Bull

AIX System Management Concepts: Operating System and Devices

AIX

ORDER REFERENCE
86 A2 21KX 02

Bull

AIX System Management Concepts: Operating System and Devices

AIX

Software

May 2000

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 21KX 02

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the right of proprietors of trademarks mentioned in this book.

AIX® is a registered trademark of International Business Machines Corporation, and is being used under licence.

UNIX is a registered trademark in the United States of America and other countries licensed exclusively through the Open Group.

Year 2000

The product documented in this manual is Year 2000 Ready.

About This Book

This book contains information for understanding the concepts of the AIX operating system and the tasks that you perform in your day–to–day life as a system administrator, as well as the tools AIX provides for system management. Use this book together with the *AIX 4.3 System Management Guide: Operating System and Devices*, 86 A2 99HX.

Note: You can also find the information in this book on the "Hypertext Library for AIX 4.3" CD-ROM. This online documentation is designed for use with an HTML version 3.2 compatible web browser.

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 4.3 System User's Guide: Operating System and Devices*, 86 A2 97HX
- *AIX 4.3 System User's Guide: Communications and Networks*, 86 A2 98HX
- *AIX 4.3 Installation Guide*, 86 A2 43GX

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

Note: The troubleshooting sections are helpful when you know the cause of your problem. If you encounter a problem for which you do not know the cause, refer to the *AIX Version 4.3 Problem Solving Guide and Reference*.

Overview of Contents

This book contains the following chapters and appendixes:

- Chapter 1, "AIX System Management," introduces the major tools provided to assist you in system management and the features specific to the system.
- Chapter 2, "System Startup and Shutdown," contains conceptual information about starting and stopping the system as well as procedural information to guide you in performing these tasks.
- Chapter 3, "Security," introduces the security features, including the Trusted Computing Base (TCB), the **virscan** command that detects viruses, auditing, and access control.
- Chapter 4, "Administrative Roles," contains conceptual information about roles and authorization as well as procedural information to guide you in setting up and maintaining roles.
- Chapter 5, "Users and Groups," discusses the features available to manage individual users and illustrates procedures used to manage groups of users.

- Chapter 6, ”Logical Volumes,” contains an overview of managing logical volume storage concepts and procedures.
- Chapter 7, ”File Systems,” contains in–depth concepts and procedures for managing files, directories, and file systems.
- Chapter 8, ”Paging Space and Virtual Memory,” discusses the practical uses of paging space allocations, how to create and maintain paging space for your system, and the Virtual Memory Manager program.
- Chapter 9, ”Backup and Restore,” introduces the commands and concepts used to save your data and restore it from backup.
- Chapter 10, ”System Environment,” discusses basic environment components you can manage and how to work with these components. Also included are instructions that explain how to change the message of the day, broadcast messages to users, and work with profiles.
- Chapter 11, ”National Language Support,” introduces the special considerations necessary for managing a system in various languages and time zones.
- Chapter 12, ”Process Management,” introduces system processes and how to use them.
- Chapter 13, ”System Resource Controller and Subsystems,” introduces this feature and discusses ways to use the controller.
- Chapter 14, ”System Accounting,” introduces the conceptual background information needed to use the wide array of system accounting commands and subroutines.
- Chapter 15, ”Web-based System Manager” describes Web-based System Manager in both stand–alone and Client–Server environments.
- Chapter 16, ”System Management Interface Tool,” describes the use and structure of the System Management Interface Tool (SMIT). SMIT is a command–building user interface that assists the system manager in constructing and recreating many system management tasks. The interface can be used in an ASCII or windows environment.
- Chapter 17, ”CDE Desktop,” describes the CDE Desktop.
- Chapter 18, ”Documentation Search Service” allows you to search online HTML documents on your documentation server that have been indexed. This section provides information on installation and configuation, and how to create your own indexes to search User created documents.
- Chapter 19, ”Power Management,” describes the system management and general user tasks for Power Management.
- Chapter 20, ”Devices,” provides a brief overview of the methods used by the operating system to manage a wide range of devices.
- Chapter 21, ”Tape Drives,” discusses system management functions related to tape drives.
- Appendix A, ”AIX for BSD System Managers,” contains information for system managers who are familiar with the 4.3 BSD UNIX or System V operating system. This chapter explains both similarities and differences.

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.

Other Key Sources of System Management Information

Publications Covering Other Aspects of System Management

In today's computing environment, it is impossible to create a single book that addresses all the needs and concerns of a system administrator. While this guide cannot address everything, we have tried to structure the rest of our library so that a few key books can provide you with direction on each major aspect of your job.

The following books cover other key topics of interest to you:

- *AIX 4.3 System Management Guide: Communications and Networks*, 86 A2 31JX, covers network administration and maintenance.
- *AIX 4.3 Installation Guide*, 86 A2 60AP
- Problem Solving and Messages:
  - *AIX Version 4.3 Problem Solving Guide and Reference*, 86 A2 32JX
  - *AIX Messages Guide and Reference*, 86 A2 33JX
- *AIX General Programming Concepts: Writing and Debugging Programs*, 86 A2 34JX, introduces you to the programming tools and interfaces available for writing and debugging application programs.
- *AIX Communications Programming Concepts*, 86 A2 35JX, provides conceptual and procedural information about various communications programming tools.
- *AIX Files Reference*, 86 A2 79AP.
- Monitoring and tuning system performance:
  - *AIX Performance Tuning Guide*, 86 A2 72AP, describes the performance monitoring and tuning tools available with the base operating system.
  - *Performance Toolbox 1.2 and 2.1 for AIX: User's Guide*, 86 A2 10AQ, describes the additional monitoring tools available with Performance Toolbox for AIX.
- *AIX 4.3 Network Installation Management Guide and Reference*, 86 A2 17HX, covers configuration and maintenance of diskless workstations.
- *Distributed SMIT 2.2 for AIX: Guide and Reference*, 86 A2 09AQ , covers information about the Distributed System Management Interface Tool (DSMIT).
- *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*, 86 A2 85AT, covers advanced tasks in customizing the appearance and behavior of the Common Desktop Environment (CDE).

AIX Support for the X/Open UNIX95 Specification

Beginning with AIX Version 4.2, the operating system is designed to support the X/Open UNIX95 Specification for portability of UNIX–based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification. Beginning with Version 4.2, AIX is even more open and portable for applications.

At the same time, compatibility with previous AIX releases is preserved. This is accomplished by the creation of a new environment variable, which can be used to set the system environment on a per–system, per–user, or per–process basis.

To determine the proper way to develop a UNIX95–portable application, you may need to refer to the X/Open UNIX95 Specification, which can be obtained on a CD–ROM by ordering the printed copy of *AIX Commands Reference*, order number 86 A2 38JX to 86 A2 43JX, or by ordering *Go Solo: How to Implement and Go Solo with the Single Unix Specification*, a book which includes the X/Open UNIX95 Specification on a CD–ROM.

Reference Information

The following publications contain information on the commands and files used in the operating system.

- *AIX Commands Reference*, 86 A2 38JX to 86 A2 43JX, is a six–volume set that contains supported commands in alphabetical order.
- *AIX Files Reference*, 86 A2 79AP, contains information on the files available with the operating system.

The following books contain other information you may find useful in day–to–day managing:

- *AIX Quick Reference*, 86 A2 55AP, contains brief descriptions of frequently used commands, along with brief summaries of the commands.
- *AIX INed Editor User's Guide* contains information on the INed editor.

Ordering Publications

You can order this book separately from Bull Electronics Angers S.A. CEDOC. See address in the Ordering Form at the end of this book.

To order additional copies of this book, use order number 86 A2 21KX.

Use *AIX and Related Products Documentation Overview* for information on related publications and how to obtain them.

Contents

Chapter 1. AIX 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 of the system administrator's activities are straightforward enough to be properly called administration. This and related guides are intended to help system administrators with their numerous duties.

The System Administrator's Objectives

The system administrator has three main objectives:

- See to it 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 his or her control. He or she must also understand the interconnected environment in which almost all of today's systems exist, and the effects of that environment on the local system's function and performance.

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 organization's priorities
  - 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 properly configured
  - 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

AIX–Unique Aspects of System Management

AIX 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
- the **man** command

Available Interfaces

In addition to conventional command–line system administration, AIX provides these optionally installable 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 environments.

  See System Management Interface Tool (SMIT) Overview, on page 16-1 for more information on managing your system with SMIT.
- Distributed System Management Interface Tool (DSMIT), an ASCII–based user interface that allows you to perform system administration tasks on ”clusters” of workstations, including machines running Sun/OS 4.1.3 and HP/UX 9.0. This product can be purchased separately. For more information on this product, see the *Distributed SMIT 2.2 for AIX: Guide and Reference*.

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 overview of logical volume storage, on page 6-2.

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 overview of the System Resource Controller, on page 14-2.

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.

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 Commands Reference*. Also see the **man** command for BSD 4.3 system managers, on page A-25.

AIX Updates

For detailed information on software updates, also termed service updates, see ”Installing Optional Software and Service Updates” in *AIX 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 boots, the system initiates a complex set of tasks. Under normal conditions, these tasks are performed automatically. For additional information about booting the system, see:

- Understanding the Boot Process, on page 2-3
- Diagnosing Boot Problems in *AIX 4.3 System Management Guide: Operating System and Devices*.

Booting the System

There are some situations when you want to instruct the system to boot; 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 in *AIX 4.3 System Management Guide: Operating System and Devices*.
- Rebooting a Running System in *AIX 4.3 System Management Guide: Operating System and Devices*.
- Booting a System That Crashed in *AIX 4.3 System Management Guide: Operating System and Devices*.

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” in *AIX 4.3 System Management Guide: Operating System and Devices*.

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 in *AIX 4.3 System Management Guide: Operating System and Devices*.
- Changing System Run Levels in *AIX 4.3 System Management Guide: Operating System and Devices*.
- Changing the /etc/inittab File in *AIX 4.3 System Management Guide: Operating System and Devices*.

Understanding the Boot Process

During the boot process, the system tests the hardware, loads and executes 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”, on page 2-4.
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 2-8.

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 2-9.

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

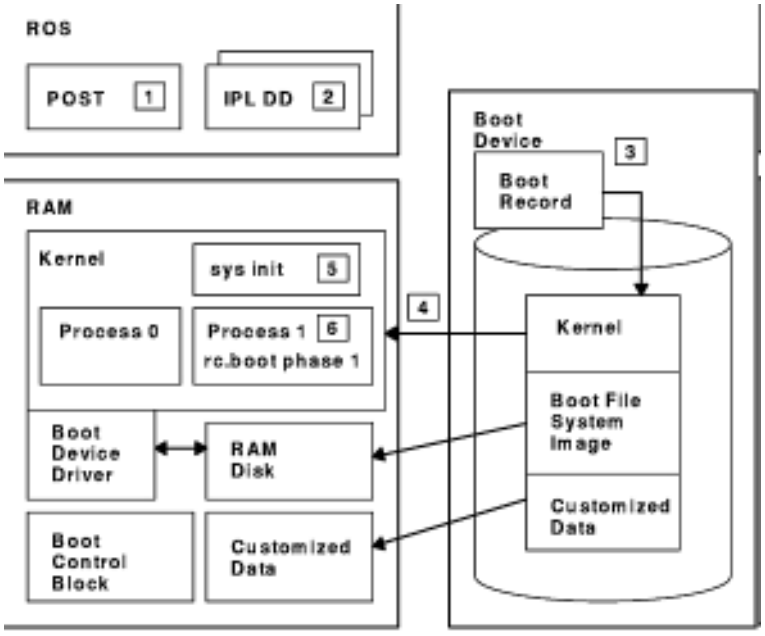
Most users perform a hard disk boot when starting the system for general operations. The system finds all information necessary to the boot process on its disk drive.

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 Init Phase diagram illustrates the kernel initialization that takes place before the system boot process is started.



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 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.

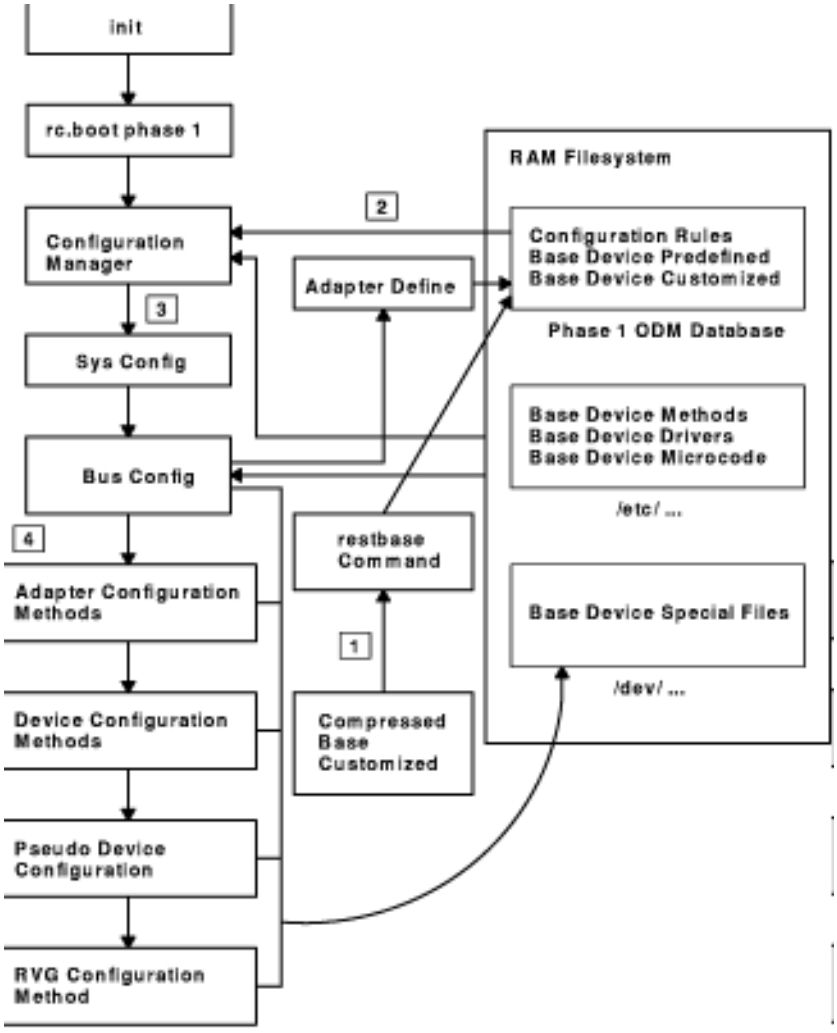
Note: The system maintains a default boot list located in ROS and a user boot list stored in NVRAM, for a normal boot. Separate default and user boot lists are also maintained for booting from the Service key position.

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 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 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 RAM file system, and base customized device information.
5. Control is passed to the kernel, which begins system initialization.
6. Process 1 executes **init**, which executes phase 1 of the **rc.boot** script.

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

Base Device Configuration Phase

The Base Device Configuration Phase diagram illustrates this part of the boot process.



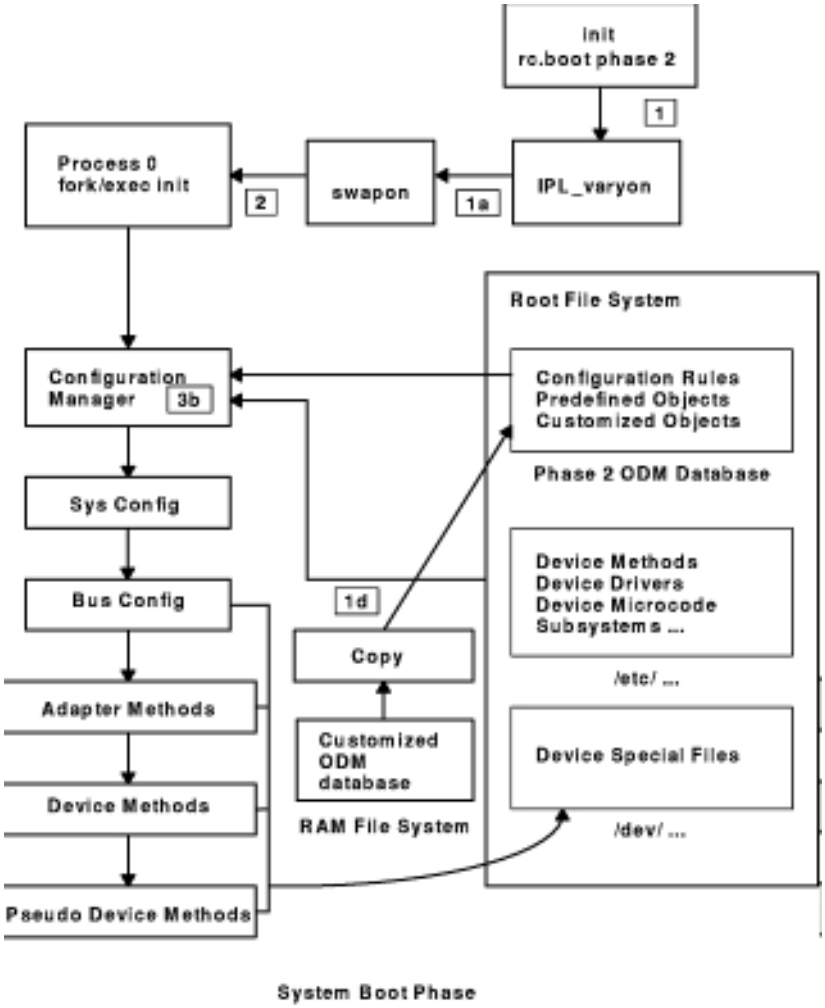
Base Device Configuration Phase

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 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 diagram illustrates the steps involved in the system boot process.



System Boot Phase

1. The **init** process starts phase 2 execution 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 the RAM file system.
   c. Run **swapon** 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. Unmount temporary mounts of hard disk file systems and then perform permanent mounts of root, /**usr**, and /**var**.

f. 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 executes the processes defined by records in the /**etc**/**inittab** file. One of the instructions in the /**etc**/**inittab** file executes phase 3 of the **rc.boot** script, which includes the following steps:
   a. Mount /**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 may arise when a service boot is needed to perform special tasks such as installing new or updated software, performing diagnostic checks, or maintenance. In this case, the system starts from a bootable medium (CD–ROM, tape), 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. The events can be outlined as follows:

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, which can be altered to suit your requirements using the **bootlist** command. 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: The system maintains a default boot list, located in ROS, and a user boot list, stored in NVRAM, for normal boot. 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 execution 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 executing 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 CD–ROM, tape, or the network is used to boot the system, the rootvg volume group (RVG) is not varied on, since the RVG may not exist (as is the case when installing the operating system on a new system). Network configuration may 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's 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 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 determine the type of boot.

A service boot RAM file system might not have the logical volume routines, since the rootvg volume group may 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 **ssh** (simple shell) 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 should 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's 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. There is a **rc.boot** stanza in the **inittab** file 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. There is a prototype file associated with each type of boot device. The prototype file is a template of files making up the RAM file system. The **mkfs** command is used by the **bosboot** 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 may 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

Chapter 3. Security

This chapter explains advanced system security topics. System security is covered in *AIX 4.3 System User's Guide: Operating System and Devices*.

Topics covered in this chapter are:

Security Administration

Proper system administration is vital to maintaining the security of information resources on a computer system. AIX security is based on establishing and maintaining proper access control and accountability policies. It is an administrator's responsibility 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 3-2
: Determines how users are identified and how their identities are authenticated.

Trusted Computing Base (TCB), on page 3-11
: Enforces the information security policies of the system.

Auditing, on page 3-16
: 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

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 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.

The operating system allows for greater control, if desired, with extended attributes. Security information can also 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,** **/etc/security/lastlog** files. Several of these attributes control how a user can login, and these attributes can be configured to lock the user account (prevent further logins) under specified conditions.

Once the user's account is locked, the user will not be able to login until the system administrator resets the user's **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 should 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 may 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 may be defined in **/etc/security/login.cfg**.

> **Note:** The root user should always be 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 4.3 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, AIX 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 proper password security, all passwords should 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 3-5 for more information.

Restrictions should be applied 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 proper 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** should 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 appear 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 will not be able 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			

Restrictions should be set 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 should set **minother** to 0. Since most words in this dictionary file do not contain characters that fall into the **minother** category, setting **minother** to 1 or more would eliminate 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 Technical Reference* contains a description of the **pwdrestrict_method**, the subroutine interface that specified password restriction methods must conform to. To properly extend the password composition restrictions, the system administrator must program this interface when writing a password restriction method. Caution is advised 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 should be examined when you generate audit records. Refer to the *AIX 4.3 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.

Introduction

Attention: Any operating environment may have unique security requirements that are not addressed in these guidelines. To ensure a secure system, system administrators may 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

Every system should 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 the ”Managing Users and Groups Tasks” table in *AIX 4.3 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 on 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 properly. 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 **li** or **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 4.3 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: user, system administrator, and system–defined.

User Groups

In general, create as few user groups as possible.

Groups should be 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 should be 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

There are several system–defined groups. 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 should 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 4.3 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 4.3 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 AIX systems.

- 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 it should never be shared. Only one person, the system administrator, should know the root password. System administrators should only operate as root to perform system administration functions that require root privileges, and then return to a 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 should have the root password to gain root authority by using the **su** command. Immediately after the system is installed, the root account should be given a password.

The root account should always be 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 executed. 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, that is, that only root can write to. It is recommended also that you do not create a **.profile** file in the / (root) directory. The **.profile** files should only be in users' **$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's **.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 will ensure that root's environment is used during the session. If a system administrator does operate as root in another user's session, then the system administrator should specify full path names throughout the session.

- Protect the **IFS** (input field separator) 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" in *AIX 4.3 System Management Guide: Operating System and Devices*, 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 3-16.

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 should 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", on page 3-11.

Administrative Roles

AIX Version 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 Administration", on page 4-1.

Networks and Communications Security

Networks and communications security are described fully in *AIX 4.3 System Management Guide: Communications and Networks*.

- TCP/IP Security
- TCP/IP Command Security
- Understanding BNU Security
- Data Security and Information Protection
- NIS Security
- 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's 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 should include 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 a file's contents 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's **.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 **sysck** 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 may 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, there are two problems:

- 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. The **watch** command can be used to execute a specified program and display the stream of audit records (if any) that are generated during the execution 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. Since 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 executed by anyone and are not set user ID (SUID) or set group ID (SGID). If the command is expected to be run on the trusted path (for example, using the **vi**, **grep**, and **cat** commands), its TCB bit should be set. 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 executable only by root, members of an administrative group, and members who are specified in the extended access control list (ACL) entries. Since they usually perform privileged operations, some of these commands may 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)

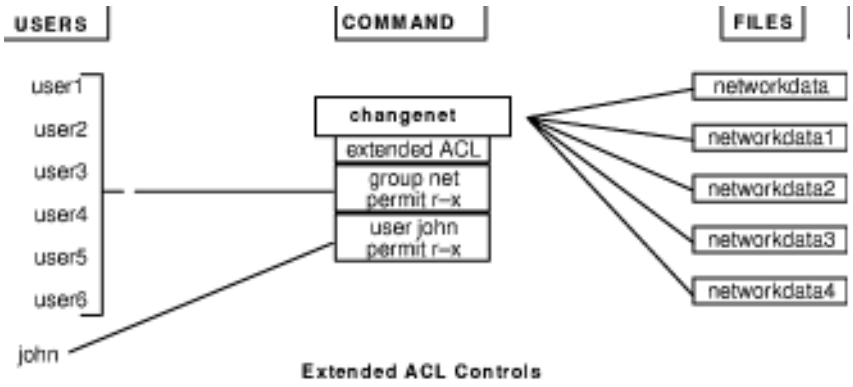
For an example of a typical administrative user command scenario, consider network files containing critical network configuration information:

```
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 Extended ACL Controls figure shows how the extended ACL is associated with the command instead of with the data files themselves.



Extended ACL Controls

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 should be 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 should be writable so users can send messages to each other and floppy disks that should be 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 communications path will be disabled. The path can be properly enabled only by reinstalling the system.

The trusted communication path is based on:

- A trusted command interpreter (**tsh** command) that only executes 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 secure attention key (SAK), which 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 assures 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 executes 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 execute the user login shell.

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, and 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, which 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 should include the name of the auditable event, the success or failure of the event, and any additional event–specific information that would be 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, which 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 which 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 prior to 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 may 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. Also, since subroutines are auditable, the extendable kernel definition, with no fixed SVC numbers, makes it difficult to record events by number, since 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

There are two different types of audit event selection: 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) and results (success/failure) 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 (prior to 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 this 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.

Chapter 4. Administrative Roles

AIX Version 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

Roles consist of authorizations that allow a user to execute functions normally requiring 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 group **security**.
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 group **security**.
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, Customer Engineer or Service Support Representative to run diagnostics and diagnostic tasks. The user must have **system** 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, and 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.

There are two types of authorizations:

Primary Authorization Allows a user to execute a specific command. For example, RoleAdmin authorization is a primary authorization allowing a user administrator to execute 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 group **security**. 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 terminates.

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 group **audit**.

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's gecos (general information) field. If the user does not have UserAdmin authorization but is in group **security**, they can change any non–administrative user's 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 terminates.

lsrole Views a role.

	mkrole	Creates a role. If the user administrator does not have the RoleAdmin authorization, the command terminates.
	rmrole	Removes a role. If the user administrator does not have the RoleAdmin authorization, the command terminates.
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 4.3 System Management Guide: Operating System and Devices* for a mapping of commands to authorizations.

Chapter 5. Users and Groups

This chapter contains information for managing users and groups. Also included in this chapter is information on disk quotas.

Disk Quota System Overview

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

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 1KB disk blocks or files below which the user should 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 will be 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 may 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: It is recommended that disk quotas not be established 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 2MB in volume groups that include physical volumes smaller than 300MB, 4MB 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, span multiple physical volumes, and 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 may or may not be contiguous with one another or with other blocks previously allocated to the file. In AIX 4.1, a given file system can be defined as having a fragment size of less than 4KB (512 bytes, 1KB, 2KB).

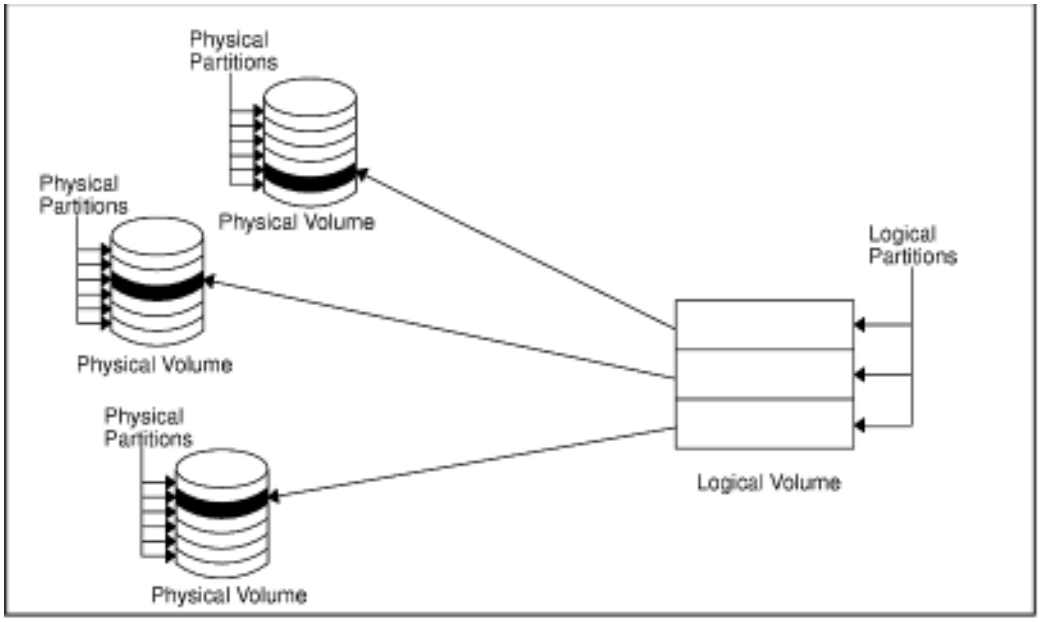
After installation, the system has one volume group (the rootvg volume group) consisting of a base set of logical volumes required to start the system and any others you specify to the installation script. Any other physical volumes you have connected to the system can be added to a volume group (using 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:

- Forcibly Varying On, on page 6-8
- Nonquorum Volume Groups, on page 6-8

Logical Volume Storage Concepts

The five basic logical storage concepts are physical volumes, volume groups, physical partitions, logical volumes, and logical partitions. The Volume Group figure illustrates the relationships among these concepts.



Volume Group

A volume group composed of three physical volumes with 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 may 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. Physical partitions are discussed below.

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 will be using 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 may require only one volume group to contain all the physical volumes attached to the system. You may 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 should contain 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 of a physical volume that is to be replaced.

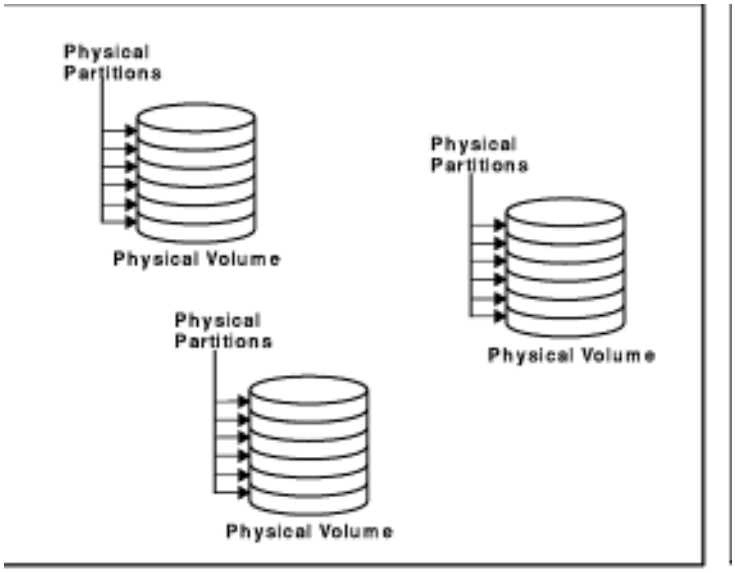
A VG that is created with smaller physical and logical volume limits can be converted to big format which can hold more PVs (upto 128) and more LVs (upto 512). This operation requires that there be enough free partitions on every PV in the VG for the Volume group descriptor area (VGDA) expansion. The number of free partitions required depends on the size of the current VGDA and the physical partition size. Since 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 will be migrated to other free partitions on the same disk. The rest of the physical partitions will be 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, you should 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 may fail since the partition number may 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. Since the VGDA space has been increased substantially, every VGDA update operation (creating an LV, changing an LV, adding a PV, etc.) may take considerably long duration.

> **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 prior to AIX Version 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). A Volume Group Containing Three Physical Volumes shows the relationship between physical partitions on physical volumes and volume groups.



A Volume Group Containing Three Physical Volumes

Logical Volumes

After you create a volume group, you can create logical volumes within that volume group. A *logical volume,* although it may 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, will be `closed`. It will become `open` when, for example, a file system has been created in the logical volume and mounted.

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

The system will allow 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 (for example, 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 7-2.

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 9GB disk and 4Mb 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 properly. 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 32/factor. For example, a factor of 2 decreases the maximum number of disks in the volume group to 16 (32/2).

Note: Once you convert a volume group, you cannot import it back to any level prior to AIX Version 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 Version 4 Kernel Extensions and Device Support Programming Concepts* and ”Logical Volume Programming Overview” in *AIX Version 4 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. For more information about the meanings of the physical volume status that can be displayed from the **varyonvg** command, refer to the **lvm_varyonvg** subroutine.

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 may 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 is 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 may 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; all other possible problem sources such as hardware, cables, adapters, and power sources should be checked before proceeding. Overriding the quorum failure during a vary–on process should be 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 will be removed from these physical volumes. Once this is done, these physical volumes will no longer take part in quorum checking, nor will they be 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 may 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. The missing physical volumes should be unconfigured and powered off until they can be diagnosed and repaired.

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

1. Use the **chpv –vr** command to temporarily remove the volume from the volume group. This physical volume will no longer be factored in quorum checking. However, in a two–disk volume group, this command will fail if you try the **chpv** command on the disk which contains the two VGDA/VGSAs. The command will 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 6-12). However, volume group size also plays a part as is explained below.

To protect against adapter and power supply failure, you need to 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 6-2.

When to Create Separate Volume Groups

You may 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, deactivating it (using **varyoffvg**), then exporting the group (using **exportvg**). After updating the system software, you can reintroduce the user–defined volume group (using **importvg**), 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 bottlenecking, 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 in the ”Logical Volume Storage Overview”, on page 6-7. 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” in *AIX 4.3 System Management Guide: Operating System and Devices*). 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 and intra–disk allocation for a logical volume.

High Availability in Case of Adapter or Power Supply Failure

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

- Use two adapters, located in the same or different cabinets. Locating the adapters in different cabinets protects against losing both adapters if there is a power supply failure in one cabinet.
- 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 will remain 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” in *AIX 4.3 System Management Guide: Operating System and Devices*).
- 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.

Decide on the Size of Physical Partitions

The physical partition size is set when the volume group is created. The default size is 4MB. The default is designed to suit most sites and systems but may not be appropriate in every case. You can choose a partition size as small as 1MB 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 4MB, you lose some sizing flexibility and may also waste space. For example, if you have 20MB partitions, then your JFS log will have to be 20MB when it only needs 4MB. Some waste may be an acceptable tradeoff if the particular site or system requires larger partitions.

Note that you may only create and extend physical partitions in increments that are a factor of their size; for example, 20MB partitions are created or extended in 20MB 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 Version 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 Performance Tuning 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:

Prerequisites

It is important that you understand the material contained in the Logical Volume Storage Overview, on page 6-2.

Analyze Needs for Performance and Availability

Determine whether the data that will be 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. Writes, unless disks are attached to different adapters, always cost the same because you must update all copies, but you only need to Read one.

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 Version 4.3.3, the boot 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**.

A dump to a mirrored logical volume results in an inconsistent dump and therefore should be avoided. Since the default dump device is the primary paging logical volume you should create a seperate dump logical volume if you mirror your paging logical volumes, and therefore if you mirror your root volume group you should create a seperate dump logicial volume also.

Determine 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, two different scheduling policies for writing to disk can be set for a logical volume with multiple copies, *sequential* and *parallel*.

The 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.

The parallel–scheduling policy starts the Write operation for all the physical partitions in a logical partition at the same time. When the Write operation to the physical partition that takes the longest to complete finishes, the Write operation is completed.

For Read operations on mirrored logical volumes with a sequential–scheduling policy, the primary copy is read. 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. Thus the bad block that prevented the first Read from completing is patched for future access.

Specifying mirrored logical volumes with a parallel–scheduling policy may 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.

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

Mirror Write Consistency (MWC) identifies which logical partitions may be inconsistent if the system or the volume group is not shut down properly. When the volume group is varied back online for use, this information is used to make logical partitions consistent again.

If a logical volume is using 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.

When MWC is being used, system performance can be adversely affected. This is caused by the overhead of logging or journalling that a Write request is active in a Logical Track Group (LTG) (32 4K–byte pages or 128K bytes). 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. When MWC is not used,

the mirrors of a mirrored logical volume can be left in an inconsistent state in the event of a system or volume group crash.

After a crash, any mirrored logical volume that has MWC turned off should do a forced sync (**syncvg –f –l LV*name***) before the data within the logical volume is used. With MWC turned off, Writes outstanding at the time of the crash can leave mirrors with inconsistent data the next time the volume group is varied on. An exception to this 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 really than a non–mirrored logical volume with respect to a Write. When LVM completely finishes with a Write request the data has been written to the drive(s) below LVM. Until LVM issues an **iodone** on a Write the outcome of the Write is unknown. Any blocks being written that have not been completed (**iodone**) when a machine crashes should be rewritten whether mirrored or not and regardless of the MWC setting.

MWC only makes mirrors consistent when the volume group is varied back online after a crash by picking one mirror and propagating that data to the other mirrors. MWC does not keep track of the latest data it only keeps track of LTGs currently being written, therefore MWC does not guarantee that the lastest data will be propagated to all the mirrors. 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).

Choose an Inter–Disk Allocation Policy for Your System

The inter–disk allocation policy specifies the number of disks on which a logical volume's physical partitions 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 will succeed 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.

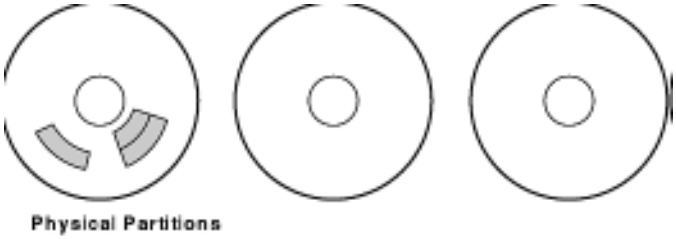
Inter–Disk Settings for a Single Copy of the Logical Volume

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* = **maximum**), 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 should contain 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. An 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.

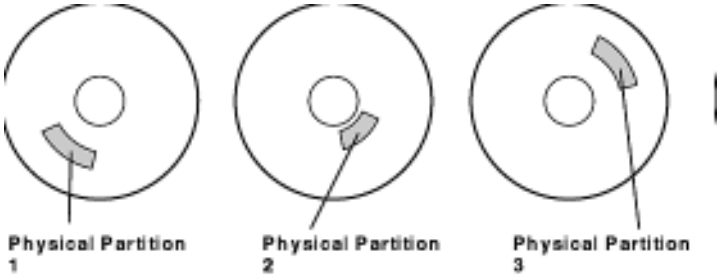
The Minimum Inter–Disk Allocation Policy figure illustrates a minimum inter–disk allocation policy:



Minimum Inter-Disk Allocation Policy
Physical Partitions Are All on a Single Disk.

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 should only be used with mirrored logical volumes.

The Maximum Inter–Disk Allocation Policy figure illustrates a maximum inter–disk allocation policy:



Maximum Inter-Disk Allocation Policy
Physical Partitions Are on Multiple Disks.

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

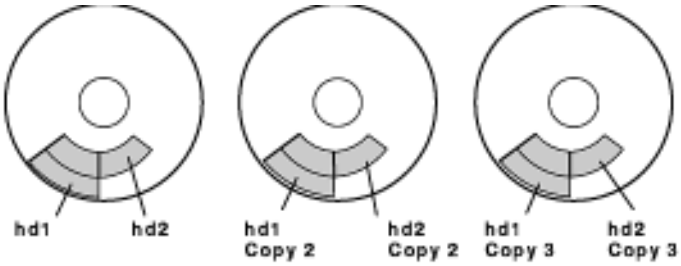
The allocation of a single copy of a logical volume on disk is fairly straightforward. When you create mirrored copies, however, the resulting allocation is somewhat complex. The figures that follow show **minimum** and **maximum** inter–disk *(Range)* settings for the first instance of a logical volume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

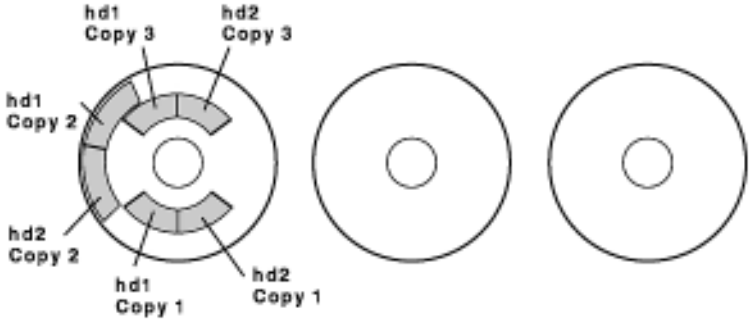
The setting *Strict* = **y** means that each copy of the logical partition will be placed on a different physical volume. The setting *Strict* = **n** means that the copies are not restricted to different physical volumes.

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

The Minimum Inter–Disk Policy/*Strict* figure illustrates a minimum inter–disk allocation policy with differing *Strict* settings:

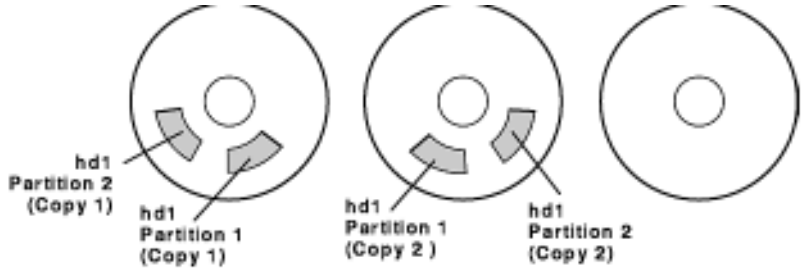


Minimum Inter-Disk Policy with a
Single Logical Volume Copy per Disk (*Strict* = y)

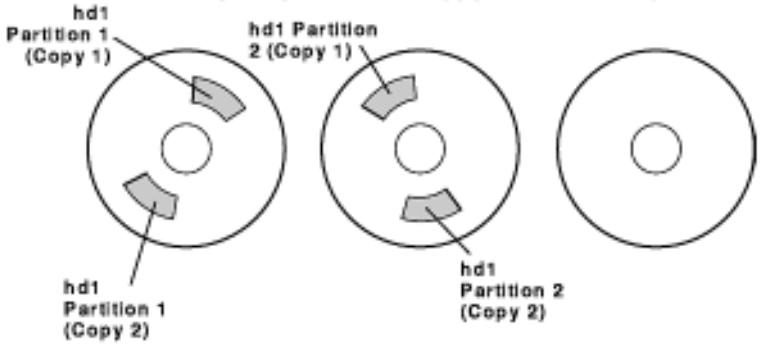


Minimum Inter-Disk Policy with
Multiple Logical Volume Copies per Disk (*Strict* = n)

The Maximum Inter–Disk Policy/*Strict* figure illustrates a maximum inter–disk allocation policy with differing *Strict* settings:



Maximum Inter-Disk Policy with a
Single Logical Volume Copy per Disk (*Strict* = y)



Maximum Inter-Disk Policy with Multiple Logical
Volume Copies per Disk (*Strict* = n)

Choose 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.

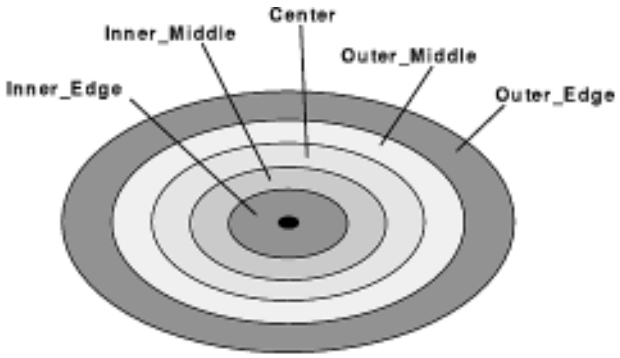
The file system log is a good candidate for allocation at the center of a physical volume because it is used by the operating system so often. At the other extreme, the boot logical volume is used infrequently and therefore should be allocated at the edge or middle of the physical volume.

The general rule, then, 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 should be allocated. This rule has two important exceptions:

1. Logical volumes on 200MB, 540MB, or 1GB disks that contain large, sequential files should be 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** should be 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. Otherwise, see ”Performance Implications of Disk Mirroring” in the section on performance–related installation guidelines in the *AIX Performance Tuning Guide*.

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: *outer edge*, *inner edge, outer middle*, *inner middle***,** and *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.

The Five Regions of a Disk illustration shows the regions that can be used for allocating physical partitions in a physical volume.



The Five Regions of a Disk

Combining Allocation Policies

If you select inter–disk and intra–disk policies that are not compatible, you may get unpredictable results. The system will assign 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 will take precedence. The system will place all of the partitions for the logical volume on one disk if possible, even if the partitions will 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. Use 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 will allocate 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. Use 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 may import a Version 3.2 created volume group into a Version 4.1 system, and you may import a Version 4.1 volume group into a Version 3.2. system, provided striping has not been applied. Once striping is put onto a disk, its importation into Version 3.2 is prevented. The current implementation of **mksysb** will 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 Version 4.3.3.
3. A dump space or boot logical volume whould not be striped.

To create a 12–partition striped logical volume called `lv07` in *VGName* with a stripe size of 16KB 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 `lv08` in *VGName* with a stripe size of 8KB 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 Performance Tuning Guide*.

Determine 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 will 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 (**mklv**) or later by changing it (**chlv**).

Implement 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 in *AIX 4.3 System Management Guide: Operating System and Devices*), or Add fixed disk without data to new volume group *AIX 4.3 System Management Guide: Operating System and Devices*. To change a volume group to nonquorum status, see Change a User–Defined Volume Group to Nonquorum Status *AIX 4.3 System Management Guide: Operating System and Devices*.
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.

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. 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 the 1016 tracking partitions are used by a disk. The default size of each physical partition during a **mkvg** command is 4MB, which implies that individual disks up to 4GB can be included into a volume group.

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

  ```
  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 will be enforced:

1. The user tries to use **mkvg** 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** option OR

   Use a suitable factor (mkvg –t option) 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 **–t** option of **chvg** command. The value of the **–t** option (factor) with **chvg** command can be chosen in such a way that the new disk can be accommodated within the new limit of (1016 * factor). However, once the volume group is converted, it cannot be imported into AIX 4.3.0 or lower versions. 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 4GB, it will change 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, will also be defined at that physical partition size.

   For RAID systems, the /**dev**/**hdiskX** name used by LVM in AIX may really consist of many non–4GB disks. In this case, the 1016 requirement still exists. LVM is unaware of the size of the individual disks that may really make up /**dev**/**hdiskX**. LVM bases the 1016 limitation on the AIX 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 will not look like the other two copies. If you are in volation of 1016, you may get a false report of a non–mirrored logical volume being stale, or you may get a false indication that one of your mirror copies has gone stale. Also, the **migratepv** command may 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 will fail 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 may 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 would 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 AIX. The enforcement of this 1016 limit is only during **mkvg** and **extendvg**. The data is safe on all versions of AIX.
Change the PP Size Limitation	The PP size limitation will not be 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 would occur in the LVM.
Rebuild the Volume Group	The later versions of AIX mentioned in this document only prevent the future creation of disks in volume groups that will violate the 1016 limitation. Disks that already violate the 1016 limit must be recreated at larger PP sizes.

- In some instances, the user will experience 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 AIX system and **importvg** the volume group into the AIX system. 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 **rootvg**. To provide AIX users more free disk space, when **rootvg** is created, **mkvg** does not use the maximum limit of 32 disks that is allowed into a volume group. Instead in AIX 3.2, the number of disks picked in the install menu of AIX is used as the reference number by **mkvg –d** during the creation of **rootvg**. Beginning with AIX 4.1, 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–install. 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: It is recommended that users 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 filesystem (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 may seem fatal, it is not the case. Once 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's face possible, incomplete importation into other AIX systems. During an **importvg**, the LVM command will scan the LVCB's of all defined logical volumes in a volume group for information concerning the logical volumes. If the LVCB is deleted, the imported volume group will still define the logical volume to the new AIX system, which, is accessing this volume group, and the user can still access the raw logical volume. However, any journaled filesystem information is lost and the associated mount point won't be imported into the new AIX system. The user must create new mount points and the availability of previous data stored in the filesystem is not assured. Also, During this import of logical volume with an erased LVCB, some non–jfs information concerning the logical volume, which is displayed by the **lslv** command, cannot be found. When this occurs, the system uses default logical volume information to populate the logical volume's ODM information. Thus, some output from **lslv** will be inconsistent with the real logical volume. If any logical volume copies still exist on the original disks, the information will not be correctly reflected in the ODM database. The user should 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 7-5.

File System Types

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

Journaled File System

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.

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 AIX 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 have to do with 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 AIX 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 AIX file tree has the following characteristics:

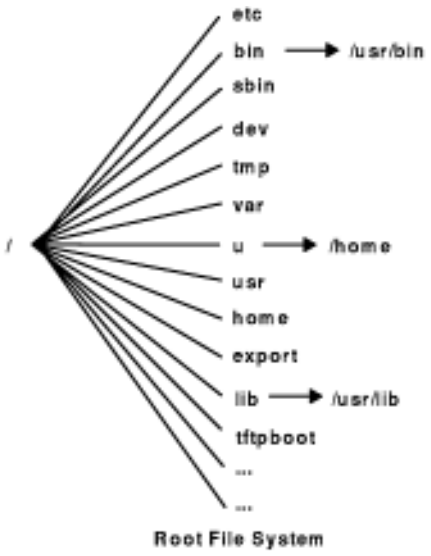
- Files that can be shared by machines of the same hardware architecture are located in the /**usr** file system.
- Variable per–client files, such as spool and mail files, are located in the /**var** file system.
- Architecture–independent, shareable text files, such as manual pages, are located in 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 boot, 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, on page 7-6
- Understanding the /usr File System, on page 7-8
- Understanding the /usr/share Directory, on page 7-10
- Understanding the /var File System, on page 7-11
- Understanding the /export Directory, on page 7-12

Understanding the Root File System

The Root File System diagram shows many of the subdirectories of the root file system:



Root File System

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 7-11 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 and disk–poor machines mount a directory from a remote server over the **/usr** file system. See ”Understanding the /usr File System”, on page 7-8 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 root file system’s **/home** directory. In a network, a server might contain user files that should be accessible from several machines. In this case, the server’s 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.

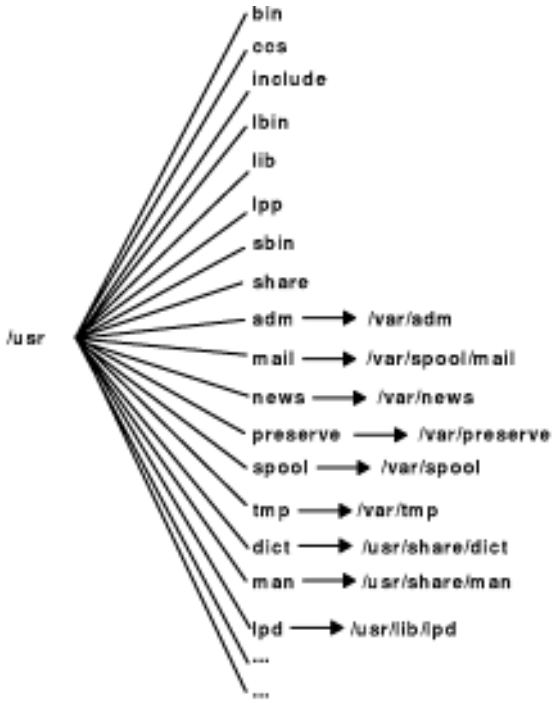
See ”Understanding the /export Directory”, on page 7-12 for more information about the file tree that resides under the **/export** directory.

/lib Symbolic link to the **/usr/lib** directory. See ”Understanding the /usr File System”, on page 7-8 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 /usr File System diagram.



/usr File System

On a standalone machine the /**usr** file system is a separate file system (in the /**dev**/**hd2** logical volume). On a diskless or disk–poor machine, 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 7-10 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, since the **/usr** directory is potentially shared by many nodes and is read–only.

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

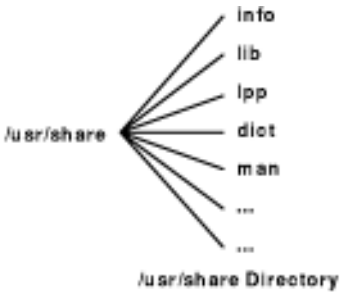
/usr/dict	Symbolic link to the **/usr/share/dict** directory.
/usr/man	Symbolic link to the **/usr/share/man** directory.
/usr/lpd	Symbolic link to the **/usr/lib/lpd** directory.

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 may 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 /usr/share Directory diagram.



/usr/share Directory

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, you will have to 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.

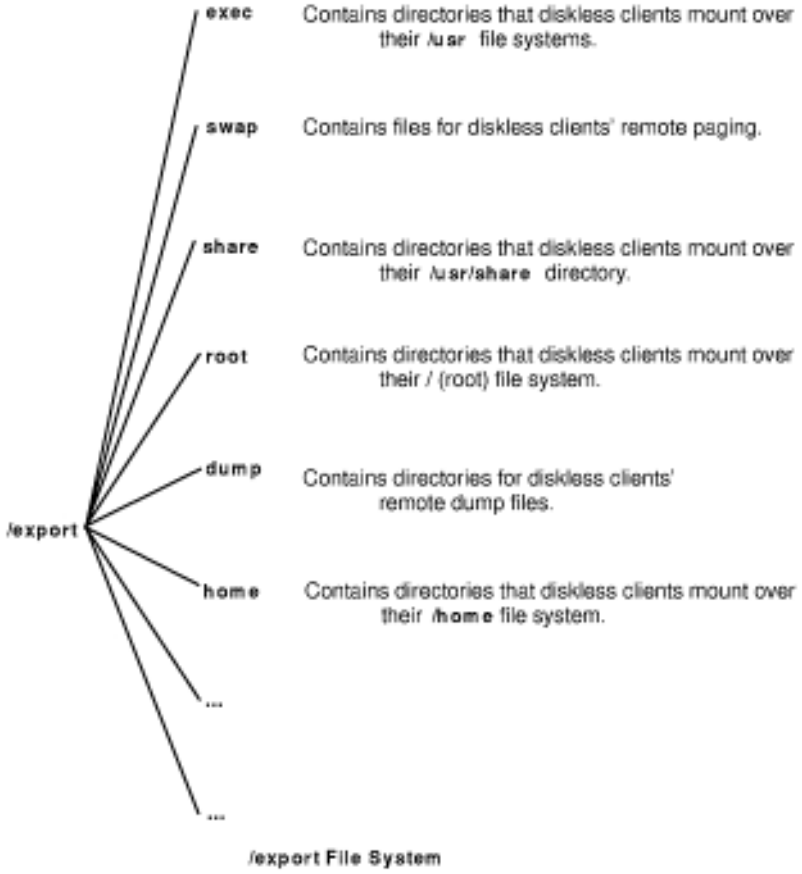
The /var directory diagram shows some of the directories in the /**var** file system.

/var Directory

/**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 or disk–poor clients. 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 /export File System diagram.



/export 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. Since 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** directory
: This directory is mounted over the client's 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**, or Shared Product Object Tree (SPOT) directory
: This directory is mounted over the client's /**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**, or share directory	This directory is mounted over the client's /**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** directory	This directory is mounted over the client's /**home** file system. It contains user directories grouped by client host names. The default location for client home directories is /**export**/**home**.
/**export**/**swap**, or 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** directory	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's host name and by default is found in the /**export**/**dump** directory.
microcode directory	This directory contains microcode for physical devices. The default location is /**export**/**exec**/**RISCAIX**/**usr**/**lib**/**microcode**.

Understanding Data Compression

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’s free space. Fragments allocated to a logical block must be contiguous on the disk. A file system experiencing free space fragmentation may have difficulty locating enough contiguous fragments for a logical block’s allocation, even though the total number of free fragments may exceed the logical block’s requirements. The JFS alleviates free space fragmentation by providing the **defragfs** utility 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 AIX. 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 7-17.

Data Compression Implementation

Attention: The root file system (/) must not be compressed.

Attention: 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 below 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 meta–data 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.

Once compressed, 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

Since 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 will be 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 may 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 may receive an ENOSPC or EDQUOT error. For example, an old file may be mapped using the **mmap** system call, and a store operation into a previously written location may result in an ENOSPC error. The **ftruncate** system call may 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 would report 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

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

The **–q** option of the **lsfs** command displays the current value for compression.

Compatibility and Migration

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

Backup/Restore

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

Compression Algorithm

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 string's first occurrence 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). Once a given amount of data is compressed, say *N* bytes, then 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

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

- It may require a great deal of time to compress and decompress data so that the usability of a compressed file system may 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

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’s 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, since a file or directory’s 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 would be allocated a single 4096–byte disk block if the logical block size is 4096 bytes. A read or write operation would therefore only have to perform a single disk I/O operation to access the data on the disk. If the logical block size were smaller requiring more than one allocation for the same amount of data, then more than one disk I/O operation may 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, since 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’s worth of disk space is allocated to a partial logical block of a file or directory. Since partial logical blocks always contain less than a logical block’s worth of data, a partial logical block will only consume a portion of the disk space allocated to it. The remaining portion remains unused since 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. A file system using 4096–byte allocation units may experience up to 45% wasted disk space. (Statistic taken from *UNIX System Manager’s Manual*, Computer Systems Research Group, University of California at Berkeley, The Regents of the University of California and/or Bell Telephone Laboratories, 1988, SMM 14.)

Optimizing Disk Utilization

In the JFS, however, 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 will be 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. For consistency with previous verions of AIX Version 3, the default fragment size is 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:

- Disk space allocations of 4096 bytes of fragments are maintained for a file or directory's logical blocks where possible.
- 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."

As the files and directories within a file system grow beyond 32KB 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 32KB 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 8MB. In AIX Version 4.1, agsize is fixed at 8MB. The allowable NBPI values are 512, 1024, 2048, 4096, 8192, and 16,384 with an agize of 8MB.

In AIX Version 4.2 or later, a larger agsize may 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 16MB, the range of NBPI values also double: 1024, 2048, 4096, 8193, 16384, and 32768.

For consistency with previous versions of AIX Version 3, the default NBPI value is 4096, and the default agsize is 8. The desired NBPI and agsize values are specified during file system creation. If the file system size is increased, the NBPI and agsize values remain set to the values specified during the file system's creation.

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 should be 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, the **statfs** subroutine can be used to identify the file system fragment size.

Compatibility and Migration

Previous versions of AIX are compatible with the current JFS, although file systems with a nondefault fragment size, NBPI value, or allocation group size may 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 AIX. 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 AIX 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 may be incompatible with previous versions of AIX. Specifically, only file system images less than or equal to 2G in size and created with the default parameters can be interchanged amongst AIX Versions 3.2, 4.1 and 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 8M can ge interchanges amongst AIX Version 4.1 and AIX Version 4.2. Finally, creating a file system with NBPI value greater than 16384 or with an agsize greater than 8M will result in a JFS file system that is only recognized on AIX Version 4.2.

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

1. Backup 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/Restore

Although backup and restore sequences can be performed between file systems with different fragment sizes and NBPI values, due to increased disk utilization and a large number of i–nodes, restore operations may fail due to a lack of free fragments or disk i–nodes if the fragment size or NBPI value of the source file system is smaller than the fragment size or NBPI value of the target file system. This is of particular interest for full file system backup and restore sequences and may 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 as 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.

Note: Any valid NBPI value can be specified for any device.

Performance Costs

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

Increased Allocation Activity

Since disk space is allocated in smaller units for a file system with a fragment size other than 4096 bytes, allocation activity may 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 would occur 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 files are extended by 4096 bytes at a time when possible.

Free Space Fragmentation

Using fragments smaller than 4096 bytes may cause greater fragmentation of the disk's free space. 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) will be allocated for these free fragments, since the fragments in a single allocation must be contiguous.

Since the fragments allocated for a file or directory's logical blocks must be contiguous, free space fragmentation may 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 16KB of free disk space, the write operation will fail, since 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. The execution of **defrags** has an impact on performance.

Increased Fragment Allocation Map Size

More virtual memory and file system disk space may 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's 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
- NO TAGFile system fragment addressability
- Journaled file system log size
- Maximum journaled file system size

Number of Inodes

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 nbpi (number of bytes per i–node) 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. The values 32768, 65536, and 131072 only apply to AIX Version 4.2 or later.

One i–node is created for every NBPI bytes of allocation group space allocated to the filesystem. 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

AIX Version 4.2 or later supports various allocation group sizes. 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 npbi 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 4MB 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 4MB, and the default log size is one partition, therefore, the root volume group normally contains a 4MB JFS log. When file systems exceed 2GB or when the total amount of file system space using a single log exceeds 2GB, the default log size may not be sufficient. In either case, the log sizes should be scaled upward as the file system size increases. The JFS log is limited to a maximum size of 256MB.

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 8GB ($512 * 2^{24} = 8GB$). When creating a JFS file system, the factors listed above (nbpi, fragment size, and allocation group size) need to be weighed carefully. The filesystem size limitation is the minimum of $NPBI * 2^{24}$ 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 Version 4.2 or later.

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

In file systems enabled for large files, file data stored before the 4MB file offset is allocated in 4096 byte blocks. File data stored beyond the 4MB 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 4KB disk blocks and 1024 128KB disk blocks. In a regular file system, the 132 MB file would require 33 single indirect blocks (each filled with 1024 4KB disk addresses). However, the large file geometry requires only 2 single indirect blocks for the 132 MB file.

Sparse File Allocation

Files that do not have disk blocks allocated for each logical block are called sparse files. Sparse files are created by seeking two different file offsets and writing data. If the file offsets are greater than 4MB, then a large disk block of 128KB is allocated. Applications using sparse files larger than 4MB may require more disk blocks in a file system enabled for large files than in a regular file system.

Free Space Fragmentation

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.

Disk Image Compatibility

File systems enabled for large files cannot be migrated to releases of Versions 3 or 4.1 of AIX. These releases of AIX will not recognize the superblock version number and may generate a message such as: `unknown file system type`. File systems enabled for large files are not the only file systems subject to this behavior. See ”Understanding Fragments and a Variable Number of I–Nodes”, on page 7-17 for more information about file system migration.

> **Note:** File systems from Versions 3 and 4.1 of AIX can be migrated forward.

Zeroing kproc for Large Allocations

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.

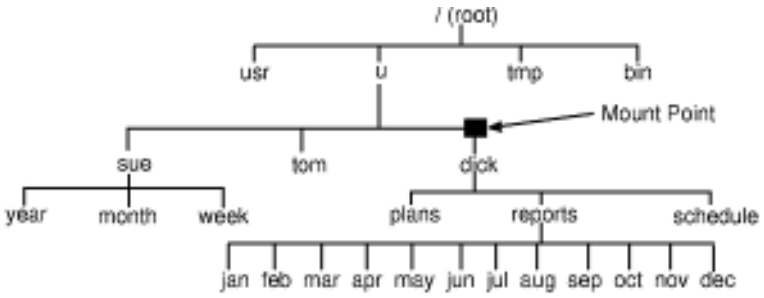
Mounting Overview

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.

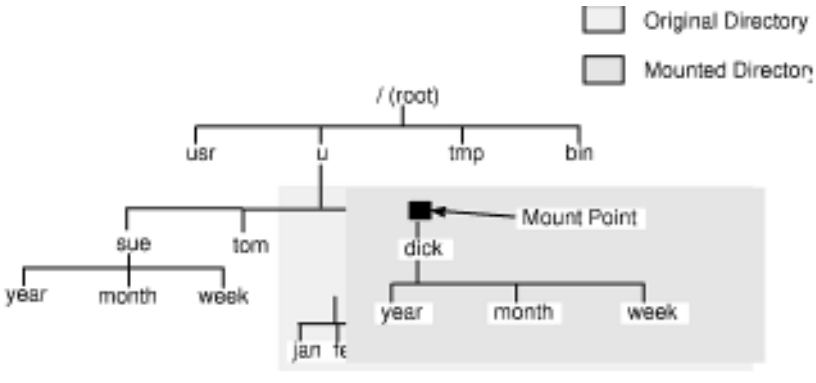
Understanding Mount Points

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. The File Tree View before Mounting figure shows a file system mount point.



File Tree View before Mounting

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. The File Tree View after Mounting figure shows the mounting of a file system.



File Tree View after Mounting

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's 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 will occur. If a mount is unsuccessful, any of the following mounts defined in the /**etc**/**filesystems** file will continue to mount. For example, if the mount of the /**home** file system is unsuccessful, the mount for the /**usr** file system will continue 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 may be able to access the server's physical devices using the client's device special files.

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 execute **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 may not match between the server and client, so users on the server may be able to run programs with capabilities that were not intended for them.

The problem is that **setuid** and **setgid** programs and device–special files should generally 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 should be 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 should also be 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 a diskless workstation's file systems are mounted from a server's /**exports** directory, from the diskless machine's perspective, 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 7-12.

Securing Diskless Mounts

In general, users on a server should 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 may want to mount this directory with the following options of the **mount** command:

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

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 should be exported with read–only permissions and must provide root access. However, you may want to mount this directory with the following options of the **mount** command:

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

Exporting the /export/share Directory

The /**export**/**share** directory should be 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’s /**home** directory:

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

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

- You can mount the /**home** directory on the server over the /**home** directory of the client. In this case, the /**home** directory should be exported with read/write permissions and without root access. To ensure the system’s 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 should both be 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 should 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 should 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 Performance Tuning Guide*.

Topics covered are:

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 8-6
- Understanding Paging Space Allocation Policies, on page 8-3

The following procedures in *AIX 4.3 System Management Guide: Operating System and Devices* 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 8-7 explains how virtual memory is related to paging space.

The default paging space size is determined during the system customization phase of AIX installation according to the following standards:

- Paging space can use no less than 16MB except for hd6 which can use no less than 32MB in AIX Version 4.2.1 and later
- Paging space can use no more than 20% of total disk space
- If real memory is less than 32MB, paging space is two times real memory
- If real memory is greater than or equal to 32MB, paging space is real memory plus 16MB

There is another type of paging space 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 may be lost, and if paging space runs out, the system may panic. When a paging–space low condition is detected, additional paging space should be defined.

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 may 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 may 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, will run 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 executed in the early allocation environment mode will not be sent the **SIGKILL** signal should 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, proper 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 may 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, since 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 will almost always be 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 250MB 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. Since 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.
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.

Active paging spaces cannot be removed. To remove an active paging space, it must first be made inactive. To accomplish this, use the **chps** command so the paging space is not used on the next system restart. Then, after restarting the system, the paging space is inactive and can 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

Once your system is in use, your next consideration should be 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 should always be 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, 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 will back 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. Can usually read data archived on another platform provided it is in cpio format.
dd	Converts and copies a file. Commonly used to convert and copy data to and from non–AIX 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 should help:

- 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 would enable 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 may 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, you should not reuse all of your backup media. Sometimes it may be months before you or some other user of your system notices that an important file is damaged or missing. You should 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 may 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.

  Your system should not be in use 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: The backup of named pipes (FIFO special files) works fine whether they are closed or open. However, the restoration fails when the backup is done on open named pipes. When restoring a FIFO special file, its inode is all that is required to recreate it since it contains all its characteristic information. The content of the named pipe is not relevant for restored. Therefore, the file size during backup should be zero (all the FIFOs closed) before the backup is made.

  Attention: System backup and restore 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. The backup and restore procedures

described herein were tested for operation on the IBM ESCALA platform and may not operate as described on other platforms.

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

Once data has been properly 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 properly. 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 archives. Used only for directories.

Developing a Backup Strategy

There are two methods of backing up large amounts of data:

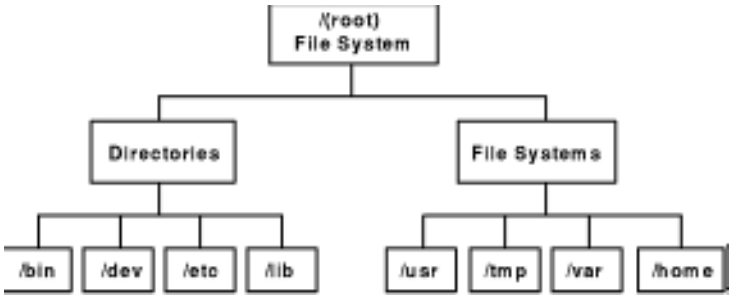
- Complete system backup
- Incremental backup

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. Once you have decided on a strategy for data placement, you can develop a backup strategy for that data. See ”Implementing Scheduled Backups” in *AIX 4.3 System Management Guide: Operating System and Devices* 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. Once 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 /**root** File System Tree illustration.



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 booted, 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 should always 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 should be kept in the /**home** file system or in file systems that are created specifically for user data.

User programs and text should not 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. These are the 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 will not be able to unmount them. It is worth scheduling the backups at low usage times so they can be unmounted; if the user data file systems remain mounted, they will be 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 should be /, /**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
```

Once you are certain that all user file systems are unmounted, refer to ”Backing Up Your System” 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 would make the SMIT dialog convoluted and confusing. SMIT may 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 properly for files that have UID (user ID) or GID (group ID) 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 properly 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's own 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. Bootable tapes cannot be made on or used to boot 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 in the *AIX 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 may 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 the Web-based System Manager fast path, **wsm devices**,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” in the *AIX 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).

Refer to the section on customizing the BOS install program ”Customizing the BOS Install Program” in the *AIX 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 in the *AIX 4.3 System Management Guide: Operating System and Devices* 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 will be 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 will be 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, refer to the **image.data** file in *AIX 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 may 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 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

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 executes 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 in *AIX 4.3 System Management Guide: Operating System and Devices*.
- Changing the Message of the Day *AIX 4.3 System Management Guide: Operating System and Devices*.
- List of Time Data Manipulation Services, on page 10-3

/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 execution.

reltimerid
: Releases a previously allocated interval timer.

Understanding AIX Support for the X/Open UNIX95 Specification

The operating system is designed to support the X/Open UNIX95 Specification for portability of UNIX–based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification and AIX has become even more open and portable for applications.

At the same time, compatibility with previous AIX releases is preserved. This is accomplished by the creation of a new environment variable, which can be used to set the system environment on a per–system, per–user, or per–process basis.

The default AIX environment is one that maintains compatibility with previous AIX releases. To obtain an environment designed to conform to the UNIX95 specification, the environment variable **XPG_SUS_ENV** must be assigned the value **ON**. If **XPG_SUS_ENV** is set to any other value, or is unset, the default AIX behavior will be used.

When **XPG_SUS_ENV** is set, every program in that environment will operate in the UNIX95–specified operating system environment. It is possible that some applications compiled for the AIX environment (perhaps for an earlier version of AIX) will not operate correctly when **XPG_SUS_ENV** is set.

Enabling Dynamic Processor Deallocation

Starting with machine type 7044 model 270, the hardware of all systems with more than two processors will be 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 may 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.

AIX, 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 to AIX. In all cases, AIX logs the error in the system error log. In addition, on multiprocessor systems, depending on the type of failure, AIX 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 AIX 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, AIX 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 AIX silently terminated their bound threads or forcefully moved them to another CPU when one of the processors needs to be deallocated. 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 AIX logs the fact that the deallocation has been aborted 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, 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. AIX logs the firmware error report in the system error log, and, when executing on a machine supporting processor deallocation, start the deallocation process.
3. AIX notifies non–kernel processes and threads bound to the last logical CPU.
4. AIX waits for all the bound threads to move away from the last logical CPU. If threads remain bound, AIX eventually times out (after ten minutes)and aborts the deallocation
5. Otherwise, AIX invokes the previously registered High Availability Event Handlers (HAEHs). An HAEH may return an error that will abort the deallocation.
6. Otherwise, AIX goes on with the deallocation process and ultimately stops the failing processor.

In case of failure at any point of the deallocation, AIX logs the failure 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, in this version of AIX, 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, AIX still reports the errors in the error log and you will see the error indicating that AIX 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, for example, 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 by AIX. The **state** attribute of a **proc** object indicates if the processor is used by AIX and, if not, the reason. This attribute can have three values:

enable

The processor is used by AIX.

disable

The processor has been dynamically deallocated by AIX.

faulty

The processor was declared defective by the firmware at boot time.

In the case of CPU errors, if a processor for which the firmware reports a predictive failure is successfully deallocated by AIX, its state goes from enable to disable. Independently of AIX, this processor is also flagged as defective in the firmware. Upon reboot, it will not be available to AIX 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) would the **proc** object change to Defined.

Examples:

Processor **proc4** is working correctly and used by AIX:

```
# 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 AIX:

```
 # 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 boot, processor **proc4** is reported by firmware as defective and not available to AIX:

```
# 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

Following are examples wth descriptions of error logentries:

errpt short format – summary

Three different error log messages are associated with CPU deallocation. 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

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    00000000
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 by AIX 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 by AIX 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 would cause the machine to have less than two available CPUs. AIX 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 would normally 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

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
- Conversion between code sets

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; it 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”, on page 11-4.

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” in *AIX 4.3 System Management Guide: Operating System and Devices*.

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(TM) 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 11-16.

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. 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.ISO8859-1`. 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
be_BY	Byelorussian	Belarus	ISO8859–5
bg_BG	Bulgarian	Bulgaria	IS08859–5
ca_ES	Catalan	Spain	IS08859–15
cs_CZ	Czech	Czech Republic	IS08859–2
Da_DK	Danish	Denmark	IBM–850
da_DK	Danish	Denmark	ISO8859–1
da_DK	Danish	Denmark	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
el_GR	Greek	Greece	ISO8859–7
en_AU	English	Australia	ISO8859–15

Locale	Language	Country	Code Set
en_BE	English	Belgium	ISO8859–15
En_GB	English	Great Britain	IBM–850
en_GB	English	Great Britain	ISO8859–1
en_GB	English	Great Britain	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_ES	Spanish	Spain	IBM–850
es_ES	Spanish	Spain	ISO8859–1
es_ES	Spanish	Spain	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
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_CH	French	Switzerland	IBM–850
fr_CH	French	Switzerland	ISO8859–1
fr_CH	French	Switzerland	ISO8859–15
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

Locale	Language	Country	Code Set
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	ISO–8859–5
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
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
sl_Sl	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
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

Locale	Language	Country	Code Set
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 System Management Interface Tool (SMIT) Manage Language Environment menu.

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; they should not be confused 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 will use unless the **LC_*** environment variables are set). The **LANG** environment variable can be changed by using the System Management Interface Tool (SMIT). The C and POSIX locales 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 should be used for the C and POSIX locales and that **NLSPATH** will be 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. Refer to 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 will be 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 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 are free to 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 Manage Language Environment SMIT interface 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 has to 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 has to 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 has to 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 may encounter on the system. This section lists common scenarios with suggested actions to be taken.

- User keeps default code set

  The user may be satisfied with the default code set for language–territory combinations even where more than one code set is supported. The user may 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 will be 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 may 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 may be converted through the Web-based System Manager Users application, the Manage the Language Environment SMIT 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 Manage Language Environment SMIT 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 may need to convert text files or message catalogs. There are several standard converters for this purpose.

When a program sends data to another program residing on a remote host, the data can require conversion from the code set of the source machine to that of the receiver. For example, when communicating with an IBM VM system, the system converts its ISO8859–1 data to EBCDIC. Code sets define character and control function assignments to code points. These coded characters must be converted when a program receives data in one code set but displays it in another code set.

There are two interfaces for doing conversions:

- iconv command
  Allows you to request a specific conversion by naming the from and to code sets.
- libiconv functions
  Allow applications to request converters by name.

The system provides a library of converters that is ready to use. You supply the name of the converter you want to use. The converter libraries are found in the following directories: **/usr/lib/nls/loc/iconv/*** and **/usr/lib/nls/loc/iconvTable/***.

In addition to code set converters, the converter library also provides a set of network interchange converters. In a network environment, the code sets of the communications systems and the protocols of communication determine how the data should be converted.

Interchange converters are used to convert data sent from one system to another. Conversions done from one internal code set to another require code set converters. Whether data must be converted from a sender's code set to a receiver's code set, or 8–bit data must be converted into 7–bit data form, a uniform interface is required. The **iconv** subroutines provide this interface.

Standard Converters

There are standard converters for use with the **iconv** command and subroutines. The following list describes the different types of converters.

For a list of converters, see *AIX General Programming Concepts: Writing and Debugging Programs*.

Converter Type

- Table converter
  Converts single–byte stateless code sets. Performs a table translation from one byte to another byte.
- Multibyte converter
  Provides conversions between multibyte code sets; for example, between Japanese PC Code (IBM–943 and IBM–932) or Japanese AIX Code (IBM–eucJP) and IBM Japanese Host Code Sets (IBM–930 and IBM–939).

Interchange Converter Types

- 7–bit converter
  Converts between internal code sets and standard interchange formats (7–bit).
- 8–bit converter
  Converts between internal code sets and standard interchange formats (8–bit).
- Compound text converter
  Converts between compound text and internal code sets.

- uucode converter
  Provides the same mapping as the **uuencode** and **uudecode** commands.
- Miscellaneous converters
  Used by some of the converters listed above.

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 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 don't 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 AIX locales 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 4.3 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 Process Management in the *AIX 4.3 System Management Concepts: Operating System and Devices*.

AIX contains tools to:

- Observe the creation, cancellation, identity, and resource consumption of processes.
  - **ps** is used to report process IDs, users, CPU–time consumption, and other attributes.
  - **who –u** reports the shell process ID of logged–on users.
  - **svmon** is used to report process real–memory consumption. (See *Performance Toolbox 1.2 and 2.1 for AIX: User's Guide* for information on the **svmon** command.)
  - **acct** 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 15-2.)
- Control the priority level at which a process contends for the CPU.
  - **nice** causes a command to be run with a specified process priority. (See *AIX 4.3 System User's Guide: Operating System and Devices.)*
  - **renice** changes the priority of a given process.
- Terminate processes that are out of control.
  - **kill** sends a termination signal to one or more processes.
- Tune the operating system's process–management mechanisms.
  - **schedtune** permits changes to the process scheduler parameters. (See *AIX Performance Tuning Guide* for information on the **schedtune** command.)

Chapter 13. Workload Management

AIX Workload Management (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).

WLM gives you the ability to create different classes of service for jobs, and specify attributes for those classes. These attributes specify minimum and maximum amounts of CPU and physical memory resources to be allocated to a class. You can then classify jobs automatically to classes using class assignment rules. These rules are based on the name of the user or group of the process or the pathname of the applications.

Managing Resources with WLM

WLM monitors and regulates the CPU utilization and physical memory consumption of the threads and processes active on the system. You can set minimum or maximum limits by class for each resource managed by WLM. In addition, a target value for each resource per class may be given. This target is representative of the amount of the resource that would be optimal for the jobs in the class. The processes are automatically assigned to a class by WLM upon execution, using a set of class assignment rules provided by the system administrator.

This new version of WLM allows WLM to start in **active** mode, where WLM does monitoring and regulation of CPU and memory (the *normal* operating mode), or in **passive** mode, where WLM only classifies processes and monitors resource utilization without interfering with the standard &Symbol.AIX; resource allocation algorithms. This mode is primarily interesting when setting up new WLM configuration:

- To test the classification and assignment rules by checking (using **ps**) that the various applications correctly end up in the class they were supposed to be in without changing the behavior of the system.
- To obtain a base line of the application's resource utilization "without WLM," using the **wlmstat** command. This should provide a reference for system administrators and help them decide how to apply resource shares and limits (when needed) to favor critical applications and/or restrict the resource usage of less important work, in order to meet their business goals.

Minimum and Maximum Resource Limits

The different resources can be limited by the 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, even if there is no contention for the resource. 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 maximum range, separated by a hyphen (–) with whitespace ignored. Each limit value is followed by a percent sign (%).

WLM does not place hard constraints on the values of the resource limits. The following are the only constraints:

- The minimum range must be less than or equal to the maximum range.
- The sum of the minimum of all classes within a tier cannot exceed 100.

WLM enforces the maximum range to ensure that a class or process within a class is not given more resource than the specified value. Note that in the case of a memory constraint, swapping performance can become very poor for processes within the constrained class. Memory minimums for other classes should be used before imposing a memory maximum for any class.

A minimum value constraint on a class means that processes within the class are always allowed to get resources up to the minimum. WLM cannot guarantee that processes actually reach their minimum limit. This depends on how the processes use their resources and on other limits that may be in effect. For example, a class may not be able to reach its minimum CPU entitlement because it cannot get enough memory.

Target Shares

The target for use of different types of resources 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 3 classes, A, B and C defined, whose targets are respectively 50, 30 and 20.

- If all 3 classes are active, the total number of shares for the active classes is 100. Their targets, expressed as percentages is 50%, 30% and 20% .
- 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%.

In this example, the sum of the shares for the 3 classes was 100 simply to make the sample calculation easier. A target can be any number between 1 and 65535.

The target represents a percentage of resources, which can vary widely, depending on how many classes are active at any given time. However, WLM makes sure that the dynamic value of the target 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.

Tier Value

The tier value for a class is the importance of the class relative to other classes. The tier value 0 is the most important; the value 9 is the least important.

Examples of 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

This example examines CPU allocation, with the assumption that each class can consume all the CPU 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 may not want batch jobs to eat up 50% of the CPU when other jobs, possibly of higher priority, are also running. In a situation like the above example, C is placed in a lower priority tier. C then receives whatever CPU is left over by A and B. In the above example, C would receive no CPU, since A and B were each capable of absorbing 100% of the CPU. In most situations, however, A and B, in a high priority tier could be 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

This 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

WLM offers a fine level of control over resource allocation. However, it is easy to set up conflicting values for the various parameters and obtain undesirable system behaviors. The following tips can help you avoid creating conflicts:

- Know your user base and their basic computing needs when defining classes and class assignment rules.
- Know the resource needs of the main applications. A way to determine what their resource usage could be to start WLM in passive mode once you have defined your classes and the associated classification rules. **wlmstat** will then give you snapshots of the resource utilization, per class, when the resource allocation is not regulated by WLM.
- Prefer targets over minimum and maximum limits. Targets give the system greater flexibility than hard limits, and targets can help prevent starving applications.
- Try to balance the load using only targets and monitor the system with the **wlmstat** command. Apply minimum limits for classes that do not receive sufficient share.
- Prioritize some jobs by using tiers.
- Use maximum only as a last resort to restrain applications that consume large quantities of share. Maximum can also be used to place hard limits on users' resource consumption (for example, for accounting purposes).

Specifying WLM Properties

You can specify the properties for the WLM subsystem by using either a Web-based System Manager graphical user interface, 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.

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 Subsystems in *AIX 4.3 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: Refer to 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

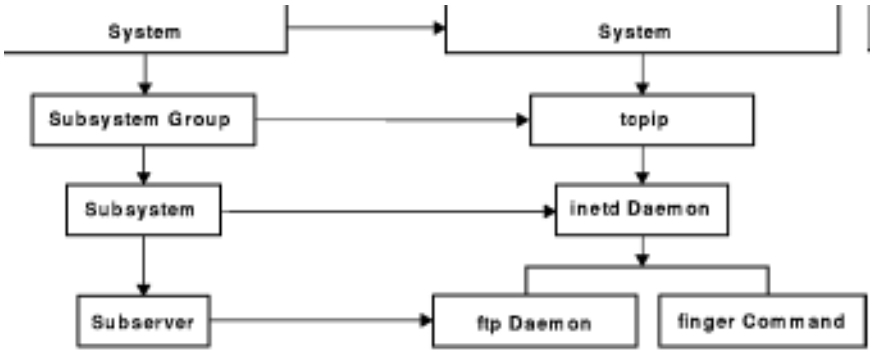
A *subserver* is a program or process that belongs to a subsystem. A subsystem can have multiple subservers and is responsible for starting, stopping, and providing status of subservers. Subservers can be defined only for a subsystem with a communication type of

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 has a hierarchical structure (see figure). The 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).



SRC Hierarchical Structure

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.

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's 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

For data to be collected automatically, a member of the adm group needs to follow the procedures described in "Setting Up an Accounting System" in *AIX 4.3 System Management Guide: Operating System and Devices*. These procedures enable the **cron** daemon to run the commands that generate data on:

- Amount of time each user spends logged in to the system
- Usage of the processing unit, memory, and I/O resources
- Amount of disk space occupied by each user's files
- Usage of printers and plotters
- 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 15-6.

Collecting Accounting Data

There are several types of accounting data: connect–time data, process data, disk–usage data, printer–usage data, and fee data. Each is described in the following paragraphs.

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 15-4.

Process Accounting

The system collects data on resource usage for each process as it runs. These data include:

- User and group numbers under which the process runs
- First eight characters of the name of the command
- Elapsed time and processor time used by the process
- Memory use
- Number of characters transferred
- 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 15-4.

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 spreads 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 15-3.

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 15-2.

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 15-2.

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*

OR

*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 1.345x109, or 1,345,000,000. And the scientific notation 1.345e–9 is equal to 1.345x10–9 or, .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, 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 Berkeley Software Distribution (BSD) systems.

lastcomm

Displays information about the last commands that were executed. This command is provided for compatibility with Berkeley Software Distribution (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 Berkeley Software Distribution (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.

time

Prints real time, user time, and system time required to execute a command.

timex

Reports in seconds the elapsed time, user time, and execution time.

sar

Writes to standard output the contents of selected cumulative activity counters in the operating system. The **sar** command reports only on local activities.

Accounting Files

There are two main accounting directories: 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 execution 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*	Previous day's **wtmp** file.

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*.
wtmp.*mmdd*	Saved copy of the **wtmp** file for *mmdd*. This file is removed after system startup by the **remove** command.

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 execution 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 AIX 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 AIX commands or SMIT.

For a detailed explanation and procedures for using Web-based System Manager, see Using Web-based System Manager in *AIX 4.3 Quick Beginnings*. For procedures for installing and configuring Web-based System Manager, see Setting Up and Running Web-based System Manager in *AIX 4.3 System Management Guide: Operating System and Devices*.

Chapter 17. System Management Interface Tool

This chapter provides a brief overview of the System Management Interface Tool (SMIT) panels, on page 17-2

System Management Interface Tool (SMIT) Overview

Although Web-based System Manager is the primary interface for AIX 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 CDE Desktop

With the CDE Desktop (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.

> **Note:** The AIX Common Desktop Environment (CDE) 1.0. Help volumes, web–based documentation, and hardcopy manuals may refer to the desktop as CDE Desktop, the AIXwindows desktop, the CDE desktop, AIX CDE 1.0, or simply, the desktop.

For additional information and procedures for using CDE, see Managing the CDE Desktop in *AIX 4.3 System Management Guide: Operating System and Devices*.

Chapter 19. Documentation Library Service

The Documentation Library Service allows you to read and search 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.

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 your own company's HTML documents 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 library's search engine, the library's 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.

It is highly recommended that the library service be fully configured since 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, you should still configure the documentation library service since it is expected that other applications may 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 Installing AIX Documentation in *AIX 4.3 Installation Guide*.

For additional information, see the following topics in the *AIX 4.3 System Management Guide: Operating System and Devices*:

- Changing the Configuration
- Working with Documents and Indexes
- Advanced Topics
- Problem Determination

Chapter 20. Power Management

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 that you can perform and their procedures, see Using Power Management in *AIX 4.3 System Management Guide: Operating System and Devices*.

The following section, Power Management Limitation Warnings, contains important information for all users of Power Management.

Power Management Limitation Warnings

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 boot.

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 will enable 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 boot 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 may have to be re–established by the user after resuming. Since locally cached data won't be available to other nodes on the network during this time and network activity cannot be monitored by the local node during this time, it is recommended that the suspend and hibernation states not be used 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 will enable the suspend and hibernation states effective on the next system boot, 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 may not be able to run. In such a situation, or whenever necessary, it should always be possible to turn off the power immediately by pressing the power button three times quickly (within a two second period). This will override whatever state transition has been selected for the power switch, and will require a full reboot.

In addition, if the Power Management daemon (/usr/bin/pmd) is never started (by an entry in /etc/inittab by default), the power switch will act as if there was no power management. A single button press will turn off the system. If /usr/bin/pmd is started and then killed, the first two button presses will be ignored, and the third will turn 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
- PCI Hot Plug Management

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 is the bus and adapters that have a dependency on the higher–level system device. The bottom of the hierarchy are devices where no other devices are connected. These devices have dependencies on all devices above it in the hierarchy.

Parent–child dependencies are used at boot time 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 will not be 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 adapter's eight ports.

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 boot 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.

A drawer's location code consists of only the drawer field and is simply a two–character code. An adapter's location code consists of the drawer and slot fields and has the format AA–BB, where AA corresponds to the drawer's 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 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 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/Plotter Location Codes

Location codes of 00-00-S1-00 or 00-00-S2-00 indicate the printer or plotter 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 the printer or plotter is connected to an adapter card other than the standard I/O board. For these printers and plotters, 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 or plotter 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 will indicate the tty device which 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

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 card's SCSI bus that the device is attached to. For a card that provides only a single SCSI bus, this field will be set to 00. Otherwise, a value of 00 indicates a device attached to the card's internal SCSI bus and a value of 01 indicates a device attached to the card's external SCSI bus.
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's diskette ports 0 or 1.

Dials/LPFKeys Location Codes

For a graphics input adapter attached Dials/LPFKeys device, 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 will be either 01 or 02, depending on whether the device is attached 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 4.3 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 Commands Reference, Volume 3*. This command displays a list of all the hot plug slots and their characteristics.
- The drslot command, in the *AIX 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 rebooting 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 reboot the operating system. Some reasons for adding an adapter could 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.
- nstalling a new system for which adapter cards become available after the initial configuration of optional hardware subsystems, including PCI adapters, and the installation and boot 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 4.3 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 could 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 4.3 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 could 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 4.3 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 reboot 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 plugged. 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 Version 4.3 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 (enX) or IEEE 802.3 (etX). 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:

Basic tasks for Tape Drives are listed in Tape Drives in *AIX 4.3 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 (for **chdev**, the **mode** attribute) to the **yes** value 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 **no** value, 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 **no** value. The default value is **yes**.

With the **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**, 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 8mm tape drives use 2.2MB of tape and can take up to 8.5 seconds to write. Regular file marks use 184K and take approximately 1.5 seconds to write.

When you use an 8mm tape in append mode, use extended file marks for better positioning after reverse operations at file marks. This reduces errors.

Retension

Setting the Retension attribute (for **chdev**, the **ret** attribute) to **yes** instructs the tape drive to retension 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 **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 **chdev**, 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 22-14 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 drive’s high density

setting. Specific permitted values and their meanings vary with different types of tape drives. These attributes do not affect the tape drive's ability 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 **chdev**, the **res_support** attribute), specifying the **yes** value 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 reserve/release commands. Some SCSI tape drives have a predefined value for this attribute so that reserve/release commands are always supported.

Variable Length Block Size

The Variable Length Block Size attribute (for **chdev**, 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. Refer to the specific tape drive SCSI specification to determine whether this is required.

Data Compression

Setting the Data Compression attribute (for **chdev**, the **compress** attribute) to **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 **chdev**, the **autoload** attribute) to **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, will cause 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 2.0GB 4mm 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 2.0GB 4mm 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. The density settings are predefined because the tape drive always writes in 2.0GB mode.

Attributes for 4.0GB 4mm 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 2.0GB mode only.
DDS2	Read in either density; write in 4.0GB 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 4.0GB 4mm 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 2.3GB 8mm 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 2.3GB 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 2.3GB mode.

Attributes for 5.0GB 8mm Tape Drives (Type 8mm5gb)

Block Size

The default value is **1024**. If a tape is being written in 2.3GB 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	5GB mode (compression capable)
21	5GB mode noncompressed tape
20	2.3GB mode
0	Default (5.0GB 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 5GB 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 5.0GB 8mm tape drive, the Retension, Reserve Support, and Variable Length Block Size attributes have predefined values which cannot be changed.

Attributes for 20000MB 8mm 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 20.0GB 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	20GB mode (compression capable)
0	Default (20.0GB 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 20.0GB 8mm tape drive, the Retension, Reserve Support, and Variable Length Block Size attributes have predefined values which cannot be changed.

Attributes for 35GB Tape Drives (Type 35gb)

Block Size

The IBM 7205 Model 311 throughput is sensitive to blocksize. The minimum recommended blocksize for this drive is 32K Bytes. Any block size less than 32K Bytes restricts the data rate (backup/restore time). The following table lists recommended block sizes by AIX command:

AIX Command Supported	Default Block Size (Bytes)	RECOMMENDATION
BACKUP	32K or 51.2K (default)	Uses either 32K or 51.2 K depending on if ”Backup” is by name or not. No change is required.
TAR	10K	There is an error in the manual that states a 512K 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 2.6GB format (density) and 6.0GB format (density) are significantly impacted when you use smaller than the recommended blocksizes. As an example: using a blocksize of 1024 bytes to backup 32GB of data takes approximately 22 hours. Backing up the same 32GB of data using a blocksize of 32K 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	2.6GB	2.6GB (No Compression)	23	17h
	6.0GB	6.0GB (No Compression)	24	18h
	10.0GB	20.0GB (Default for drive)	25	19h
DLTtapeIIIxt	15.0GB	30.6GB (Default for drive)	25	19h
DLTtapeIV	20.0GB	40.0GB	26	1Ah
	35.0GB	70.0GB (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 150MB 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 150MB 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 525MB 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 525MB 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 1200MB 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 1200MB 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 12000MB 4mm Tape Drives (Self Configuring)

Block Size

The IBM 12000MB 4mm Tape Drive's throughput is sensitive to blocksize. The minimum recommended blocksize for this drive is 32K Bytes. Any block size less than 32K Bytes restricts the data rate (backup/restore time). The following table lists recommended block sizes by AIX command:

AIX Command Supported	Default Block Size (Bytes)	RECOMMENDATION
BACKUP	32K or 51.2K (default)	Will use either 32K or 51.2 K depending on if "Backup" is by name or not. No change is required.
TAR	10K	There is an error in the manual that states a 512K byte block size. Set the Blocking Parameter to **–N64**.
MKSYSB	See BACKUP	MKSYSB uses the BACKUP Command. No change is required.

AIX Command Supported	Default Block Size (Bytes)	RECOMMENDATION
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 12000MB 4mm 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	2.0GB	4.0GB	19	13h
DDS2	4.0GB	8.0GB	36	24h
DDS3	12.0GB	24.0GB	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 13000MB 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 13000MB 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 properly, 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**. 8mm and 4mm tape drives usually use **1024** and waste space on the tape if the block size attribute is left at **512**. **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 may be set to **yes**, if the drive supports reserve/release commands. If you are unsure, **no** is a safer value.

Variable Length Block Size

0 is the default value. Nonzero values are used primarily on quarter inch cartridge (QIC) drives. Refer to 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 Extended File Marks, Retension, and Data Compression attributes 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**, ... **/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's Density Setting #1 or with the tape drive's 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. AIX for BSD System Managers

AIX 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:

AIX for BSD System Managers Overview

The following articles describe general information for 4.3 BSD administrators:

The following articles describes in detail specific system administration topics:

Introduction to AIX for BSD System Managers

The following hints will help you get started managing the system:

- You should start by logging in as root at the graphics console.
- You should 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.
- Several AIX facilities are involved in 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 17-2.
  - 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 ”Devices Overview for System Management”, on page 21-1.
  - 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 14-2.

Major Differences between 4.3 BSD and AIX

This article summarizes the major differences between AIX and 4.3 BSD systems. For more detailed discussions of these topics, see the list of articles in the ”AIX for BSD System Managers Overview”, on page A-2.

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 AIX configuration data 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.

AIX also supports dbm versions of password and user information. Furthermore, the **/etc/passwd**, **/etc/group**, and **/etc/inittab** files are examples of AIX files where the information is stored in traditional form rather than in stanza form.

Other AIX configuration data 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 may cause the system to fail, and may prevent you from successfully restarting the system. You should 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 an AIX 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 start–up 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 AIX, 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 AIX 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.

AIX commands 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 the 4.3 BSD version and the AIX version of these commands are discussed in ”File Systems”, on page A-19.

4.3 BSD maintains a list of file systems in the **/etc/fstab** file. AIX maintains a stanza for each file system in the **/etc/filesystems** file.

The 4.3 BSD file system usually reads in large blocks of 8KB, but can store several small files in one block using *fragments* usually consisting of 1KB. The AIX file system does not support fragments and each file consumes at least one block. The block size is 4KB.

New Commands

To handle new configuration and disk management systems, AIX has about 150 new commands that are new to 4.3 BSD administrators. For more information, see ”Commands for AIX System Administration”, on page A-11.

Boot and Startup

AIX supports automatic identification and configuration of devices. Consequently, the boot and 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, AIX 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 prior to 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.

AIX 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 AIX.

The AIX line printing 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

AIX 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 may be changed by the administrator.

Notes:

1. AIX 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

Both the AIX accounting files in the **/usr/lib/acct** directory and the system activity reporting tools in the **/usr/lib/sa** directory 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.

AIX 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.

AIX 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 AIX **tar** command is not fully compatible with the 4.3 BSD **tar** command. The AIX **tar** command requires the **–B** option (blocking input) if it is reading from a pipe. The AT&T **cpio** command is compatible with the AIX version.

AIX can read and write in **dump** and **restore** command format. For example, the AIX **backup** command 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 AIX **restore** command with the syntax:

```
restore -mivf Device
```

is the same as the 4.3 BSD **restore** command with the syntax:

```
restore ivf Device
```

AIX also has the 4.3 BSD **rdump** and **rrestore** commands. The only difference between the two versions is that for AIX 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 AIX **backup** command 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

AIX 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 Kernel Extensions and Device Support Programming Concepts*.

For more information, see ”Backup Overview for System Management”, on page 9-2.

Boot and Startup for BSD 4.3 System Managers

On 4.3 BSD systems, the **init** program is the last step in the boot 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.

AIX follows the System V initialization procedure. You can edit the AIX /**etc**/**inittab** file by directly editing the file, using the **telinit** command, or by using the following AIX 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 AIX System Administration for BSD 4.3 System Managers

This list contains commands that are specifically for administering the AIX environment.

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 a device's characteristics.
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 **/etc/environment** file for 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** option. Floating licensing can be enabled or disabled (on or off) through the **–f** option.
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.
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 AIX **cron** daemon 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 an AIX 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 may mean the hardware is faulty or it may mean that the configuration database in the /**etc**/**objrepos** directory is damaged.

The **cfgmgr** command processes the configuration database in the /**etc**/**objrepos** directory 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 AIX

The following table compares file names and functions between the 4.3 BSD, SVR4, and AIX operating systems.

File Comparison Table				
4.3 BSD File	**SVR4 File**	**AIX File**	**Database**	**Type (odm/dbm)**
L–Devices	Devices	Devices	no	
L–dialcodes	Dialcodes	Dialcodes	no	
L.cmds	Permissions	Permissions	no	
L.sys	Systems	Systen	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 AIX documentation.

File Systems for BSD 4.3 System Managers

This information offers a brief comparison of AIX file systems to other systems' file systems, and an outline of the supported file systems types on AIX systems.

The AIX 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.

AIX 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 may 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 AIX

AIX supports disk quotas.

AIX does not allow mounting of diskettes as file systems.

The syntax of the AIX **mount** and **umount** commands 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	**AIX Syntax**	**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 7-2 for more information.

Finding and Examining Files for BSD 4.3 System Managers

AIX supports the following 4.3 BSD file commands:

- **which**
- **whereis**
- **what**
- **file**

AIX 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 may 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 AIX 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.
swapon(1)	Specifies additional devices for paging and swapping.

If a large paging space is required, you should 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 on an AIX system, additional AIX commands and command options, name and address resolution on AIX systems, and differences between 4.3 BSD network management and AIX network management.

How to Change Default Startup to Permit 4.3 BSD ASCII Configuration

You can administer AIX network interfaces 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 will 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 AIX **ifconfig** command 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 AIX **netstat** command 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 AIX:

securetcpip The **securetcpip** shell script enables controlled access mode, which provides enhanced network security. It disallows execution 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 will be allocated to the **mbuf** and cluster **mbuf** pool.
arpt_killc	Specifies the time in minutes before an inactive complete Address Resolution Protocol (ARP) entry will be 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 AIX and 4.3 BSD

On AIX systems, 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 AIX:

- 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

AIX 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 AIX **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, you should 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 the Web-based System Manager fast path, **wsm 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 ESCALA, we suggest you specify the **power** identifier for family (CPU).

Passwords for BSD 4.3 System Managers

The following information details the differences between managing passwords in AIX systems and 4.3 BSD systems.

Setting a User Password

When you use the AIX /**bin**/**passwd** command as the root user, you are prompted for the current root user password. An example of using the AIX /**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 may cause problems. Change the stack size to 65,536 and run **usrck** command again.

You should also run the **grpck** and **usrck** command to verify group and user attributes.

Editing the Password File

In AIX, 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 at once, 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.

AIX does not support the **vipw** command but does support the **mkpasswd** command. However, you can still administer passwords on an AIX 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 for AIX V3. 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 AIX commands.

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's **PATH** environmental variable specifies that the /**usr**/**ucb** directory be searched prior to the /**bin** directory.

Performance Measurement and Tuning for BSD 4.3 System Managers

All devices on AIX 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 may 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 rebooted.

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.

AIX supports the **vmstat** and **iostat** commands, but not the **systat** command or load averages.

Printers for BSD 4.3 System Managers

AIX 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 AIX printer subsystems are different. However, they both use the **lpd** protocol for remote printing. Both systems use /**etc**/**hosts.lpd** if it exists and /**etc**/**host.equiv** otherwise. The AIX printer subsystem offers a gateway to 4.3 BSD printer subsystems, so AIX systems 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 AIX.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 AIX 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 the 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 an AIX 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”.

AIX 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.
prtty(1)	Prints to the printer port of the terminal.
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.

AIX stores terminal port information in the ODM and starts terminals when the **init** program reads the **/etc/inittab** file. In AIX, you should 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 AIX which port should 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, AIX uses **terminfo** entries in **/usr/lib/terminfo/?/*** files. Users with 4.3 BSD Systems may 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.

AIX includes source for many **terminfo** entries. Some of these may need to be compiled with the **tic** command. The **termcap** file is provided in **/lib/libtermcap/termcap.src** file.

Dave Regan has donated his program **untic** to the public domain. This program uncompiles **terminfo** entries so that the source form may be modified and recompiled with **tic**. It is available from sites that archive **comp.sources.unix**.

UUCP for BSD 4.3 System Managers

AIX 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.

AIX 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 4.3 System User's Guide: Communications and Networks*.

Appendix B. Managing the InfoExplorer Program

In previous releases of AIX, documentation was shipped as InfoExplorer databases. With the current version of AIX, documentation is shipped in HTML format for use with a web browser. InfoExplorer databases are **not** shipped with the current version of AIX. You will only be able to browse InfoExplorer databases if you purchased the InfoExplorer feature with AIX. This chapter is usefull only to customers who purchaed InfoExplorer.

Customizing the InfoExplorer Program

The InfoExplorer program can be set up to access the library from either a CD–ROM or a fixed disk. Accessing the library from the fixed disk can improve performance, but requires additional storage space due to the size of the information bases.

To customize the InfoExplorer program for your system, you can set up public notes, on page B-3 to give users information that is specific to your installation or your information needs. You can also set up bookmark lists and history files and transfer them to other users to provide tutorials or similar ordered lists of information.

Understanding the InfoExplorer Information Bases

The InfoExplorer library and code are located in the **/usr/lpp/info** directory. In this directory are different subdirectories that contain executables, information bases, fonts, and public note storage. Following is a listing of these directories:

bin	Contains executables for ASCII and graphics InfoExplorer tools. The **mergenote** command, used to combine groups of note files into a single file, is also located in this directory.
data	Contains the **ispaths** file that describes the installed information bases and some system definition files, and provides storage location for public notes created. See ”Creating InfoExplorer Public Notes”, on page B-9. Also contains definition files used by InfoExplorer executables.
data/JP	Contains definition files for MBCS Japanese language environment.
X11fonts/JP	Contains Japanese fonts used by the InfoExplorer window interface for MBCS Japanese language environment.
notes	Contains system notes if any are installed.
lib/*Language*	Contains installed information bases for the national language specified by the *Language* directory name. The directory name is based on the installed language for the system. For example, on a French Canadian system the directory name would be **fr_CF**. The default is the name **en_US** on U.S. English systems.
lib/Language/*library*	Contains additional library subdirectories within a library directory.

On a system with multiple languages installed, more than one *Language* directory can exist in the **/usr/lpp/info/lib** directory. For example, on a system that uses German and French, the information base for German can be installed in the **/usr/lpp/info/lib/de_DE** directory, while the information base for French is installed in the **/usr/lpp/info/lib/fr_FR** directory. Users can use either language by changing environment variable settings for **LANG**, **INFOLANG**, or **INFOLOCALE**. For more information, see ”Changing InfoExplorer Languages”, on page B-8.

Other files that are not in **/usr/lpp/info** include**:**

/usr/bin/info	Contains the shell script that determines whether to invoke the ASCII or graphical version of InfoExplorer.
/usr/lib/x11/app–defaults/Info_gr	Contains the application defaults file that contains system resource definitions.

Managing InfoExplorer Public Notes

Public notes can be read by any hypertext user. By default, InfoExplorer notes are private, accessible only to the user who created them. These private notes are saved in the user's **$HOME/info** and **$HOME/info/<library>/notes** directories.

Users with write access to the /**usr**/**lpp**/**info**/**data** directory can create public notes, which are stored in this directory, by converting their private notes to public notes with the **mergenote** command.

Accessing InfoExplorer from CD–ROM

The first time you access InfoExplorer databases from your CD–ROM, you must:

- Create a CD–ROM file system.
- Mount the CD–ROM file system.
- Run the **linkinfocd** script.

  Note: You can also install databases from the CD–ROM. Some databases on this CD–ROM may have already been installed with the operating system or other licensed products. Run the **lslpp** command or SMIT to get a list of the database packages already installed on your system.

  The installation application that you use (SMIT or one of the VSM applications) will create a temporary mount point for the CD–ROM.

Prerequisites

1. You must have root user authority or be a member of the system group to create and mount the CD–ROM file system and run the **linkinfocd** script.

Create a CD–ROM File System

1. Set the power switch to On if you are using an external CD–ROM drive.
2. Remove the CD–ROM from the plastic case and place it in the disc caddy or disc tray.
3. Insert the disc caddy containing the Hypertext Information Base Library CD–ROM into the disc caddy opening or on the disc tray.
4. Use the **smit crcdrfs** fast path to create the CD–ROM file system. The Add a CDROM File System menu appears.
5. Press F4 for the DEVICE name field to get a list of the devices available. The DEVICE NAME overlay appears over the previous screen.
6. Specify the available CD–ROM device you plan to use.
7. Highlight the MOUNT POINT field.
8. Type the following, but do not press Enter until you get to step 10.

   ```
   /infocd
   ```

9. Highlight the Mount AUTOMATICALLY at system restart? field. Select one of the following choices:

   a. Mount InfoExplorer every time the system starts, press the Tab key to specify **yes.**

   b. Mount InfoExplorer manually, use the default value **no**.

10. Press Enter after you have completed making changes to the entry fields.
11. Press F10 to exit SMIT.

Mount the CD–ROM File System

Mount your CD–ROM to the file system you created by following the steps below:

1. Enter the SMIT fast path at the system prompt:

   ```
   smit mountfs
   ```

   The Mount a File System menu appears.

2. Highlight the FILE SYSTEM name field.

   Note: The system always mounts the CD–ROM as a read–only file system. You can use the tab key to select **yes** or **no** in the Mount as READ ONLY file system field.

3. Press F4 to list file system names. The FILE SYSTEM name overlay appears over the previous menu.
4. Select a line similar to the following:

   ```
   /dev/cdx /infocd cdrfs
   ```

   where *x* is the number of your CD–ROM drive.
5. Press Enter.
6. Select **Do** to mount the CD–ROM file system.
7. Press F10 to exit SMIT when the Command: status field changes to **OK**.

The InfoExplorer databases are now mounted and ready to be accessed from the CD–ROM.

Notes:

1. If the CD–ROM is ejected from the system while it is still mounted, the connection is broken and you cannot access the information. To remove the CD–ROM from the system, unmount that file system using the **unmount** command before ejecting the CD–ROM. To access the CD–ROM again, you must remount the CD–ROM file system, using the **mount** or **smit** command.
2. You can keep copies of the information bases on your fixed disk, in case your CD–ROM becomes inaccessible, or you can delete them. For more information, see ”Removing InfoExplorer Information Bases”, on page B-7.

Run the linkinfocd Script

The **linkinfocd** script links the database subdirectories from the **/infocd** CD–ROM file system to the **/usr/lpp/info/lib/en_US/aix41** directory. Each database subdirectory is linked individually, just as each database can be installed separately. This allows you to have databases installed on your fixed disk drive, databases linked from a mounted CD–ROM, or a combination of both. The script also links the **ispaths** file from the **/infocd** CD–ROM file system to the **/usr/lpp/info/data** directory.

The **linkinfocd** script does the following:

- Checks to see if InfoExplorer (**/usr/lpp/info**) is installed on the system. If **/usr/lpp/info** does not exist, the script exits.
- Checks to see if the **/usr/lpp/info/lib/en_US/aix41** directory exists and creates it if it is not found.
- Checks to see if the database subdirectories exist. If the database subdirectory name is found in **/usr/lpp/info/lib/en_US/aix41** as:
  - A link from the CD–ROM, the script prints a message that the database is already linked from the CD–ROM.
  - A link from elsewhere, the script forces the link from the CD–ROM over the existing link.
  - A directory, the script prints a message that you must deinstall that database if you want to link it from the mounted CD–ROM.

  If the database subdirectory is not found in **/usr/lpp/info/lib/en_US/aix41** , the script links that database subdirectory from **/infocd/usr/lpp/info/lib/en_US/aix41** to **/usr/lpp/info/lib/en_US/aix41**.
- Checks to see if the **/usr/lpp/info/data/ispaths** file exists. If the **ispaths** file name is found as:
  - A link from the CD–ROM, the script prints a message that the **ispaths** file is already linked from the CD–ROM.

– A link from elsewhere, the script copies the linked **ispaths** file to **ispaths.linked** and links the **ispaths** file from the CD–ROM into /**usr**/**lpp**/**info**/**data**.

– A file, the script copies the existing **ispaths** to **ispaths.orig** and links the **ispaths** file from the CD–ROM into /**usr**/**lpp**/**info**/**data**.

If no /**usr**/**lpp**/**info**/**data**/**ispaths** file is found, the script links the **ispaths** file from the CD–ROM into /**usr**/**lpp**/**info**/**data**.

To run the **linkinfocd** script, enter:

```
/infocd/linkinfocd
```

Removing InfoExplorer Information Bases

The method you use to remove an information base depends on whether it is installed on the fixed disk from an installation media or linked from the mounted hypertext CD–ROM.

Prerequisites

You must have write access to the **/usr/lpp/info/lib/***$LANG* directory. *$LANG* refers to the language you are using for the InfoExplorer program.

Remove Information Bases Installed on Fixed Disk

To determine what database packages are installed on your system, run the **lslpp** command or use SMIT to generate a list. To remove an information base that is installed on the fixed disk, you must remove the corresponding software option. Refer to ”Maintaining Optional Software” in the *AIX Installation Guide*.

Remove Information Bases Linked from CD–ROM

To determine what databases are linked from the hypertext CD–ROM, use the **cd** command to change to the **/infocd/usr/lpp/info/lib/en_US/aix41** directory and run the **ls –l** command. Database directories that are linked from the hypertext CD–ROM will have symbolic links to **/infocd** listed.

Use the **rm** command to delete the symbolic links for any unneeded information bases. To delete the links, use the following form of the **rm** command:

```
rm –f  /usr/lpp/info/lib/ $LANG / InformationBaseName
```

For example, to remove the symbolic link for the **files** information base (*AIX Files Reference*) from a system where the language is U.S. English, enter:

```
rm –f /usr/lpp/info/lib/en_US/aix41/files
```

Note: Information bases cannot be deleted from the CD–ROM. For performance enhancements, you may decide to install frequently accessed information bases onto the fixed disk. Refer to ”Installing Optional Software and Service Updates” in the *AIX Installation Guide*.

Changing InfoExplorer Languages

Procedure

On a system with multiple languages installed, each language information base has its own directory in the **/usr/lpp/info/lib** directory. For example, the information base for German is installed in the **/usr/lpp/info/lib/de_DE** directory, while the information bases for U.S. English are installed in the **/usr/lpp/info/lib/en_US** directory.

InfoExplorer determines what languages to use for database content separately from the language to use for messages (like menu bar entries or button titles). The language used for messages is determined by the value of the **LANG** or the **LC_MESSAGES** environment variable. If **LC_MESSAGES** is set, that value is used; otherwise, the value of **LANG** is used to determine which messages to use.

InfoExplorer uses several methods for determining the language to use for database content. The precedence is as follows:

1. If the **INFOLANG** environment variable is set, then InfoExplorer attempts to read databases from the **/usr/lpp/info/lib/<$INFOLANG>** directory.
2. If no libraries are found there or **INFOLANG** environment variable is not set, then InfoExplorer uses the **INFOLOCALE** environment variable. You can specify a list of locales in the **INFOLOCALE** environment variable by separating each locale with a colon. The first locale specified is the first language that InfoExplorer attempts to read. InfoExplorer continues trying to read databases based on the locales specified in the **INFOLOCALE** environment variable until a match is found.
3. If no libraries are found using the **INFOLOCALE** environment variable, the environment variable **LC_MESSAGES** is used.
4. If no libraries are found using the **LC_MESSAGES**, the **LANG** environment variable is used.
5. If no libraries are found, then InfoExplorer defaults to using the libraries installed in **/usr/lpp/info/lib/en_US**.

Creating InfoExplorer Public Notes

Public notes can be read by any hypertext user. Public notes are created by merging and relocating private notes files.

Prerequisites

1. You must have write access to the **/usr/lpp/info/data** directory.
2. You must create and save private notes to a file in the InfoExplorer window or ASCII interface.

Procedure

Use one of the following methods to create public notes:

- For the default InfoExplorer library, use the **mergenote** command to merge private notes files into a single notes file. Designate the **/usr/lpp/info/data** directory as the location of the new public notes list.
- For public libraries other than the default, private notes are stored in the **$HOME/info/***LibraryName* directory. Use the **mergenote** command to merge private notes files into a single notes file. Designate the **/usr/lpp/info/data/***LibraryName* directory as the location of the new public notes list.

  Notes:

  a. Private notes and note lists are saved in the user's **$HOME/info** and **$HOME/info/***LibraryName***/notes** directories.

  b. Public notes for the default InfoExplorer library should be placed in the **/usr/lpp/info/data** directory.

Bookmarks created by one user can be copied for access by other hypertext users. Bookmark files are saved in the user's **$HOME/info** or **$HOME/info/***LibraryName* directory with a **.bmk** extension.

Prerequisites

You must have read and write access to the users **$HOME/info** directories.

Procedure

1. Use the **cp** command to copy a bookmark file from one user to another.

   For example, to copy bookmark file `review.bmk` in user `sharon`'s **$HOME** directory to user `donna`'s **$HOME** directory, enter:

   ```
   cd /home/sharon/info
   cp review.bmk /home/donna/info
   ```

2. In the InfoExplorer program, reset the new user's default bookmark file using the Defaults Editor window.

Transferring InfoExplorer Bookmarks from One User to Another

Bookmarks created by one user can be copied for access by other hypertext users. Bookmark files are saved in the user's **$HOME/info** or **$HOME/info**/*LibraryName* directory with a **.bmk** extension.

Prerequisites

You must have read and write access to the users **$HOME/info** directories.

Procedure

1. Use the **cp** command to copy a bookmark file from one user to another.

   For example, to copy bookmark file `review.bmk` in user `sharon`'s **$HOME** directory to user `donna`'s **$HOME** directory, enter:

   ```
   cd /home/sharon/info
   cp review.bmk /home/donna/info
   ```

2. In the InfoExplorer program, reset the new user's default bookmark file using the Defaults Editor window.

Index

Symbols

A

B

C

D

E

F

N

P

Q

R

S

T

U

V

W

X

Y

Z

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull AIX System Management Concepts: Operating System and Devices

Nº Reférence / Reference Nº : 86 A2 21KX 02

Daté / Dated : May 2000

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : _____ Date : _____

SOCIETE / COMPANY : _____

ADRESSE / ADDRESS : _____

_____

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

To order additional publications, please fill up a copy of this form and send it via mail to:
Pour commander des documents techniques, remplissez une copie de ce formulaire et envoyez-la à :

BULL ELECTRONICS ANGERS
CEDOC
ATTN / MME DUMOULIN
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

Managers / Gestionnaires :
Mrs. / Mme : **C. DUMOULIN** +33 (0) 2 41 73 76 65
Mr. / M : **L. CHERUBIN** +33 (0) 2 41 73 63 96

FAX : +33 (0) 2 41 73 60 19
E–Mail / Courrier Electronique : srv.Cedoc@franp.bull.fr

Or visit our web site at: / Ou visitez notre site web à:
`http://www-frec.bull.com` (PUBLICATIONS, Technical Literature, Ordering Form)

CEDOC Reference # Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
[_ _] : **no revision number means latest revision** / pas de numéro de révision signifie révision la plus récente					

NOM / NAME : ____________________ Date : ________

SOCIETE / COMPANY : ____________________

ADRESSE / ADDRESS : ____________________

____

PHONE / TELEPHONE : ____________________ FAX : ____________________

E–MAIL : ____________________

For Bull Subsidiaries / Pour les Filiales Bull :
Identification: ____________________

For Bull Affiliated Customers / Pour les Clients Affiliés Bull :
Customer Code / Code Client : ____________________

For Bull Internal Customers / Pour les Clients Internes Bull :
Budgetary Section / Section Budgétaire : ____________________

For Others / Pour les Autres :
Please ask your Bull representative. / Merci de demander à votre contact Bull.

PLACE BAR CODE IN LOWER LEFT CORNER

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 21KX 02

Bull

Utiliser les marques de découpe pour obtenir les étiquettes.
Use the cut marks to get the labels.

AIX

AIX System Management Concepts: Operating System and Devices

86 A2 21KX 02

AIX

AIX System Management Concepts: Operating System and Devices

86 A2 21KX 02

AIX

AIX System Management Concepts: Operating System and Devices

86 A2 21KX 02