: entry in the **/etc/group** file. This allows a system administrator to log in as root and make changes if the system is unable to communicate with the NIS server.

NFS can be configured by using Web-based System Manager (type `wsm`, then select `Network`), or the SMIT fast path, **smit nfs**. The Web-based System Manager and SMIT menus refer to NIS (formerly Yellow Pages) as NIS. Many of the NFS and NIS commands are found in the **/etc** and **/usr/etc** directory.

Some NFS environments use an **arch** command to identify machine families and types of machines. For example if you are using the IBM RS/6000, specify the **power** identifier for family (CPU), and the **ibm6000** identifier for type (machine).

Passwords for BSD 4.3 System Managers

The following information details the differences between managing passwords in this operating system and 4.3 BSD systems.

Setting a User Password

When you use the **/bin/passwd** command for this operating system as the root user, you are prompted for the current root user password. An example of using the **/bin/passwd** command follows:

```
# passwd cslater
Changing password for "cslater"
Enter root's Password or
cslater's Old password:
cslater's New password:
Re-enter cslater's
new password:
#
```

The 4.3 BSD version does not prompt for the current root user password. An example of the 4.3 BSD version follows:

```
# passwd cslater
New password:
Retype new password:
#
```

Importing a 4.3 BSD Password File

You can import a 4.3 BSD password file by first copying it to the **/etc/passwd** file and entering:

```
pwdck -y ALL
```

Then the **/etc/security/limits** file must be updated with a null stanza for any new users. The **usrck** command does this, but using the **usrck** command can cause problems unless the **/etc/group** file is imported with the **/etc/passwd** file.

> **Note:** If the **/etc/security/limits** file is modified, the stack must not exceed 65,536 bytes. If it does, running the **usrck** command can cause problems. Change the stack size to 65,536 and run **usrck** command again.

Also run the **grpck** and **usrck** command to verify group and user attributes.

Editing the Password File

In this operating system, the **lsuser**, **mkuser**, **chuser**, and **rmuser** commands are provided for managing passwords. All of these commands can be used by running Web-based System Manager or SMIT. However, all of these commands deal with only one user at a time.

> **Note:** Using an editor to change several user name entries at one time requires editing of several files simultaneously, because passwords are stored in **/etc/security/passwd** file, authorization information is stored in the **/etc/security/user** file, and the remaining user data is stored in the **/etc/passwd** file.

This operating system does not support the **vipw** command but does support the **mkpasswd** command. However, you can still administer passwords on this operating system in a 4.3 BSD manner. Use the following procedure:

1. Put a 4.3 BSD password file in the **/etc/shadow** file.
2. Change the permissions to the file by entering:
   ```
   chmod 000 /etc/shadow
   ```
3. Place the following **vipw** shell script in the **/etc** directory:
   ```
   -----------------------------------------------------
   ----
   #!/bin/bsh
   #
   # vipw. Uses pwdck for now. May use usrck someday
   ```

```
#
PATH=/bin:/usr/bin:/etc:/usr/ucb # Add to this if your editor is
                                 # some place else
if [ -f /etc/ptmp ] ; then
                 echo "/etc/ptmp exists. Is someone else using vipw?"
        exit 1
fi
if [ ! -f /'which "$EDITOR" | awk '{ print $1 }'' ] ; then
        EDITOR=vi
fi
cp /etc/shadow /etc/ptmp
if (cmp /etc/shadow /etc/ptmp) ; then
        $EDITOR /etc/ptmp
else
        echo cannot copy shadow to ptmp
        exit 1
fi
if (egrep "^root:" /etc/ptmp >/dev/null) ; then
        cp /etc/ptmp /etc/shadow ; cp /etc/ptmp /etc/passwd
        chmod 000 /etc/passwd /etc/shadow
        pwdck -y ALL 2>1 >/dev/null # return code 114 may change
                rc=$?
        if [ $rc -eq 114 ]; then
                chmod 644 /etc/passwd
                        rm -f /etc/passwd.dir /etc/passwd.pag
                mkpasswd /etc/passwd
                        # update /etc/security/limits, or ftp
                        # will fail
                else
                         pwdck -y ALL
                fi
else
        echo bad entry for root in ptmp
fi
rm /etc/ptmp
-----------------------------------------------------------
```

4. If you use the **vipw** shell script or the **mkpasswd** command, be aware that Web-based System Manager, SMIT, and the **mkuser**, **chuser**, and **rmuser** commands, do not use the **mkpasswd** command. You must run:

   ```
   mkpasswd /etc/passwd
   ```

   to update the **/etc/passwd.dir** and **/etc/passwd.pag** files.

   Attention: Initialization of the *IFS* variable and the **trap** statements guard against some of the common methods used to exploit security holes inherent in the **setuid** feature. However, the **vipw** and **passwd** shell scripts are intended for relatively open environments where compatibility is an important consideration. If you want a more secure environment, use only the standard commands fir this operating system.
5. Put the following passwd shell script in the **/usr/ucb** directory:

```
-----------------------------------------------------
#!/bin/ksh
#
# matches changes to /etc/security/passwd file with changes to
#/etc/shadow
#
IFS=" "
PATH=/bin
trap "exit 2" 1 2 3 4 5 6 7 8 10 12 13 14 15 16 17 18 21 22 \
        23 24 25 27 28 29 30 31 32 33 34 35 36 60 61 62
if [ -n "$1" ]; then
        USERNAME=$1
else
        USERNAME=$LOGNAME
```

```
fi
if [ -f /etc/ptmp ]; then
        echo password file busy
        exit 1
fi
        trap "rm /etc/ptmp; exit 3" 1 2 3 4 5 6 7 8 10 12 13 \
                14 15 16 17 18 21 22 23 24 25 27 28 29 30 31 \
                32 33 34 35 36 60 61 62
if (cp /etc/security/passwd /etc/ptmp) ; then
        chmod 000 /etc/ptmp else
        rm -f /etc/ptmp exit 1
fi
if ( /bin/passwd $USERNAME ) ; then
        PW=′ awk ' BEGIN { RS = "" }
                $1 == user { print $4 } ' user="$USERNAME:" \
/etc/security/passwd ′
else
        rm -f /etc/ptmp
        exit 1
fi
rm -f /etc/ptmp
awk -F: '$1 == user { print $1":"pw":"$3 ":"$4":"$5":"$6":"$7 }
        $1 != user { print $0 }' user="$USERNAME" pw="$PW" \
                /etc/shadow > /etc/ptmp
chmod 000 /etc/ptmp
mv -f /etc/ptmp /etc/shadow
---------------------------------------------------------
```

6. Change the permissions to the **passwd** script by entering:

```
chmod 4711 /usr/ucb/passwd
```

7. Ensure that each user **PATH** environmental variable specifies that the **/usr/ucb** directory be searched before the **/bin** directory.

Performance Measurement and Tuning for BSD 4.3 System Managers

All devices on this operating system have attributes associated with them. To view device attributes, enter:

```
lsattr -E -l devicename
```

Any attributes with the value True can be modified with the command:

```
chdev -l devicename -a attr=value
```

Attention: Changing device parameters incorrectly can damage your system.

By default, the maximum number of processes per user is 40. The default value might be too low for users who have many windows open simultaneously. The following command can be used to change the value systemwide:

```
hdev -l sys0 -a maxuproc=100
```

This example changes the maximum number to 100. The new value is set once the system has restarted.

To view the current setting of this and other system attributes type:

```
lsattr -E -l sys0
```

The **maxmbuf** attribute is not currently supported by the mbuf services.

This operating system supports the **vmstat** and **iostat** commands, but not the **systat** command or load averages.

Printers for BSD 4.3 System Managers

In AIX 5.1 and later, the operating system supports two printing subsystems: 4.3 BSD and System V. The System V style of printing subsystem uses System V Release 4 commands, queues, and files and is administered the same way. The following paragraphs describe what you need to know to manage the 4.3 BSD style of printing subsystem. You control which subsystem is made active through SMIT. Only one subsystem can be active at a time.

Printing is managed by programs and configurations in the **/usr/lpd** directory. The design, configuration, queueing mechanism, and daemon processes of the 4.3 BSD and printer subsystems for this operating system are different. However, they both use the **lpd** protocol for remote printing. Both systems use **/etc/hosts.lpd**, if it exists, or **/etc/host.equiv** otherwise. The printer subsystem for this operating system offers a gateway to 4.3 BSD printer subsystems, so systems using this operating system can submit print jobs to 4.3 BSD systems and accept print jobs submitted by 4.3 BSD systems.

The **/etc/printcap** file of 4.3 BSD does not exist in this operating system. This file is a combination of spooler configuration and printer capability data base. Users need to understand the format and keywords of the **printcap** file to set up a printer correctly.

The **/etc/qconfig** file of this operating system contains only the spooler configuration information. The printer capability is defined in the ODM predefined/customized data base. You can use the **mkvirprt** (make virtual printer) command to define to the system the capabilities of a particular printer.

To make printer lp0 available to print on the remote host viking, put the following in a 4.3 BSD system **/etc/printcap** file:

```
lp0|Print on remote printer attached to
viking:Z
:lp=:rm=viking:rp=lp:st=/usr/spool/lp0d
```

To do the same on this operating system, put the following in the **/etc/qconfig** file:

```
lp0:
        device = dlp0
        host = viking
        rq = lp
dlp0:
        backend = /usr/lib/lpd/rembak
```

For more information about the printer subsystem, see the Printer Overview for System Management.

This operating system supports the following printer commands and library functions:

cancel (1)	Cancels requests to a line printer
chquedev (1)	Changes the printer or plotter queue device names
chvirprt (1)	Changes the attribute values of a virtual printer
disable (1)	Disables a printer queue
enable(1)	Enables a printer queue
hplj(1)	Postprocesses **troff** output for HP LaserJetII with the K cartridge
ibm3812 (1)	Postprocesses **troff** output for IBM 3812 Mod 2 Pageprinter
ibm3816 (1)	Postprocesses **troff** output for IBM 3816 Pageprinter
ibm5587G (1)	Postprocesses **troff** output for IBM 5587G with 32x32/24x24 cartridge
lp(1)	Sends requests to a line printer
lpr(1)	Enqueues print jobs
lprm(1)	Removes jobs from the line printer spooling queue
lpstat (1)	Displays line printer status information
lptest(1)	Generates the line printer ripple pattern
lsallqdev (1)	Lists all configured printer queue device names within a queue
lsvirprt (1)	Displays the attribute values of a virtual printer

mkque(1)	Adds a printer queue to the system
mkquedev (1)	Adds a printer queue device to the system
mkvirprt (1)	Makes a virtual printer
pac(1)	Prepares printer/plotter accounting records
piobe(1)	Print Job Manager for the printer backend
pioburst (1)	Generates burst pages (header and trailer pages) for printer output
piocmdout (3)	Subroutine that outputs an attribute string for a printer formatter
piodigest (1)	Digests attribute values for a virtual printer definition and stores
pioexit (3)	Subroutine that exits from a printer formatter
pioformat (1)	Drives a printer formatter
piofquote (1)	Converts certain control characters destined for PostScript printers
piogetstr (3)	Subroutine that retrieves an attribute string for a printer formatter
piogetvals(3)	Subroutine that initializes Printer Attribute database variables for printer formatter
piomsgout (3)	Subroutine that sends a message from a printer formatter
pioout(1)	Printer backend's device driver interface program
piopredef (1)	Creates a predefined printer data stream definition
proff(1)	Formats text for printers with personal printer data streams
qadm(1)	Performs system administration for the printer spooling system
qconfig (4)	Configures a printer queueing system.
qstatus(1)	Provides printer status for the print spooling system
restore (3)	Restores the printer to its default state
rmque(1)	Removes a printer queue from the system
rmquedev (1)	Removes a printer or plotter queue device from the system
rmvirprt (1)	Removes a virtual printer
splp(1)	Changes or displays printer driver settings
xpr(1)	Formats a window dump file for output to a printer.

Terminals for BSD 4.3 System Managers

Traditionally, 4.3 BSD system managers enable or disable terminal ports by modifying the **/etc/ttys** file and sending a **HUP** signal to the **init** program.

This operating system stores terminal port information in the ODM and starts terminals when the **init** program reads the **/etc/inittab** file. In this operating system, use the Web-based System Manager Devices application or SMIT to configure terminal ports.

There is no fixed mapping between the port and the device special file name in the **/dev** directory. Consequently, it is confusing to system managers who are new to this operating system which port is to be configured. When using SMIT, the first planar serial port (physically labeled s1) is referred to as location `00-00-S1`, adapter `sa0`, and port `s1` in the SMIT menus. The second planar serial port (physically labeled s2) is referred to as location `00-00-S2`, adapter `sa1`, and port `s2`.

Use the **penable** and **pdisable** commands to enable and disable a port.

termcap and terminfo

Like System V, this operating system uses **terminfo** entries in **/usr/lib/terminfo/?/*** files. Users with 4.3 BSD Systems might find the following commands helpful:

captoinfo (1)	Converts a **termcap** file to a **terminfo** file
tic(1)	Translates the **terminfo** files from source to compiled format.

This operating system includes source for many **terminfo** entries. Some of these might need to be compiled with the **tic** command. The **termcap** file is provided in **/lib/libtermcap/termcap.src** file.

UUCP for BSD 4.3 System Managers

This operating system provides System V Basic Networking Utilities (BNU) which are often referred to as the HDB UUCP.

Dialers (4)	Lists modems used for BNU remote communications links
Maxuuxqts(4)	Limits the number of instances of the BNU **uuxqt** daemons that can run
Permissions(4)	Specifies BNU command permissions for remote systems
Poll(4)	Specifies when the BNU program should poll remote systems
Systems(4)	Lists remote computers with which the local system can communicate
rmail (1)	Handles remote mail received through BNU
uucheck (1)	Checks for files and directories required by BNU
uuclean (1)	Removes files from the BNU spool directory
uucleanup (1)	Deletes selected files from the BNU spooling directory
uucpadm (1)	Enters basic BNU configuration information
uudemon.admin(1)	Provides periodic information on the status of BNU file transfers
uudemon.cleanu(1)	Cleans up BNU spooling directories and log files
uudemon.hour(1)	Initiates file transport calls to remote systems using the BNU program
uudemon.poll(1)	Polls the systems listed in the BNU Poll file
uulog(1)	Provides information about BNU file-transfer activities on a system
uupoll(1)	Forces a poll of a remote BNU system
uuq(1)	Displays the BNU job queue and deletes specified jobs from the queue
uusnap(1)	Displays the status of BNU contacts with remote systems
uustat (1)	Reports the status of and provides limited control over BNU operations.

This operating system also provides the 4.3 BSD **uuencode** and **uudecode** commands. The HDB **uugetty** command is not supported.

For more information, see the lists of BNU files, file formats, and directories in *AIX 5L Version 5.1 System User's Guide: Communications and Networks*.

Appendix B. Notices

Index

Readers' Comments — We'd Like to Hear from You

AIX 5L Version 5.1
System Management Concepts:
Operating System and Devices

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You

IBM

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Corporation
Publications Department
Internal Zip 9561
11400 Burnet Road
Austin, TX
78758-3493

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A