

AIX 5L Version 5.3



Operating system and device management

AIX 5L Version 5.3



Operating system and device management

Note

Before using this information and the product it supports, read the information in "Notices," on page 541.

Second Edition (November 2007)

This edition applies to AIX 5L Version 5.3 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to Information Development, Department 04XA-905-6C006, 11501 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

(c) Copyright KnowledgeSet Corporation, Mountainview, California, 1990.

(c) Copyright AT&T, 1984, 1985, 1986, 1987, 1988, 1989. All rights reserved.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California. We acknowledge the following institutions for their role in its development: the Electrical Engineering and Computer Sciences Department at the Berkeley Campus.

The Rand MH Message Handling System was developed by the Rand Corporation and the University of California. Portions of the code and documentation described in this book were derived from code and documentation developed under the auspices of the Regents of the University of California and have been acquired and modified under the provisions that the following copyright notice and permission notice appear:

Copyright Regents of the University of California, 1986, 1987, 1988, 1989. All rights reserved.

Redistribution and use in source and binary forms are permitted provided that this notice is preserved and that due credit is given to the University of California at Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. This software is provided "as is" without express or implied warranty.

Copyright (c) 1993, 1994 Hewlett-Packard Company

Copyright (c) 1993, 1994 International Business Machines Corp.

Copyright (c) 1993, 1994 Sun Microsystems, Inc.

Copyright (c) 1993, 1994 Novell, Inc.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. HEWLETT-PACKARD COMPANY, INTERNATIONAL BUSINESS MACHINES CORP., SUN MICROSYSTEMS, INC., AND UNIX SYSTEMS LABORATORIES, INC., MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

© Copyright International Business Machines Corporation 2006, 2007. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book	xi
Chapter 1. System management	1
Available system management interfaces	1
Unique features of AIX	2
Using the man command	4
Chapter 2. System startup	5
Uninstalled system boot	5
Rebooting a running system	5
Remote reboot of an unresponsive system	5
Booting from hard disk for maintenance	6
Booting a system that has crashed	7
Systems that will not boot	7
Reactivation of an inactive system	8
Systems with planar graphics	11
Boot problem diagnostics	11
Boot images	12
Creating boot images	12
System run level	13
Run level script execution	15
Modifying the /etc/inittab file	15
Boot process	16
System boot processing	17
Maintenance boot process	18
RAM file system	19
Shutdown process	19
Resetting an unknown root password	22
Priority hang detection	23
Lost I/O hang detection	23
Chapter 3. Logical volume storage	25
Logical volume storage concepts	25
Volume group strategy	48
Logical volume strategy	49
Volume group policy implementation	59
Logical Volume Manager	60
Logical Volume Manager configuration tasks	60
Troubleshooting LVM	62
Vary-on process	73
Quorum	73
Nonquorum volume groups	74
Conversion of a volume group to nonquorum status	74
Replacing a failed physical volume in a mirrored volume group	75
Notifying the administrator when a physical volume is missing	79
Splitting a mirrored disk from a volume group	80
Chapter 4. Paging space and virtual memory	83
Paging space addition and activation	83
Paging performance improvement	83
Setting the PSALLOC environment variable for early allocation mode	83
Changing or removing a paging space	84
hd6 paging space resize and relocation	84

Paging space troubleshooting	86
Virtual Memory Manager	86
Paging space	88
Chapter 5. File systems	93
Organization and contents of the file tree	94
File system management tasks	102
File systems configuration tasks	103
File systems maintenance tasks	104
Disk overflows	108
Fixing a damaged file system.	112
Comparing file systems on different machines	113
Reducing the size of a file system in your root volume group	114
Mounting	117
Fixing a corrupted magic number in the file system superblock	122
File system types	123
Directories	137
Chapter 6. Backup methods	145
Backup policies.	146
Backup media	147
Backup strategy	148
Backing up user files or file systems	150
Backup of the system image and user-defined volume groups	150
Implementing scheduled backups	153
Backing up file systems using the cron command	154
Backup of files on a DMAPI-managed JFS2 file system	154
Formatting diskettes (format or fdformat command)	155
Checking the integrity of a file system (fsck command)	155
Copying to or from diskettes (fscopy command)	156
Copying files to tape or disk (cpio -o command)	156
Copying files from tape or disk (cpio -i command)	157
Copying to or from tapes (tcopy command)	158
Checking the integrity of a tape (tapechk command)	158
File backup	158
Compressing files (compress and pack commands)	159
Expanding compressed files (uncompress and unpack commands).	161
Restoring backed-up files	162
Creating a remote archive	164
Restoring user files from a backup image	166
Restoring access to an unlinked or deleted system library	166
Restoring a deleted symbolic link	167
Restoring a deleted system library file	167
Archiving files (tar command)	169
Command summary for backup files and storage media	169
Recreating a corrupted boot image	169
Making an online backup of a JFS.	170
Making and backing up a snapshot of a JFS2	171
Chapter 7. System environment	173
Profiles	173
Time data manipulation services	174
Dynamic Processor Deallocation	174
Filesets needed for 64-bit mode.	181
Hardware required for 64-bit mode.	182
Kernel extensions vs. 64-bit kernel.	182

Commands to change from 32-bit to 64-bit.	182
Commands to change from 64-bit to 32-bit.	182
32-bit and 64-bit performance comparisons	182
Changing the default browser	183
System environment variable setup	183
Chapter 8. Process management	187
Process monitoring	188
Process priority alteration	190
Process termination	190
Binding or unbinding a process	191
Chapter 9. Workload management	195
Workload management concepts	195
Workload Manager	201
Classes	213
Process classifications in Workload Manager	217
Resource management with Workload Manager.	221
Setup	228
Application programming interface	230
Commands	233
Examples of Workload Manager classification, rules, and limits	233
Backward compatibility in Workload Manager.	235
Exclusive use processor resource sets	236
Chapter 10. System Resource Controller	239
Subsystem components	239
SRC hierarchy	240
SRC administration commands	240
Starting the System Resource Controller	240
Starting or stopping a subsystem, subsystem group, or subserver	241
Displaying status of a subsystem or subsystems	242
Refreshing a subsystem or subsystem group	242
Subsystem control.	243
Chapter 11. System accounting	245
Setting up an accounting system	245
Displaying Accounting system activity	247
Showing Accounting system activity while running a command	247
Displaying the process time of active Accounting processes	248
Displaying the process time of finished Accounting processes.	248
Showing the CPU usage for each accounting process	248
Showing the CPU accounting usage for each user.	249
Displaying connect time usage for accounting	249
Displaying disk space utilization for Accounting	250
Displaying printer or plotter usage accounting records	250
Fixing tacct errors	251
Fixing wtmp errors	251
Fixing incorrect Accounting file permissions	252
Fixing Accounting errors	252
Holidays file update	256
System data collection and reporting	256
Connect-time accounting data	257
Connect-time reports.	257
Process accounting data	257
Process accounting reports	258

Disk-usage accounting data	258
Disk-usage accounting report	259
Printer-usage accounting data	259
Printer-Usage accounting report.	259
Fee accounting data	259
Fee accounting report	259
Fiscal accounting reports	259
Accounting system activity reports	260
Accounting data reports.	260
Accounting commands	263
Accounting files.	265
Chapter 12. Web-based System Manager	269
Chapter 13. System management interface tool	271
Chapter 14. Preparing to install a device	273
Chapter 15. Installing an IDE device	275
Determining the number and location of IDE controllers	275
Selecting an IDE controller and an IDE address on the controller	276
Setting up the hardware for IDE device installation.	278
Adding an IDE device to the customized configuration database.	278
Customizing the attributes for an IDE device	279
Chapter 16. Configuring a read/write optical drive	281
Chapter 17. Configuration of a large number of devices	283
Chapter 18. Device nodes	285
Device classes	285
Device configuration database and device management.	285
Device states	286
Chapter 19. Adding a removable media drive	287
Chapter 20. Device location codes	289
Adapter location codes	289
Printer and plotter location codes	290
tty location codes	290
SCSI device location codes	291
Direct-bus-attached location codes.	291
Serial-linked disk location codes	291
Diskette drive location codes.	292
Dials/LPFKeys location codes	292
Multiprotocol port location codes	292
Chapter 21. Setting up iSCSI	295
Configuring the iSCSI adapter in AIX	295
Updating the flat file of an iSCSI target	295
Adding a statically-discovered iSCSI target into ODM.	295
Adding statically-discovered iSCSI targets from a flat file into ODM.	296
Chapter 22. PCI hot plug management	297
Displaying PCI hot-plug slot information.	298
Unconfiguring PCI communications adapters	299

Removing or replacing a PCI hot plug adapter	299
Adding a PCI hot plug adapter	300
Chapter 23. Multiple Path I/O	303
MPIO-capable device management	304
MPIO device configuration.	306
Supported multi-path devices.	306
MPIO device attributes	307
Path control module attributes	308
Communications adapter removal	309
Adapter removal problem resolution	315
Unconfiguring storage adapters	316
Unconfiguring async adapters	317
Determining the cause of device problems.	317
Chapter 24. Tape drives	321
Tape drive attributes	321
Special files for tape drives	331
Chapter 25. Login names, system IDs, and passwords	333
Logging in to the operating system	333
Logging in more than one time (login command)	334
Becoming another user on a system (su command)	334
Suppressing login messages	335
Logging out of the operating system (exit and logout commands)	335
Shutdown command	335
Displaying your login name (whoami and logname commands)	336
Displaying the operating system name (uname command)	336
Displaying your system name (uname command)	337
Displaying all users who are logged in	337
Command for displaying user IDs (id command).	337
Passwords	338
Command summary for login names, system IDs, and passwords	339
Chapter 26. User environment	341
User environment files and customization procedures.	341
System devices list (lscfg command)	341
Displaying console names.	343
Displaying the terminal name (tty command)	343
Listing available displays (lsdisp command)	343
Listing available fonts (lsfont command).	343
Listing the current software keyboard map (lskbd command)	344
Listing available software products (lspp command)	344
List of control key assignments for your terminal (stty command)	344
Listing environment variables (env command)	345
Displaying an environment variable value (printenv command)	346
Bidirectional languages (aixterm command)	346
Command summary for user environment and system information	346
Chapter 27. Common Desktop Environment	347
Starting and stopping the Common Desktop Environment	347
Desktop profile modification	348
Adding and removing displays and terminals for the Common Desktop Environment	348
Display device customization for the Common Desktop Environment	350
Chapter 28. Commands and processes	353

Commands	353
Processes.	362
Command summary for commands and processes.	369
Chapter 29. Input and output redirection	371
Standard input, standard output, and standard error files	371
Standard output redirection	371
Redirecting output to a file.	372
Redirecting output to append to a file.	372
Creating a text file with redirection from the keyboard.	372
Text file concatenation	372
Standard input redirection	373
Discarding output with the /dev/null file	373
Standard error and other output redirection	373
Redirecting output to inline input (here) documents	374
Redirecting output using pipes and filters	374
Displaying program output and copying to a file (tee command)	375
Clearing your screen (clear command)	375
Sending a message to standard output	376
Appending a single line of text to a file (echo command)	376
Copying your screen to a file (capture and script commands)	376
Command to display text in large letters on your screen (banner command)	377
Command summary for input and output redirection	377
Chapter 30. Files	379
Types of files	379
File handling procedures	382
File and directory links	392
DOS files	394
Command summary for files	395
Chapter 31. Printers, print jobs, and queues	397
Printing terminology	397
Command for starting a print job (qprt command)	399
Canceling a print job (qcan command)	402
Prioritizing a print job (qpri command)	403
Moving a print job to another print queue (qmov command)	403
Holding and releasing a print job (qhld command)	404
Checking print job status (qchk command).	405
Command for formatting files for printing (pr command)	406
Command and flags for printing ASCII files on a PostScript printer	408
Automating the conversion of ASCII to PostScript	409
Overriding automatic determination of print file types	409
Command summary for printing.	409
Chapter 32. File and system security	411
Security threats.	411
File ownership and user groups.	412
Access control lists	416
Locking your terminal (lock or xlock command)	423
Command summary for file and system security.	424
Chapter 33. User environment customization	425
System startup files	425
AIXwindows startup files	427
Exporting shell variables (export shell command)	431

Changing the default font (chfont command)	431
Changing control keys (stty command)	432
Changing your system prompt	432
Chapter 34. Shells	435
Shell features	435
Korn shell or POSIX shell commands	440
Quotation of characters in the Korn shell or POSIX shell	445
Reserved words in the Korn shell or POSIX shell	447
Command aliasing in the Korn shell or POSIX shell	447
Parameters in the Korn shell	449
Parameter substitution in the Korn shell or POSIX shell	449
Predefined special parameters in the Korn shell or POSIX shell	450
Variables set by the Korn shell or POSIX shell	451
Variables used by the Korn shell or POSIX shell	451
Command substitution in the Korn shell or POSIX shell	453
Arithmetic evaluation in the Korn shell or POSIX shell	454
Field splitting in the Korn shell or POSIX shell	455
File name substitution in the Korn shell or POSIX shell	456
Input and output redirection in the Korn shell or POSIX shell	457
Exit status in the Korn shell or POSIX shell	460
Korn shell or POSIX shell built-in commands	460
List of Korn shell or POSIX shell special built-in commands	470
Korn shell or POSIX shell regular built-in commands	470
Conditional expressions for the Korn shell or POSIX shell	471
Job control in the Korn shell or POSIX shell	472
Inline editing in the Korn shell or POSIX shell	473
Enhanced Korn shell (ksh93)	479
Bourne shell	483
Restricted shell	484
Restricted Korn shell	484
Bourne shell commands	485
Variable substitution in the Bourne shell	492
User-defined variables in the Bourne shell	492
Conditional substitution in the Bourne shell	495
Positional parameters in the Bourne shell	496
File name substitution in the Bourne shell	497
Character classes	497
Input and output redirection in the Bourne shell	498
List of Bourne shell built-in commands	498
C shell	499
C shell commands	500
History substitution in the C shell	508
Alias substitution in the C shell	511
Variable substitution in the C shell	512
File name substitution in the C shell	513
File name expansion in the C shell	514
File name abbreviation in the C shell	514
Character classes in the C shell	515
Environment variables in the C shell	515
Input and output redirection in the C shell	517
Job control in the C shell	518
C shell built-in commands	519
Chapter 35. BSD systems reference	521
Introduction to AIX for BSD system managers	521

Major differences between 4.3 BSD and AIX	521
Accounting for BSD 4.3 system managers	524
Backup for BSD 4.3 system managers	525
Startup for BSD 4.3 system managers	525
Commands for system administration for BSD 4.3 system managers	526
Cron for BSD 4.3 system managers	528
Devices for BSD 4.3 system managers	528
File comparison table for 4.3 BSD, SVR4, and AIX.	529
File systems for BSD 4.3 system managers	530
Finding and examining files for BSD 4.3 system managers	531
Paging space for BSD 4.3 system managers	532
Changing the default startup to permit 4.3 BSD ASCII configuration	532
Additional options for ifconfig and netstat commands	532
Additional network management commands	532
Name and address resolution	533
Online documentation and man command for BSD 4.3 system managers	533
NFS and NIS (formerly Yellow Pages) for BSD 4.3 system managers	534
User passwords for BSD 4.3 system managers	534
Importing a BSD 4.3 password file.	535
Editing the password file for BSD 4.3 system managers	535
Performance measurement and tuning for BSD 4.3 system managers	537
Printers for BSD 4.3 system managers	537
Terminals for BSD 4.3 system managers	539
UUCP for BSD 4.3 system managers	539
Appendix. Notices	541
Trademarks	542
Index	545

About this book

This book provides users and system administrators with complete information that can affect your selection of options when performing such tasks as backing up and restoring the system, managing physical and logical storage, sizing appropriate paging space, and so on. It provides complete information about how to perform such tasks as managing logical volumes, storage, and resources. System users can learn how to perform such tasks as running commands, handling processes, handling files and directories, and basic printing. Other topics useful to users and system administrators include creating and re-sizing paging space, managing virtual memory, backing up and restoring the system, managing hardware and pseudo devices, using the System Resource Controller (SRC), securing files, using storage media, customizing environment files, and writing shell scripts. This book is also available on the documentation CD that is shipped with the operating system.

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.
<i>Italics</i>	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.

Case-Sensitivity in AIX

Everything in the AIX operating system is case-sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type **LS**, the system responds that the command is "not found." Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

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

Related Publications

- *Security*
- *Installation and migration*
- *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.3 Communications Programming Concepts*
- *AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts*
- *AIX 5L Version 5.3 Files Reference*
- *Performance Toolbox Version 2 and 3 for AIX: Guide and Reference*
- *Performance management*
- *Networks and communication management*
- *Printers and printing*
- *AIX 5L Version 5.3 Commands Reference*
- *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide.*

Beginning with the AIX 5.2 documentation library, any information that this book contained regarding AIX system security, or any security-related topic, has moved. For all security-related information, see the *Security*.

Chapter 1. System management

System management is the task of an individual who is usually referred to, in UNIX[®] literature, as the system administrator.

Unfortunately, only a few system administrator activities are straightforward enough to be correctly called administration. This and related guides are intended to help system administrators with their numerous duties.

This operating system provides its own particular version of system-management support in order to promote ease of use and to improve security and integrity.

Available system management interfaces

In addition to conventional command line system administration, this operating system provides the SMIT and Web-based System Manager interfaces.

The following are the SMIT and Web-based System Manager interfaces:

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

With SMIT, you can:

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

See Chapter 13, “System management interface tool,” on page 271 for more information on managing your system with SMIT.

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

Web-based System Manager can be configured to run in a variety of operating modes. The operating environments in which it can be started are standalone application, client-server, applet, and remote client. See Chapter 12, “Web-based System Manager,” on page 269 for more information about managing your system with Web-based System Manager.

Unique features of AIX

The following are some unique system management features of AIX® .

Logical Volume Manager

The Logical Volume Manager (LVM) maintains the hierarchy of logical structures that manage disk storage.

Disk drives are defined within this hierarchy as physical volumes. Every physical volume in use belongs to a volume group. Within each volume group, one or more logical volumes of information are defined. Data on logical volumes appears to be contiguous to the user, but can be discontinuous 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 Chapter 3, “Logical volume storage,” on page 25.

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 or 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 “Starting the System Resource Controller” on page 240.

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 (**lspp**) 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: applying, applied, committing, committed, rejecting, or broken.

Workload management

Workload Manager (WLM) is designed to provide the system administrator with increased control over how the scheduler virtual memory manager (VMM) and the disk I/O subsystem 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.

Attention: Efficient use of WLM requires extensive knowledge of existing system processes and performance. If the system administrator configures WLM with extreme or inaccurate values, performance will be significantly degraded.

WLM is primarily intended for use with large systems. Large systems are often used for server consolidation, in which workloads from many different server systems (such as printer, database, general user, and transaction processing systems) are combined into a single large system to reduce the cost of system maintenance. These workloads often interfere with each other and have different goals and service agreements.

WLM also provides isolation between user communities with very different system behaviors. This can prevent effective starvation of workloads with certain behaviors (for example, interactive or low CPU usage jobs) by workloads with other behaviors (for example, batch or high memory usage jobs).

Also, WLM ties into the accounting subsystem (see Chapter 11, “System accounting,” on page 245) allowing users to do resource usage accounting per WLM class in addition to the standard accounting per user or group.

Operating system updates

The operating system package is divided into filesets, where each fileset contains a group of logically related customer deliverable files. Each fileset can be individually installed and updated.

Revisions to filesets are tracked using the version, release, maintenance, and fix (VRMF) levels. By convention, each time an AIX fileset update is applied, the fix level is adjusted. Each time an AIX maintenance package or technology level is applied, the modification level is adjusted, and the fix level is reset to zero. The initial installation of an AIX version, for example, AIX 5.2, is called a base installation. The operating system provides updates to its features and functionality, which might be packaged as a maintenance package, a technology level, a program temporary fix (PTF), or a service pack (a group of PTFs).

Maintenance Packages and Technology Levels

Maintenance packages and technology levels provide new functionality that is intended to upgrade the release. The maintenance part of the VRMF is updated in a maintenance package. For example, the first maintenance package for AIX 5.2 would be 5.2.1.0; the second would be 5.2.2.0, and so forth. To list the maintenance package, use the **oslevel -r** command.

PTFs Between releases, you might receive PTFs to correct or prevent a particular problem. A particular installation might need some, all, or even none of the available PTFs.

Recommended Maintenance Packages

A recommended maintenance package is a set of PTFs between technology levels that have been extensively tested together and are recommended for preventive maintenance.

Interim Fixes

An interim fix is similar to a PTF, but it is usually offered when a PTF is not available. Interim fixes are also released when the PTF would upgrade a system to the next maintenance level and users might want their systems to remain at the current level.

To determine the version and release level, maintenance package, technology level, and service pack level, as well as which filesets need to be updated to reach a particular level, see the `oslevel` and the `lslpp` commands in *AIX 5L Version 5.3 Commands Reference*.

Updating your system

Use these commands to determine whether your system needs updating.

To determine the maintenance package or technology level installed on a particular system, type:

```
oslevel
```

To determine which filesets need updating for the system to reach a specific maintenance package or technology level (in this example, 4.3.3.0), use the following command:

```
oslevel -l 4.3.3.0
```

To determine if a recommended maintenance package or technology level is installed (in this example, 5100-02), use the following command:

```
oslevel -r 5100-02
```

To determine which filesets need updating for the system to reach the 5100-02 maintenance package or technology level, use the following command:

```
oslevel -r1 5100-02
```

To determine the maintenance package or technology level of a particular fileset (in this example, **bos.mp**), use the following command:

```
lslpp -L bos.mp
```

Using the man command

The **man** command is used mainly to access reference information on commands, subroutines, and files.

For example, to view information on the **ls** command, enter:

```
>man ls
```

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

For more information on the **man** command, see the *AIX 5L Version 5.3 Commands Reference, Volume 3*. Also see “Online documentation and man command for BSD 4.3 system managers” on page 533.

Chapter 2. System startup

When the base operating system starts, the system initiates a complex set of tasks. Under normal conditions, these tasks are performed automatically.

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

Uninstalled system boot

The procedure for booting a new or uninstalled system is part of the installation process.

For information on how to boot an uninstalled system, see *Installation and migration*.

Rebooting a running system

Because processes might be running that should be terminated more gracefully than a **reboot** permits, **shutdown** is the preferred method for all systems.

There are two methods for shutting down and rebooting your system, **shutdown** and **reboot**. Always use the **shutdown** method when multiple users are logged onto the system.

Rebooting a Running System Tasks		
Web-based System Manager	wsm, then select System	
-OR-		
Task	SMIT Fast Path	Command or File
Rebooting a Multiuser System	smit shutdown	shutdown -r
Rebooting a Single-User System	smit shutdown	shutdown -r or reboot

Remote reboot of an unresponsive system

The remote reboot facility allows the system to be rebooted through a native (integrated) system port.

The POWER5™ integrated *system ports* are similar to serial ports except that system ports are available only for specifically supported functions. Refer to Function differences between system ports and serial ports for more information.

The system is rebooted when the **reboot_string** is received at the port. This facility is useful when the system does not otherwise respond but is capable of servicing system port interrupts. Remote reboot can be enabled on only one native system port at a time. Users are expected to provide their own external security for the port. This facility runs at the highest device interrupt class and a failure of the UART (Universal Asynchronous Receive/Transmit) to clear the transmit buffer quickly may have the effect of causing other devices to lose data if their buffers overflow during this time. It is suggested that this facility only be used to reboot a machine that is otherwise hung and cannot be remotely logged into. File systems will *not* be synchronized, and a potential for some loss of data which has not been flushed exists. It is strongly suggested that when remote reboot is enabled that the port not be used for any other purpose, especially file transfer, to prevent an inadvertent reboot.

Two native system port attributes control the operation of remote reboot.

reboot_enable

Indicates whether this port is enabled to reboot the machine on receipt of the remote **reboot_string**, and if so, whether to take a system dump prior to rebooting.

no - Indicates remote reboot is disabled
reboot - Indicates remote reboot is enabled
dump - Indicates remote reboot is enabled, and prior to rebooting a system dump will be taken on the primary dump device

reboot_string

Specifies the remote **reboot_string** that the serial port will scan for when the remote reboot feature is enabled. When the remote reboot feature is enabled and the **reboot_string** is received on the port, a > character is transmitted and the system is ready to reboot. If a 1 character is received, the system is rebooted; any character other than 1 aborts the reboot process. The **reboot_string** has a maximum length of 16 characters and must not contain a space, colon, equal sign, null, new line, or Ctrl-\ character.

Remote reboot can be enabled through SMIT or the command line. For SMIT the path **System Environments -> Manage Remote Reboot Facility** may be used for a configured TTY. Alternatively, when configuring a new TTY, remote reboot may be enabled from the **Add a TTY** or **Change/Show Characteristics of a TTY** menus. These menus are accessed through the path **Devices -> TTY**.

From the command line, the **mkdev** or **chdev** commands are used to enable remote reboot. For example, the following command enables remote reboot (with the dump option) and sets the reboot string to **ReBoOtMe** on **tty1**.

```
chdev -l tty1 -a remreboot=dump -a reboot_string=ReBoOtMe
```

This example enables remote reboot on **tty0** with the current **reboot_string** in the database only (will take effect on the next reboot).

```
chdev -P -l tty0 -a remreboot=reboot
```

If the tty is being used as a normal port, then you will have to use the **pdisable** command before enabling remote reboot. You may use **penable** to reenabte the port afterwards.

Booting from hard disk for maintenance

You can boot a machine in maintenance mode from a hard disk.

Prerequisites

A bootable removable media (tape or CD-ROM) must not be in the drive. Also, refer to the hardware documentation for the specific instructions to enable maintenance mode boot on your particular model.

Procedure

To boot a machine in maintenance mode from a hard disk:

1. To reboot, either turn the machine off and then power it back on, or press the reset button.
2. Press the key sequence for rebooting in maintenance mode that is specified in your hardware documentation.
3. The machine will boot to a point where it has a console device configured. If there is a system dump that needs to be retrieved, the system dump menu will be displayed on the console.

Note:

- a. If the console fails to configure when there is a dump to be retrieved, the system will hang. The system must be booted from a removable medium to retrieve the dump.

- b. Beginning in AIX 5.3, the system will automatically dump to the specified dump device when the reset button is pressed. For more information on how the system dumps, see *Starting a System Dump in AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts*. To change the primary or secondary dump device designation in a running system, see the **sysdumpdev** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5*.
4. If there is no system dump, or if it has been copied, the diagnostic operating instructions will be displayed. Press Enter to continue to the **Function Selection** menu.
5. From the **Function Selection** menu, you can select diagnostic or single-user mode:
Single-User Mode: To perform maintenance in a single-user environment, choose this option (option 5). The system continues to boot and enters single-user mode. Maintenance that requires the system to be in a standalone mode can be performed in this mode, and the **bosboot** command can be run, if required.

Booting a system that has crashed

In some instances, you might have to boot a system that has stopped (crashed) without being properly shut down.

The prerequisites for this procedure are:

- Your system crashed and was not properly shut down due to unusual conditions.
- Your system is turned off.

This procedure covers the basics of how to boot if your system was unable to recover from a crash. Perform the following steps:

1. Ensure that all hardware and peripheral devices are correctly connected.
2. Turn on all of the peripheral devices.
3. Watch the screen for information about automatic hardware diagnostics.
 - a. If any hardware diagnostics tests are unsuccessful, refer to the hardware documentation.
 - b. If all hardware diagnostics tests are successful, turn the system unit on.

Systems that will not boot

If a system will not boot from the hard disk, you may still be able to gain access to the system in order to ascertain and correct the problem.

If you have a system that will not boot from the hard disk, see the procedure on how to access a system that will not boot in *Troubleshooting your installation* in the *Installation and migration*.

This procedure enables you to get a system prompt so that you can attempt to recover data from the system or perform corrective action enabling the system to boot from the hard disk.

Note:

- This procedure is intended only for experienced system managers who have knowledge of how to boot or recover data from a system that is unable to boot from the hard disk. Most users should not attempt this procedure, but should contact their service representative.
- This procedure is not intended for system managers who have just completed a new installation, because in this case the system does not contain data that needs to be recovered. If you are unable to boot from the hard disk after completing a new installation, contact your service representative.

Reactivation of an inactive system

Your system can become inactive because of a hardware problem, a software problem, or a combination of both.

This procedure guides you through steps to correct the problem and restart your system. If your system is still inactive after completing the procedure, refer to the problem-determination information in your hardware documentation.

Use the following procedures to reactivate an inactive system:

Hardware check

There are several procedures you can use to check your hardware.

Check your hardware by:

Checking the power:

If the Power-On light on your system is active, go to **Checking the operator panel display**, below.

If the Power-On light on your system is not active, check that the power is on and the system is plugged in.

Checking the operator panel display:

If your system has an operator panel display, check it for any messages.

If the operator panel display on your system is blank, go to **Activating your display or terminal**, below.

If the operator panel display on your system is not blank, go to the service guide for your unit to find information concerning digits in the Operator Panel Display.

Activating your display or terminal:

Check several parts of your display or terminal, as follows:

- Make sure the display cable is securely attached to the display and to the system unit.
- Make sure the keyboard cable is securely attached.
- Make sure the mouse cable is securely attached.
- Make sure the display is turned on and that its Power-On light is lit.
- Adjust the brightness control on the display.
- Make sure the terminal's communication settings are correct.

If your system is now active, your hardware checks have corrected the problem.

If your system became inactive while you were trying to restart the system, go to "Restarting the system" on page 10.

If your system did not become inactive while you were trying to restart the system, go to "Checking the processes" on page 9.

Checking the processes

A stopped or stalled process might make your system inactive.

Check your system processes by:

1. Restarting line scrolling
2. Using the Ctrl+D key sequence
3. Using the Ctrl+C key sequence
4. Logging in from a remote terminal or host
5. Ending stalled processes remotely

Restarting line scrolling:

Restart line scrolling halted by the Ctrl-S key sequence by doing the following:

1. Activate the window or shell with the problem process.
2. Press the Ctrl-Q key sequence to restart scrolling. The Ctrl-S key sequence stops line scrolling, and the Ctrl-Q key sequence restarts line scrolling.

If your scroll check did not correct the problem with your inactive system, go to the next section, **Using the Ctrl-D key sequence**.

Using the Ctrl-D key sequence:

1. Activate the window or shell with the problem process.
2. Press the Ctrl-D key sequence. The Ctrl-D key sequence sends an end of file (EOF) signal to the process. The Ctrl-D key sequence may close the window or shell and log you out.

If the Ctrl-D key sequence did not correct the problem with your inactive system, go to the next section, **Using the Ctrl-C key sequence**.

Using the Ctrl-C key sequence:

End a stopped process by doing the following:

1. Activate the window or shell with the problem process.
2. Press the Ctrl-C key sequence. The Ctrl-C key sequence stops the current search or filter.

If the Ctrl-C key sequence did not correct the problem with your inactive system, go to the next section, **Logging in from a remote terminal or host**.

Logging in from a remote terminal or host:

Log in remotely in either of two ways:

- Log in to the system from another terminal if more than one terminal is attached to your system.
- Log in from another host on the network (if your system is connected to a network) by typing the **tn** command as follows:

```
tn YourSystemName
```

The system asks for your regular login name and password when you use the **tn** command.

If you were able to log in to the system from a remote terminal or host, go to the next section, **Ending stalled processes remotely**.

If you were not able to log in to the system from a remote terminal or host, go to “Restarting the system” on page 10.

You can also start a system dump to determine why your system became inactive. For more information, see System Dump Facility .

Ending stalled processes remotely:

End a stalled process from a remote terminal by doing the following:

1. List active processes by typing the following **ps** command:

```
ps -ef
```

The **-e** and **-f** flags identify all active and inactive processes.

2. Identify the process ID of the stalled process.

For help in identifying processes, use the **grep** command with a search string. For example, to end the **xlock** process, type the following to find the process ID:

```
ps -ef | grep xlock
```

The **grep** command allows you to search on the output from the **ps** command to identify the process ID of a specific process.

3. End the process by typing the following **kill** command:

Note: You must have root user authority to use the **kill** command on processes you did not initiate.

```
kill -9 ProcessID
```

If you cannot identify the problem process, the most recently activated process might be the cause of your inactive system. End the most recent process if you think that is the problem.

If your process checks have not corrected the problem with your inactive system, go to “Restarting the system.”

You can also start a system dump to determine why your system became inactive. For more information, see System Dump Facility.

Restarting the system

In addition to checking the hardware and checking the processes, you can restart you system to reactivate an inactive system.

If the procedures for “Hardware check” on page 8 and “Checking the processes” on page 9 fail to correct the problem that makes your system inactive, you need to restart your system.

Note: Before restarting your system, complete a system dump. For more information, see System Dump Facility .

1. Check the state of the boot device.

Your system boots with either a removable medium, an external device, a small computer system interface (SCSI) device, an integrated device electronics (IDE) device, or a local area network (LAN). Decide which method applies to your system, and use the following instructions to check the boot device:

- For a removable medium, such as tape, make sure the medium is inserted correctly.
- For IDE devices, verify that the IDE device ID settings are unique per adapter. If only one device is attached to the adapter, the IDE device ID must be set to the master device.
- For an externally attached device, such as a tape drive, make sure:
 - The power to the device is turned on.
 - The device cables are correctly attached to the device and to the system unit.
 - The ready indicator is on (if the device has one).

- For external SCSI devices, verify that the SCSI address settings are unique.
- For a LAN, verify that the network is up and operable.

If the boot device is working correctly, continue to the next step.

2. Load your operating system by doing the following:
 - a. Turn off your system's power.
 - b. Wait one minute.
 - c. Turn on your system's power.
 - d. Wait for the system to boot.

If the operating system failed to load, boot the hard disk from maintenance mode or hardware diagnostics.

If you are still unable to restart the system, use an SRN to report the problem with your inactive system to your service representative.

Systems with planar graphics

If the machine has been installed with the planar graphics subsystem only, and later an additional graphics adapter is added to the system, the following occurs:

1. A new graphics adapter is added to the system, and its associated device driver software is installed.
2. The system is rebooted, and one of the following occurs:
 - a. If the system console is defined to be `/dev/lft0` (`lscons` displays this information), the user is asked to select which display is the system console at reboot time. If the user selects a graphics adapter (non-TTY device), it also becomes the new default display. If the user selects a TTY device instead of an LFT device, no system login appears. Reboot again, and the TTY login screen is displayed. It is assumed that if the user adds an additional graphics adapter into the system and the system console is an LFT device, the user will not select the TTY device as the system console.
 - b. If the system console is defined to be a TTY, then at reboot time the newly added display adapter becomes the default display.
3. If the system console is `/def/lft0`, then after reboot, DPMS is disabled in order to show the system console selection text on the screen for an indefinite period of time. To re-enable DPMS, reboot the system again.

Note: Since the TTY is the system console, it remains the system console.

Boot problem diagnostics

A variety of factors can cause a system to be unable to boot.

Some of these factors are:

- Hardware problems
- Defective boot tapes or CD-ROMs
- Improperly configured network boot servers
- Damaged file systems
- Errors in scripts such as `/sbin/rc.boot`

For information on accessing a system that will not boot from the disk drive, see “Systems that will not boot” on page 7.

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 about the **bosboot** command, see **bosboot**.

Creating boot images

To install the base operating system or to access a system that will not boot from the system hard drive, you need a boot image. This procedure describes how to create boot images. The boot image varies for each type of device.

The associated RAM disk file system contains device configuration routines for the following devices:

- **Disk**
- **Tape**
- **CD-ROM**
- **Network Token-Ring, Ethernet, or FDDI device**
- You must have root user authority to use the **bosboot** command.
- The **/tmp** file system must have at least 20 MB of free space.
- The physical disk must contain the boot logical volume. To determine which disk device to specify, type the following at a command prompt:

```
lsvg -l rootvg
```

The **lsvg -l** command lists the logical volumes on the root volume group (rootvg). From this list you can find the name of the boot logical volume.

Then type the following at a command prompt:

```
lsvg -M rootvg
```

The **lsvg -M** command lists the physical disks that contain the various logical volumes.

Creating a boot image on a boot logical volume

If the base operating system is being installed (either a new installation or an update), the **bosboot** command is called to place the boot image on the boot logical volume. The boot logical volume is a physically contiguous area on the disk created through the Logical Volume Manager (LVM) during installation.

For a list of prerequisites for this procedure, see “Creating boot images.”

The **bosboot** command does the following:

1. Checks the file system to see if there is enough room to create the boot image.
2. Creates a RAM file system using the **mkfs** command and a prototype file.
3. Calls the **mkboot** command, which merges the kernel and the RAM file system into a boot image.
4. Writes the boot image to the boot logical volume.

To create a boot image on the default boot logical volume on the fixed disk, type the following at a command prompt:

```
bosboot -a
```

OR:

```
bosboot -ad /dev/ipldevice
```

Note: Do not reboot the machine if the **bosboot** command fails while creating a boot image. Resolve the problem and run the **bosboot** command to successful completion.

You must reboot the system for the new boot image to be available for use.

Creating boot images for network devices

You can create boot images for an Ethernet boot or Token-Ring boot.

For a list of prerequisites for this procedure, see “Creating boot images” on page 12.

To create a boot image for an Ethernet boot, type the following at a command prompt:

```
bosboot -ad /dev/ent
```

For a Token-Ring boot:

```
bosboot -ad /dev/tok
```

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 [Changing the /etc/inittab File](#).

Identifying the system run level

Before performing maintenance on the operating system or changing the system run level, you might need to examine the various run levels.

This procedure describes how to identify the run level at which the system is operating and how to display a history of previous run levels. The `init` command determines the system run level.

Identification of the current run level

At the command line, type `cat /etc/.init.state`. The system displays one digit; that is the current run level. See the `init` command or the `/etc/inittab` file for more information about run levels.

Displaying a history of previous run levels

You can display a history of previous run levels using the `fwtmp` command.

Note: The `bosext2.acct.obj` code must be installed on your system to use this command.

1. Log in as root user.
2. Type the following at a command prompt:

```
/usr/lib/acct/fwtmp </var/adm/wtmp |grep run-level
```

The system displays information similar to the following:

```
run-level 2 0 1 0062 0123 697081013 Sun Feb 2 19:36:53 CST 1992
run-level 2 0 1 0062 0123 697092441 Sun Feb 2 22:47:21 CST 1992
run-level 4 0 1 0062 0123 698180044 Sat Feb 15 12:54:04 CST 1992
run-level 2 0 1 0062 0123 698959131 Sun Feb 16 10:52:11 CST 1992
run-level 5 0 1 0062 0123 698967773 Mon Feb 24 15:42:53 CST 1992
```

System run level changes

There are two methods of changing system run levels for multi-user or single-user systems.

When the system starts the first time, it enters the default run level defined by the `initdefault` entry in the `/etc/inittab` file. The system operates at that run level until it receives a signal to change it.

The following are the currently-defined run levels:

0-9	When the init command changes to run levels 0-9, it kills all processes at the current run levels then restarts any processes associated with the new run levels.
0-1	Reserved for the future use of the operating system.
2	Default run level.
3-9	Can be defined according to the user's preferences.
a, b, c	When the init command requests a change to run levels a , b , or c , it does not kill processes at the current run levels; it simply starts any processes assigned with the new run levels.
Q, q	Tells the init command to reexamine the <code>/etc/inittab</code> file.

Changing run levels on multiuser systems

You can change run levels on multiuser systems.

1. Check the `/etc/inittab` file to confirm that the run level to which you are changing supports the processes that you are running. The `getty` process is particularly important, since it controls the terminal line access for the system console and other logins. Ensure that the `getty` process is enabled at all run levels.
2. Use the **wall** command to inform all users that you intend to change the run level and request that users log off. For more information about the **wall** command, see **wall**.
3. Use the **smit telinit** fast path to access the **Set System Run Level** menu.
4. Type the new run level in the **System RUN LEVEL** field.
5. Press Enter to implement all of the settings in this procedure. The system responds by telling you which processes are terminating or starting as a result of the change in run level and by displaying the message:

```
INIT: New run level: n
```

where *n* is the new run-level number.

Changing run levels on single-user systems

You can change run levels on single-user systems.

1. Check the `/etc/inittab` file to confirm that the run level to which you are changing supports the processes that you are running. The `getty` process is particularly important, since it controls the terminal line access for the system console and other logins. Ensure that the `getty` process is enabled at all run levels. For more information about the `inittab` file, see **inittab**.
2. Use the **smit telinit** fast path to access the **Set System Run Level** menu. For more information about the **telinit** command, see **telinit**.
3. Type the new system run level in the **System RUN LEVEL** field.
4. Press Enter to implement all of the settings in this procedure.

The system responds by telling you which processes are terminating or starting as a result of the change in run level and by displaying the message:

```
INIT: New run level: n
```

where *n* is the new run-level number.

Run level script execution

Run level scripts allow users to start and stop selected applications while changing the run level.

Put run level scripts in the subdirectory of **/etc/rc.d** that is specific to the run level:

- **/etc/rc.d/rc2.d**
- **/etc/rc.d/rc3.d**
- **/etc/rc.d/rc4.d**
- **/etc/rc.d/rc5.d**
- **/etc/rc.d/rc6.d**
- **/etc/rc.d/rc7.d**
- **/etc/rc.d/rc8.d**
- **/etc/rc.d/rc9.d**

The **/etc/rc.d/rc** will run the scripts it finds in the specified directory when the run level changes - first running the stop application scripts then running the start application scripts.

Note: Scripts beginning with K are stop scripts, while scripts beginning with S are start scripts.

Modifying the /etc/inittab file

Four commands are available to modify the records in the **etc/inittab** file.

Adding records - mkitab command

To add a record to the **/etc/inittab** file, type the following at a command prompt:

```
mkitab Identifier:Run Level:Action:Command
```

For example, to add a record for **tty2**, type the following at a command prompt:

```
mkitab tty002:2:respawn:/usr/sbin/getty /dev/tty2
```

In the above example:

tty002	Identifies the object whose run level you are defining.
2	Specifies the run level at which this process runs.
respawn	Specifies the action that the init command should take for this process.
/usr/sbin/getty /dev/tty2	Specifies the shell command to be executed.

Changing records - chitab command

To change a record to the **/etc/inittab** file, type the following at a command prompt:

```
chitab Identifier:Run Level:Action:Command
```

For example, to change a record for **tty2** so that this process runs at run levels 2 and 3, type:

```
chitab tty002:23:respawn:/usr/sbin/getty /dev/tty2
```

In the above example:

tty002	Identifies the object whose run level you are defining.
23	Specifies the run levels at which this process runs.
respawn	Specifies the action that the init command should take for this process.

/usr/sbin/getty /dev/tty2

Specifies the shell command to be executed.

Listing records - lsitab command

To list all records in the **/etc/inittab** file, type the following at a command prompt:

```
lsitab -a
```

To list a specific record in the **/etc/inittab** file, type:

```
lsitab Identifier
```

For example, to list the record for **tty2**, type: `lsitab tty2`.

Removing records - rmitab command

To remove a record from the **/etc/inittab** file, type the following at a command prompt:

```
rmitab Identifier
```

For example, to remove the record for **tty2**, type: `rmitab tty2`.

Boot process

There are three types of system boots and two resources that are required in order to boot the operating system.

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

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

There are three types of system boots:

Hard Disk Boot

A machine is started for normal operations. For more information, see “System boot processing” on page 17.

Diskless Network Boot

A diskless or dataless workstation is started remotely over a network. A machine is started for normal operations. One or more remote file servers provide the files and programs that diskless or dataless workstations need to boot.

Maintenance Boot

A machine is started from a hard disk, network, tape, or CD-ROM in maintenance mode. A system administrator can perform tasks such as installing new or updated software and running diagnostic checks. For more information, see “Maintenance boot process” on page 18.

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 “RAM file system” on page 19.

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.

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:

ROS kernel init phase

The ROS kernel resides in firmware.

Its initialization phase involves the following steps:

1. The firmware checks to see if there are any problems with the system board. Control is passed to ROS, which performs a power-on self-test (POST).
2. The ROS initial program load (IPL) checks the user boot list, a list of available boot devices. This boot list can be altered to suit your requirements using the **bootlist** command. If the user boot list in non-volatile random access memory (NVRAM) is not valid or if a valid boot device is not found, the default boot list is then checked. In either case, the first valid boot device found in the boot list is used for system startup. If a valid user boot list exists in NVRAM, the devices in the list are checked in order. If no user boot list exists, all adapters and devices on the bus are checked. In either case, devices are checked in a continuous loop until a valid boot device is found for system startup.

Note: The system maintains a default boot list that is stored in NVRAM for normal mode boot. A separate service mode boot list is also stored in NVRAM, and you should refer to the specific hardware instructions for your model to learn how to access the service mode boot list.

3. When a valid boot device is found, the first record or program sector number (PSN) is checked. If it is a valid boot record, it is read into memory and is added to the IPL control block in memory. Included in the key boot record data are the starting location of the boot image on the boot device, the length of the boot image, and instructions on where to load the boot image in memory.
4. The boot image is read sequentially from the boot device into memory starting at the location specified in NVRAM. 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. The kernel runs **init**, which runs phase 1 of the **rc.boot** script.

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

Base device configuration phase

The **init** process starts the **rc.boot** script. Phase 1 of the **rc.boot** script performs the base device configuration.

Phase 1 of the **rc.boot** script includes the following steps:

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

Booting the system

Use these steps to complete the system boot phase.

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

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

Maintenance boot process

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

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

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

Note: For a normal boot, the system maintains a default boot list located in ROS, and a user boot list stored in NVRAM. Separate default and user boot lists are also maintained for booting in maintenance mode.

4. When a valid boot device is found, the first record or program sector number (PSN) is checked. If it is a valid boot record, it is read into memory and is added to the initial program load (IPL) control block in memory. Included in the key boot record data are the starting location of the boot image on the boot device, the length of the boot image, and the offset to the entry point to start running when the boot image is in memory.
5. The boot image is read sequentially from the boot device into memory, starting at the location specified in NVRAM.
6. Control is passed to the kernel, which begins running programs in the RAM file system.
7. The ODM database contents determine which devices are present, and the **cfgmgr** command dynamically configures all devices found, including all disks which are to contain the root file system.

8. If a CD-ROM, tape, or the network is used to boot the system, the rootvg volume group (or rootvg) is not varied on, because the rootvg might not exist (as is the case when installing the operating system on a new system). Network configuration can occur at this time. No paging occurs when a maintenance 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 rootvg 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 that allows you to enter various diagnostics modes or single-user mode. Selecting single-user mode allows the user to continue the boot process and enter single-user mode, where the **init** run level is set to "S". The system is then ready for maintenance, software updates, or running the **bosboot** command.

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 are specific to the type of boot.

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

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

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

Shutdown process

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

You might want to shut down your system:

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

System termination

The **shutdown** command is the safest and most thorough way to halt the operating system.

When you designate the appropriate flags, this command notifies users that the system is about to go down, kills all existing processes, unmounts file systems, and halts the system. See **shutdown** for more information.

There are several methods for shutting down the system.

Shutting down the system without rebooting

There are two ways of shutting down the system with no reboot.

You can use two methods to shut down the system without rebooting: the SMIT fastpath, or the **shutdown** command.

Prerequisites

You must have root user authority to shut down the system.

To shut down the system using SMIT:

1. Log in as root.
2. At the command prompt, type:
`smit shutdown`

To shut down the system using the **shutdown** command:

1. Log in as root.
2. At the command prompt, type:
`shutdown`

Shutting down the system to single-user mode

In some cases, you might need to shut down the system and enter single-user mode to perform software maintenance and diagnostics.

1. Type `cd /` to change to the root directory. You must be in the root directory to shut down the system to single-user mode to ensure that file systems are unmounted cleanly.
2. Type `shutdown -m`. The system shuts down to single-user mode.

A system prompt displays and you can perform maintenance activities.

Shutting down the system in an emergency

Use the **shutdown** command to stop the system quickly without notifying other users.

You can use the **shutdown** command to shut down the system under emergency conditions.

Type `shutdown -F`. The **-F** flag instructs the **shutdown** command to bypass sending messages to other users and shut down the system as quickly as possible.

System hang management

System hang management allows users to run mission-critical applications continuously while improving application availability. System hang detection alerts the system administrator of possible problems and then allows the administrator to log in as root or to reboot the system to resolve the problem.

shconf command

The **shconf** command is invoked when **System Hang Detection** is enabled. The **shconf** command configures which events are surveyed and what actions are to be taken if such events occur. You can

specify any of the following actions, the priority level to check, the time out while no process or thread executes at a lower or equal priority, the terminal device for the warning action, and the **getty** command action:

- Log an error in **errlog** file
- Display a warning message on the system console (alphanumeric console) or on a specified TTY
- Reboot the system
- Give a special **getty** to allow the user to log in as root and launch commands
- Launch a command

For the **Launch a command** and **Give a special getty** options, system hang detection launches the special **getty** command or the specified command at the highest priority. The special **getty** command prints a warning message that it is a recovering **getty** running at priority 0. The following table captures the various actions and the associated default parameters for priority hang detection. Only one action is enabled for each type of detection.

Option	Enablement	Priority	Timeout (seconds)
Log an error in errlog file	disabled	60	120
Display a warning message	disabled	60	120
Give a recovering getty	enabled	60	120
Launch a command	disabled	60	120
Reboot the system	disabled	39	300

Note: When **Launch a recovering getty on a console** is enabled, the **shconf** command adds the **-u** flag to the **getty** command in the **inittab** that is associated with the console login.

For lost IO detection, you can set the time out value and enable the following actions:

Option	Enablement
Display a warning message	disabled
Reboot the system	disabled

Lost IO events are recorded in the Web-based System Manager error log file.

shdaemon daemon

The **shdaemon** daemon is a process that is launched by **init** and runs at priority 0 (zero). It is in charge of handling system hang detection by retrieving configuration information, initiating working structures, and starting detection times set by the user.

System hang detection configuration

You can manage the system hang detection configuration from the SMIT management tool.

SMIT menu options allow you to enable or disable the detection mechanism, display the current state of the feature, and change or show the current configuration. The fast paths for system hang detection menus are:

smit shd

Manage System Hang Detection

smit shstatus

System Hang Detection Status

smit shpriocfg

Change/Show Characteristics of Priority Problem Detection

smit shreset

Restore Default Priority Problem Configuration

smit shliocfg

Change/Show Characteristics of Lost I/O Detection

smit shlioreset

Restore Default Lost I/O Detection Configuration

You can also manage system hang detection using the **shconf** command.

Resetting an unknown root password

The following procedure describes how to recover access to root privileges when the system's root password is unavailable or unknown.

The following procedure requires some system downtime. If possible, schedule your downtime when it least impacts your workload to protect yourself from a possible loss of data or functionality.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Insert the product media for the same version and level as the current installation into the appropriate drive.
2. Power on the machine.
3. When the screen of icons appears, or when you hear a double beep, press the F1 key repeatedly until the **System Management Services** menu appears.
4. Select **Multiboot**.
5. Select **Install From**.
6. Select the device that holds the product media and then select **Install**.
7. Select the AIX version icon.
8. Define your current system as the system console by pressing the F1 key and then press Enter.
9. Select the number of your preferred language and press Enter.
10. Choose **Start Maintenance Mode for System Recovery** by typing 3 and press Enter.
11. Select **Access a Root Volume Group**. A message displays explaining that you will not be able to return to the Installation menus without rebooting if you change the root volume group at this point.
12. Type 0 and press Enter.
13. Type the number of the appropriate volume group from the list and press Enter.
14. Select **Access this Volume Group and start a shell** by typing 1 and press Enter.
15. At the # (number sign) prompt, type the **passwd** command at the command line prompt to reset the root password. For example:

```
# passwd
Changing password for "root"
root's New password:
Enter the new password again:
```
16. To write everything from the buffer to the hard disk and reboot the system, type the following:

```
sync;sync;sync;reboot
```

When the login screen appears, the password you set in step 15 should now permit access to root privileges.

For more information

For more information, refer to:

- The **passwd** and **reboot** commands descriptions in *AIX 5L Version 5.3 Commands Reference, Volume 4*
- "Boot Problems" in *Problem Solving and Troubleshooting in AIX 5L™*, an IBM® Redbooks® publication

Priority hang detection

AIX can detect system hang conditions and try to recover from such situations, based on user-defined actions.

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

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

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

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

For more information on system hang detection, see "System hang management" on page 20.

Lost I/O hang detection

AIX can detect system hang conditions and try to recover from such situations, based on user-defined actions.

Because of I/O errors, the I/O path can become blocked and further I/O on that path is affected. In these circumstances it is essential that the operating system alert the user and execute user defined actions. As part of the Lost I/O detection and notification, the **shdaemon**, with the help of the Logical Volume Manager, monitors the I/O buffers over a period of time and checks whether any I/O is pending for too long a period of time. If the wait time exceeds the threshold wait time defined by the **shconf** file, a lost I/O is detected and further actions are taken. The information about the lost I/O is documented in the error log. Also based on the settings in the **shconf** file, the system might be rebooted to recover from the lost I/O situation.

For lost I/O detection, you can set the time out value and also enable the following actions:

Action	Default Enabled	Default Device
Console message	no	/dev/console
Crash and reboot	no	-

For more information on system hang detection, see “System hang management” on page 20.

Chapter 3. Logical volume storage

Logical volumes are groups of information located on physical volumes.

A hierarchy of structures is used to manage disk storage. Each individual 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. 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.

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

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

Logical volumes can serve a number of system purposes, such as paging, but each logical volume serves a single purpose only. Many logical volumes contain a single journaled file system (JFS or JFS2). Each JFS consists of a pool of page-size (4 KB) blocks. When data is to be written to a file, one or more additional blocks are allocated to that file. These blocks might not be contiguous with one another or with other blocks previously allocated to the file. A given file system can be defined as having a fragment size of less than 4 KB (512 bytes, 1 KB, 2 KB).

After installation, the system has one volume group (the rootvg volume group) consisting of a base set of logical volumes required to start the system and any other logical volumes 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, the menu-driven System Management Interface Tool (SMIT) interface, or Web-based System Manager.

Logical volume storage concepts

The logical volume (which can span physical volumes) is composed of logical partitions allocated onto physical partitions.

The following figure illustrates the relationships among the basic logical storage concepts.

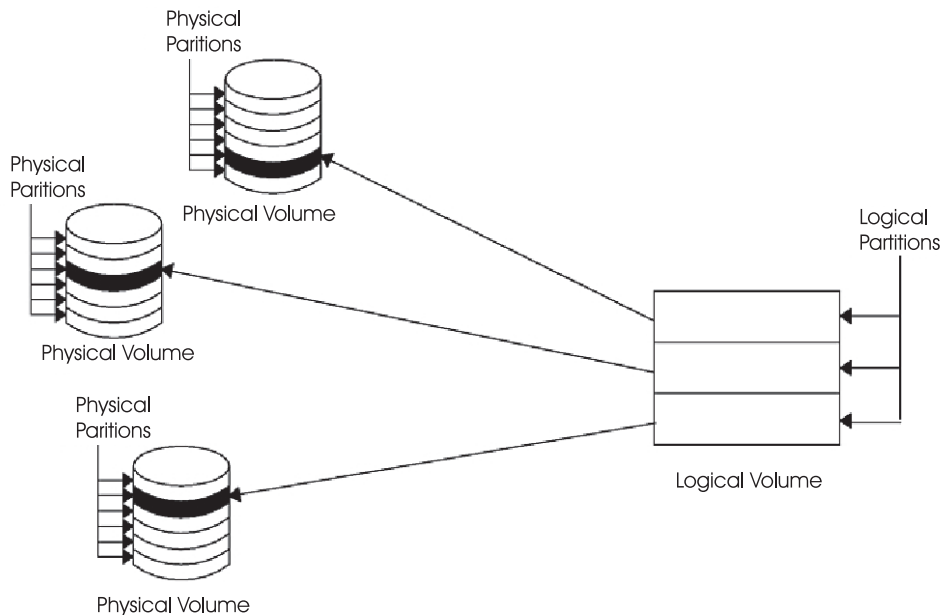


Figure 1. Volume Group. This illustration shows a volume group composed of three physical volumes with the maximum range specified. The logical volume (which can span physical volumes) is composed of logical partitions allocated onto physical partitions.

Physical volumes

A disk must be designated as a physical volume and be put into an available state before it can be assigned to a volume group.

A physical volume has certain configuration and identification information written on it. This information includes a physical volume identifier that is unique to the system.

In AIX 5.2 and later versions, the LVM can make use of the additional space that a redundant array of identical disks (RAID) can add to a logical unit number (LUN), by adding physical partitions to the physical volume associated with the LUN.

Volume groups

A *volume group* is a collection of 1 to 32 physical volumes of varying sizes and types.

A big volume group can have from 1 to 128 physical volumes. A scalable volume group can have up to 1024 physical volumes. A physical volume can belong to only one volume group per system; there can be up to 255 active volume groups.

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. For more information, see “Physical partitions” on page 27.

When you install the system, one volume group (the root volume group, called `rootvg`) is automatically created that contains the base set of logical volumes required to start the system, as well as any other logical volumes you specify to the installation script. The `rootvg` 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 user-defined volume groups. For example, the `rootvg` cannot be imported or exported. When performing a command or procedure on the `rootvg`, you must be familiar with its unique characteristics.

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

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

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

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

Physical partitions

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

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

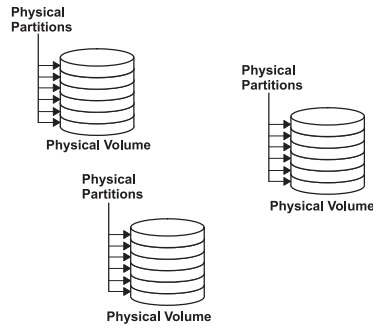


Figure 2. A Volume Group Containing Three Physical Volumes. This illustration shows three physical volumes, each with six physical partitions, within a single volume group.

Logical volumes

After you create a volume group, you can create logical volumes within that volume group.

A *logical volume*, although it can reside on noncontiguous physical partitions or even on more than one physical volume, appears to users and applications as a single, contiguous, extensible disk volume. You can create additional logical volumes with the **mkiv** 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 512 logical partitions, unless specified to be larger. The **chlv** command is used to override this limitation.

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

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

The system allows you to define up to 255 logical volumes per standard volume group (511 for a big volume group and 4095 for a scalable volume group), but the actual number you can define depends on the total amount of physical storage defined for that volume group and the size of the logical volumes you define.

Logical partitions

When you create a logical volume, you specify the number of *logical partitions* for the logical volume.

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

File systems

The logical volume defines allocation of disk space down to the physical-partition level. Finer levels of data management are accomplished by higher-level software components such as the Virtual Memory Manager or the file system. Therefore, the final step in the evolution of a disk is the creation of *file systems*.

You can create one file system per logical volume. To create a file system, use the **crfs** command. For more information on file systems, see Chapter 5, “File systems,” on page 93.

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 for a big volume group, 1024 for a scalable 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.

Limitations for Logical Storage Management

Category	Limit
Volume group	<ul style="list-style-type: none">• 255 volume groups for the 32 bit kernel• 4096 volume groups for the 64 bit kernel <p>Note: The device table on the 64 bit kernel restricts the number of active major numbers to 1024. Consequently, the number of active volume groups is restricted to less than 1024 volume groups.</p>
Physical volume	(MAXPVS/volume group factor) per volume group. MAXPVS is 32 for a standard volume group, 128 for a big volume group, and 1024 for a scalable volume group.
Physical partition	Normal and Big Volume groups: (1016 x volume group factor) per physical volume up to 1024 MB each in size. Scalable volume groups: 2097152 partitions up to 128 GB in size. There is no volume group factor for scalable volume groups.
Logical volume	MAXLVS per volume group, which is 255 for a standard volume group, 511 for a big volume group, and 4095 for a scalable volume group.

If you previously created a volume group before the 1016 physical partitions per physical volume restriction was enforced, stale partitions (no longer containing the most current data) in the volume group are not correctly tracked unless you convert the volume group to a supported state. You can convert the volume group with the **chvg -t** command. A suitable factor value is chosen by default to accommodate the largest disk in the volume group.

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

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

volume group, a factor of 2 decreases the maximum number of disks in the volume group to 64 (128/2). For a scalable volume group, a factor of 2 decreases the maximum number of disks in the volume group to 512 (1024/2).

LVM maintenance

Instructions for additional maintenance tasks are located later in this section.

You must have root authority to perform most of the following tasks. For your convenience, links to all the logical volume, physical volume, and volume group maintenance tasks are listed below.

LVM maintenance commands and fastpaths

The simplest tasks you might need when maintaining the entities that LVM controls (physical and logical volumes, volume groups, and file systems) are grouped within the following table.

Table 1. Managing Logical Volumes and Storage Tasks

Task	SMIT Fast Path	Command or File
Activate a volume group	smit varyonvg	
Add a fixed disk without data to existing volume group	smit extendvg	
Add a fixed disk without data to new volume group	smit mkvg	
Add a logical volume ^{Note 1}	smit mklv	
Add a volume group	smit mkvg	
Add and activate a new volume group	smit mkvg	
Change a logical volume to use data allocation	smit chlv1	
Change the name of a volume group ^{Note 2}	<ol style="list-style-type: none"> 1. smit varyoffvg 2. smit exportvg 3. smit importvg 4. smit mountfs 	<ol style="list-style-type: none"> 1. varyoffvg <i>OldVGName</i> 2. exportvg <i>OldVGName</i> 3. importvg <i>NewVGName</i> 4. mount all
Change a volume group to use automatic activation	smit chvg	
Change or set logical volume policies	smit chlv1	
Copy a logical volume to a new logical volume ^{Note 3}	smit cplv	
Copy a logical volume to an existing logical volume of the same size ^{Attn 1}	smit cplv	

Table 1. Managing Logical Volumes and Storage Tasks (continued)

Task	SMIT Fast Path	Command or File
Copy a logical volume to an existing logical volume of smaller size ^{Attn 1} <small>Note 3</small>	Do not use SMIT ^{Attn 2}	<ol style="list-style-type: none"> 1. Create logical volume. For example: mkiv -y hdiskN vg00 4 2. Create new file system on new logical volume. For example: crfs -v jfs -d hdiskN -m /doc -A yes 3. Mount file system. For example: mount /doc 4. Create directory at new mount point. For example: mkdir /doc/options 5. Transfer files system from source to destination logical volume. For example: cp -R /usr/adam/oldoptions/* \ /doc/options
Copy a logical volume to an existing logical volume of larger size ^{Attn 1}	smit cplv	
Deactivate a volume group	smit varyoffvg	
Enable write-verify and change scheduling policy	smit chlv1	
Increase the maximum size of a logical volume	smit chlv1	
Increase the size of a logical volume	smit extendlv	
List all logical volumes by volume group	smit lslv2	
List all physical volumes in system	smit lspv2	
List all volume groups	smit lsvg2	
List the status, logical volumes, or partitions of a physical volume	smit lspv	
List the contents of a volume group	smit lsvg1	
List a logical volume's status or mapping	smit lslv	
Mirror a logical volume with or without data allocation	smit mklvcopy	
Power off a removable disk	smit offdisk	Available with the hot-removability feature only
Power on a removable disk	smit ondisk	Available with the hot-removability feature only
Remove a disk with data from the operating system	smit exportvgrds	
Remove a disk without data from the operating system	smit reducevgrds	
Remove mirroring from a volume group	smit unmirrorvg	
Remove a volume group	smit reducevg2	
Reorganize a volume group	smit reorgvg	

Table 1. Managing Logical Volumes and Storage Tasks (continued)

Task	SMIT Fast Path	Command or File
Unconfigure and power off a disk	smit rmvdsk1 or smit rmvdsk then smit opendoor	

Attention:

1. Using this procedure to copy to an existing logical volume will overwrite any data on that volume without requesting user confirmation.
2. Do not use the SMIT procedure or the **cplv** command to copy a larger logical volume to a smaller one. Doing so results in a corrupted file system because some of the data (including the superblock) is not copied to the smaller logical volume.

Note:

1. After you create a logical volume, the state will be closed because no LVM structure is using that logical volume. It will remain closed until a file system has been mounted over the logical volume or the logical volume is opened for raw I/O. See also “Defining a raw logical volume for an application” on page 40.
2. You cannot change the name of, import, or export **rootvg**.
3. You must have enough direct access storage to duplicate a specific logical volume.

Adding disks while the system remains available

The following procedure describes how to turn on and configure a disk using the hot-removability feature, which lets you add disks without powering off the system.

You can add a disk for additional storage or to correct a disk failure. To remove a disk using the hot-removability feature, see “LVM maintenance” on page 30. This feature is only available on certain systems.

1. Install the disk in a free slot of the cabinet. For detailed information about the installation procedure, see the service guide for your machine.
2. Power on the new disk by typing the following fast path on the command line:

```
smit ondisk
```

At this point, the disk is added to the system but it is not yet usable. What you do next depends on whether the new disk contains data.

- If the disk has no data, add it as a physical volume to a volume group using one of the following:
 - To add the disk to an existing volume group, type the following fast path on the command line:
smit extendvg
 - To add the disk to a new volume group, type the following fast path on the command line:
smit mkvg
- If the disk contains data, import the data using the procedure in “Importing or exporting a volume group” on page 37.

Changing the name of a logical volume

The following procedure describes how to rename a logical volume without losing data on the logical volume.

In the following examples, the logical volume name is changed from lv00 to lv33.

1. Unmount all file systems associated with the logical volume, by typing:

```
umount /FSname
```

Where *FSname* is the full name of a file system.

Note:

- a. The **umount** command fails if the file system you are trying to unmount is currently being used. The **umount** command executes only if none of the file system's files are open and no user's current directory is on that device.
 - b. Another name for the **umount** command is **umount**. The names are interchangeable.
2. Rename the logical volume, by typing:

```
chlv -n NewLVname OldLVname
```

Where the **-n** flag specifies the new logical volume name (*NewLVname*) and *OldLVname* is the name you want to change. For example:

```
chlv -n lv33 lv00
```

Note: If you rename a JFS or JFS2 log, the system prompts you to run the **chfs** command on all file systems that use the renamed log device.

3. Remount the file systems you unmounted in step 1 on page 32 by typing:

```
mount /test1
```

At this point, the logical volume is renamed and available for use.

Copy of a logical volume to another physical volume

Depending on your needs, there are several ways to copy a logical volume to another physical volume while retaining file system integrity.

There are multiple methods for copying a logical volume or JFS to another physical volume. Choose the method that best serves your purposes.

Copying a logical volume:

The simplest method is to use the **cplv** command to copy the original logical volume and create a new logical volume on the destination physical volume.

1. Stop using the logical volume. Unmount the file system, if applicable, and stop any application that accesses the logical volume.
2. Select a physical volume that has the capacity to contain all of the data in the original logical volume.

Attention: If you copy from a larger logical volume containing data to a smaller one, you can corrupt your file system because some data (including the superblock) might be lost.
3. Copy the original logical volume (in this example, it is named **lv00**) and create the new one, using the following command:

Note: The following **cplv** command fails if it creates a new logical volume and the volume group is varied on in concurrent mode.

```
cplv lv00
```

4. Mount the file systems, if applicable, and restart applications to begin using the logical volume.

At this point, the logical volume copy is usable.

Copying a logical volume while original logical volume remains usable:

If your environment requires continued use of the original logical volume, you can use the **splitvcopy** command to copy the contents, as shown in the following example.

To successfully complete this procedure in a concurrent volume group environment, AIX 4.3.2 or later must be installed on all concurrent nodes.

1. Mirror the logical volume, using the following SMIT fast path:

```
smit mklvcopy
```

2. Stop using the logical volume. Unmount the file system, if applicable, and stop or put into quiescent mode any application that accesses the logical volume.

Attention: The next step uses the **splitlvcopy** command. Always close logical volumes before splitting them and unmount any contained file systems before using this command. Splitting an open logical volume can corrupt your file systems and cause you to lose consistency between the original logical volume and the copy if the logical volume is accessed simultaneously by multiple processes.

3. With root authority, copy the original logical volume (`oldlv`) to the new logical volume (`newlv`) using the following command:

```
splitlvcopy -y newlv oldlv
```

The **-y** flag designates the new logical volume name. If the `oldlv` volume does not have a logical volume control block, the **splitlvcopy** command completes successfully but generates a message that the `newlv` volume has been created without a logical volume control block.

4. Mount the file systems, if applicable, and restart applications to begin using the logical volume.

At this point, the logical volume copy is usable.

Copying a raw logical volume to another physical volume:

To copy a raw logical volume to another physical volume, perform the following steps.

To successfully complete this procedure in a concurrent volume group environment, AIX 4.3.2 or later must be installed on all concurrent nodes.

1. Create a mirrored copy of the logical volume on a new physical volume in the volume group using the following command:

```
mk1vcopy LogVol_name 2 new_PhysVol_name
```

2. Synchronize the partitions in the new mirror copy using the following command:

```
syncvg -l LogVol_name
```

3. Remove the copy of the logical volume from the physical volume using the following command:

```
rm1vcopy LogVol_name 1 old_PhysVol_name
```

At this point, the raw logical volume copy is usable.

Creating a file system log on a dedicated disk for a user-defined volume group:

A JFS or JFS2 *file system log* is a formatted list of file system transaction records. The log ensures file system integrity (but not necessarily data integrity) in case the system goes down before transactions have been completed.

A dedicated disk is created on `hd8` for `rootvg` when the system is installed. The following procedure helps you create a JFS log on a separate disk for other volume groups. When you create a JFS2 log, the procedure requires the following changes:

- The log device type is **jfs2log**.
- The **logform** command requires a **-V jfs2** option to specify a JFS2 log device.
- The **crfs** commands must specify **jfs2** instead of **jfs**.

Creating a file system log file for user-defined volume groups can improve performance under certain conditions, for example, if you have an NFS server and you want the transactions for this server to be processed without competition from other processes.

To create a log file for user-defined volume groups, the easiest way is to use the Web-based System Manager wizard, as follows:

1. If Web-based System Manager is not already running, with root authority, type `wsm` on the command line.

2. Select a host name.
3. Select the **Volumes** container.
4. Select the **Logical Volumes** container.
5. In the Volumes menu, select **New Logical Volume** (Wizard). The wizard will guide you through the procedure. Online help is available if you need it.

Alternatively, you can use the following procedure, which creates a volume group (fsvg1) with two physical volumes (hdisk1 and hdisk2). The file system is on hdisk2 (a 256-MB file system mounted at **/u/myfs**) and the log is on hdisk1. By default, a JFS log size is 4 MB. You can place little-used programs, for example, **/blv**, on the same physical volume as the log without impacting performance.

The following instructions explain how to create a JFS log for a user-defined volume group using SMIT and the command line interface:

1. Add the new volume group (in this example, fsvg1) using the SMIT fast path:

```
smit mkvg
```

2. Add a new logical volume to this volume group using the SMIT fast path:

```
smit mklv
```

3. On the **Add a Logical Volume** screen, add your data to the following fields. For example:

```
Logical Volumes NAME           fsvg1log
Number of LOGICAL PARTITIONS   1
PHYSICAL VOLUME names         hdisk1
Logical volume TYPE           jfslog
POSITION on Physical Volume    center
```

4. After you set the fields, press Enter to accept your changes and exit SMIT.
5. Type the following on a command line:

```
/usr/sbin/logform /dev/fsvg1log
```

6. When you receive the following prompt, type **y** and press Enter:

```
Destroy /dev/fsvg1log
```

Despite the wording in this prompt, nothing is destroyed. When you respond **y** to this prompt, the system formats the logical volume for the JFS log so that it can record file-system transactions.

7. Add another logical volume using the following SMIT fast path:

```
smit mklv
```

8. Type the name of the same volume group as you used in step 2 (fsvg1 in this example). In the Logical Volumes screen, add your data to the following fields. Remember to designate a different physical volume for this logical volume than you did in step 3. For example:

```
Logical Volumes NAME           fslv1
Number of LOGICAL PARTITIONS   64
PHYSICAL VOLUME names         hdisk2
Logical volume TYPE           jfs
```

After you set the fields, press Enter to accept your changes and exit SMIT.

9. Add a file system to the new logical volume, designate the log, and mount the new file system, using the following sequence of commands:

```
crfs -v jfs -d LogVolName -m FileSysName -a logname=FSLogPath
```

```
mount FileSysName
```

Where *LogVolName* is the name of the logical volume you created in step 2; *FileSysName* is the name of the file system you want to mount on this logical volume; and *FSLogPath* is the name of the volume group you created in step 2. For example:

```
crfs -v jfs -d fslv1 -m /u/myfs -a logname=/dev/fsvg1log
mount /u/myfs
```

10. To verify that you have set up the file system and log correctly, type the following command (substituting your volume group name):

```
lsvg -l fsvg1
```

The output shows both logical volumes you created, with their file system types, as in the following example:

LV NAME	TYPE	...
/dev/fsvg1log	jfslog	...
fslv1	jfs	...

At this point, you have created a volume group containing at least two logical volumes on separate physical volumes, and one of those logical volumes contains the file system log.

Enabling hot spare disk support for an existing volume group:

The following steps use Web-based System Manager to enable hot spare disk support for an existing volume.

1. Start Web-based System Manager (if not already running) by typing `wsm` on the command line.
2. Select the **Volumes** container.
3. Select the **Volume Groups** container.
4. Select the name of your target volume group, and choose **Properties** from the **Selected** menu.
5. Select the **Hot Spare Disk Support** tab and check beside **Enable hot spare disk support**.
6. Select the **Physical Volumes** tab to add available physical volumes to the volume group as hot spare disks.

At this point, your mirrored volume group has one or more disks designated as spares. If your system detects a failing disk, depending on the options you selected, the data on the failing disk can be migrated to a spare disk without interruption to use or availability.

Enabling hot spare disk support while creating a new volume group:

The following steps use Web-based System Manager to enable hot spare disk support while you are creating a new volume group.

1. Start Web-based System Manager (if not already running) by typing `wsm` on the command line.
2. Select the **Volumes** container.
3. Select the **Volume Groups** container.
4. From the Volumes menu, select **New→Volume Group (Advanced Method)**. The subsequent panels let you choose physical volumes and their sizes, enable hot spare disk support, select unused physical volumes to assign as hot spares, then set the migration characteristics for your hot spare disk or your hot spare disk pool.

At this point, your system recognizes a new mirrored volume group with one or more disks designated as spares. If your system detects a failing disk, depending on the options you selected, the data on the failing disk can be migrated to a spare disk without interruption to use or availability.

Enablement and configuration of hot spot reporting: Beginning with AIX 5.1, you can identify hot spot problems with your logical volumes and remedy those problems without interrupting the use of your system. A *hot-spot* problem occurs when some of the logical partitions on your disk have so much disk I/O that your system performance noticeably suffers.

The following procedures use Web-based System Manager to enable host spot reporting and manage the result.

Enabling hot spot reporting at the volume group level:

The following steps use Web-based System Manager to enable hot spot reporting at the volume group level.

1. Start Web-based System Manager (if not already running) by typing `wsm` on the command line.
2. Select the **Volumes** container.
3. Select the **Volume Groups** container.
4. Select the name of your target volume group, and choose **Hot Spot Reporting...** from the Selected menu.
5. Check beside **Enable hot spot reporting** and **Restart the Statistics Counters**.

At this point, the hot spot feature is enabled. Use the pull-down or pop-up menu in Web-based System Manager to access the **Manage Hot Spots...** Sequential dialog. In the subsequent panels, you can define your reporting and statistics, display your statistics, select logical partitions to migrate, specify the destination physical partition, and verify the information before committing your changes.

Enabling hot spot reporting at the logical volume level:

The following steps use Web-based System Manager to enable hot spot reporting at the logical volume level so you can avoid enabling it for an entire volume group.

1. Start Web-based System Manager (if not already running) by typing `wsm` on the command line.
2. Select the **Volumes** container.
3. Select the **Logical Volumes** container.
4. Select the name of your target logical volume and choose **Hot Spot Reporting...** from the Selected menu.
5. Check beside **Enable hot spot reporting** and **Restart the Statistics Counters**.

At this point, the hot spot feature is enabled. Use the pull-down or pop-up menu in Web-based System Manager to access the **Manage Hot Spots...** Sequential dialog. In the subsequent panels, you can define your reporting and statistics, display your statistics, select logical partitions to migrate, specify the destination physical partition, and verify the information before committing your changes.

Importing or exporting a volume group:

The following table explains how to use import and export to move a user-defined volume group from one system to another. (The `rootvg` volume group cannot be exported or imported.)

The export procedure removes the definition of a volume group from a system. The import procedure serves to introduce the volume group to its new system.

You can also use the import procedure to reintroduce a volume group to the system when it once was associated with the system but had been exported. You can also use import and export to add a physical volume that contains data to a volume group by putting the disk to be added in its own volume group.

Attention: The `importvg` command changes the name of an imported logical volume if a logical volume of that name already exists on the new system. If the `importvg` command must rename a logical volume, it prints an error message to standard error. When there are no conflicts, the `importvg` command also creates file mount points and entries in the `/etc/filesystems` file.

Import and Export Volume Group Tasks		
Task	SMIT Fast Path	Command or File
Import a volume group	smit importvg	
Export a volume group	<ol style="list-style-type: none"> 1. Unmount files systems on logical volumes in the volume group: smit umntdsk 2. Vary off the volume group: smit varyoffvg 3. Export the volume group: smit exportvg 	

Attention: A volume group that has a paging space volume on it cannot be exported while the paging space is active. Before exporting a volume group with an active paging space, ensure that the paging space is not activated automatically at system initialization by typing the following command:

```
chps -a n paging_space name
```

Then, reboot the system so that the paging space is inactive.

Migrating the contents of a physical volume:

To move the physical partitions belonging to one or more specified logical volumes from one physical volume to one or more other physical volumes in a volume group, use the following instructions. You can also use this procedure to move data from a failing disk before replacing or repairing the failing disk. This procedure can be used on physical volumes in either the root volume group or a user-defined volume group.

Attention: When the boot logical volume is migrated from a physical volume, the boot record on the source must be cleared or it could cause a system hang. When you execute the **bosboot** command, you must also execute the **chpv -c** command described in step 4 of the following procedure.

1. If you want to migrate the data to a new disk, do the following steps. Otherwise, continue with step 2.
 - a. Check that the disk is recognizable by the system and available by typing:

```
lsdev -Cc disk
```

The output resembles the following:

```
hdisk0 Available 10-60-00-8,0 16 Bit LVD SCSI Disk Drive
hdisk1 Available 10-60-00-9,0 16 Bit LVD SCSI Disk Drive
hdisk2 Available 10-60-00-11,0 16 Bit LVD SCSI Disk Drive
```

- b. If the disk is listed and in the available state, check that it does not belong to another volume group by typing:

```
lspv
```

The output looks similar to the following:

```
hdisk0      0004234583aa7879      rootvg      active
hdisk1      00042345e05603c1      none        active
hdisk2      00083772caa7896e      imagesvg    active
```

In the example, `hdisk1` can be used as a destination disk because the third field shows that it is not being used by a volume group.

If the new disk is not listed or unavailable, refer to “Configuring a disk” on page 60 or Chapter 3, “Logical volume storage,” on page 25.

- c. Add the new disk to the volume group by typing:

```
extendvg VGName diskname
```

Where *VGName* is the name of your volume group and *diskname* is the name of the new disk. In the example shown in the previous step, *diskname* would be replaced by *hdisk1*.

2. The source and destination physical volumes must be in the same volume group. To determine whether both physical volumes are in the volume group, type:

```
lsvg -p VGname
```

Where *VGname* is the name of your volume group. The output for a root volume group looks similar to the following:

```
rootvg:
PV_NAME      PV STATE      TOTAL PPs   FREE PPs   FREE DISTRIBUTION
hdisk0       active        542         85         00..00..00..26..59
hdisk1       active        542         306        00..00..00..00..06
```

Note the number of FREE PPs.

3. Check that you have enough room on the target disk for the source that you want to move:

- a. Determine the number of physical partitions on the source disk by typing:

```
lspv SourceDiskName | grep "USED PPs"
```

Where *SourceDiskName* is of the name of the source disk, for example, *hdisk0*. The output looks similar to the following:

```
USED PPs:      159 (636 megabytes)
```

In this example, you need 159 FREE PPs on the destination disk to successfully complete the migration.

- b. Compare the number of USED PPs from the source disk with the number of FREE PPs on the destination disk or disks (step 2). If the number of FREE PPs is larger than the number of USED PPs, you have enough space for the migration.
4. Follow this step only if you are migrating data from a disk in the rootvg volume group. If you are migrating data from a disk in a user-defined volume group, proceed to step 5.

Check to see if the boot logical volume (**hd5**) is on the source disk by typing:

```
lspv -l SourceDiskNumber | grep hd5
```

If you get no output, the boot logical volume is not located on the source disk. Continue to step 5.

If you get output similar to the following:

```
hd5          2  2  02..00..00..00..00  /blv
```

then run the following command:

```
migratepv -l hd5 SourceDiskName DestinationDiskName
```

You will receive a message warning you to perform the **bosboot** command on the destination disk. You must also perform a **mkboot -c** command to clear the boot record on the source. Type the following sequence of commands:

```
bosboot -a -d /dev/DestinationDiskName
bootlist -m normal DestinationDiskName
mkboot -c -d /dev/SourceDiskName
```

5. Migrate your data by typing the following SMIT fast path:

```
smit migratepv
```

6. List the physical volumes, and select the source physical volume you examined previously.

7. Go to the **DESTINATION** physical volume field. If you accept the default, all the physical volumes in the volume group are available for the transfer. Otherwise, select one or more disks with adequate space for the partitions you are moving (from step 4).

8. If you wish, go to the Move only data belonging to this **LOGICAL VOLUME** field, and list and select a logical volume. You move only the physical partitions allocated to the logical volume specified that are located on the physical volume selected as the source physical volume.
9. Press Enter to move the physical partitions.

At this point, the data now resides on the new (destination) disk. The original (source) disk, however, remains in the volume group. If the disk is still reliable, you could continue to use it as a hot spare disk (see “Enabling hot spare disk support for an existing volume group” on page 36 and “Enabling hot spare disk support while creating a new volume group” on page 36). Especially when a disk is failing, it is advisable to do the following steps:

1. To remove the source disk from the volume group, type:

```
reducevg VGName SourceDiskName
```
2. To physically remove the source disk from the system, type:

```
rmdev -l SourceDiskName -d
```

Defining a raw logical volume for an application

A *raw logical volume* is an area of physical and logical disk space that is under the direct control of an application, such as a database or a partition, rather than under the direct control of the operating system or a file system.

Bypassing the file system can yield better performance from the controlling application, especially from database applications. The amount of improvement, however, depends on factors such as the size of a database or the application’s driver.

Note: You will need to provide the application with the character or block special device file for the new raw logical volume, as appropriate. The application will link to this device file when it attempts opens, reads, writes, and so on.

Attention: Each logical volume has a logical-volume control block (LVCB) located in the first 512 bytes. Data begins in the second 512-byte block. In a raw logical volume, the LVCB is not protected. If an application overwrites the LVCB, commands that normally update the LVCB will fail and generate a message. Although the logical volume might continue to operate correctly and the overwrite can be an allowable event, overwriting the LVCB is not recommended.

The following instructions use SMIT and the command line interface to define a raw logical volume. You can also use the **Create a new logical volume** wizard in Web-based System Manager (select **Volumes** → **Overview and Tasks** → **Create a new logical volume**). To define a raw logical volume within the wizard, accept the default use, **applications and data**, from its first selection screen. Online help is available if you need it.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. With root authority, find the free physical partitions on which you can create the raw logical volume by typing the following SMIT fast path:

```
smit lspv
```
2. Select a disk.
3. Accept the default in the second dialog (status) and click **OK**.
4. Multiply the value in the **FREE PPs** field by the value in the **PP SIZE** field to get the total number of megabytes available for a raw logical volume on the selected disk. If the amount of free space is not adequate, select a different disk until you find one that has enough available free space.
5. Exit SMIT.

6. Use the **mklv** command to create the raw logical volume. The following command creates a raw logical volume named `lvdb2003` in the `db2vg` volume group using 38 4-MB physical partitions:

```
mklv -y lvdb2003 db2vg 38
```

Use the **-y** flag to provide a name for the logical volume instead of using a system-generated name.

At this point, the raw logical volume is created. If you list the contents of your volume group, a raw logical volume is shown with the default type, which is `jfs`. This type entry for a logical volume is simply a label. It does not indicate a file system is mounted for your raw logical volume.

Consult your application's instructions on how to open `/dev/rawLVname` and how to use this raw space.

For more information

- The **mklv** command description in *AIX 5L Version 5.3 Commands Reference*
- *AIX Logical Volume Manager from A to Z: Introduction and Concepts*, an IBM Redbook
- *AIX Storage Management*, an IBM Redbook

Mirroring a volume group

These scenarios explain how to mirror a normal volume group.

If you want to mirror the root volume group (`rootvg`), see “Mirroring the root volume group.”

The following instructions show you how to mirror a root volume group using the System Management Interface Tool (SMIT). You can also use Web-based System Manager

(select a volume group in the **Volumes** container, then choose **Mirror** from the **Selected** menu). Experienced administrators can use the **mirrorvg** command.

1. With root authority, add a disk to the volume group using the following SMIT fast path:

```
smit extendvg
```

2. Mirror the volume group onto the new disk by typing the following SMIT fast path:

```
smit mirrorvg
```

3. In the first panel, select a volume group for mirroring.
4. In the second panel, you can define mirroring options or accept defaults. Online help is available if you need it.

Note: When you complete the SMIT panels and click OK or exit, the underlying command can take a significant amount of time to complete. The length of time is affected by error checking, the size and number of logical volumes in the volume group, and the time it takes to synchronize the newly mirrored logical volumes.

At this point, all changes to the logical volumes will be mirrored as you specified in the SMIT panels.

Mirroring the root volume group

The following scenario explains how to mirror the root volume group (`rootvg`).

Note:

1. Mirroring the root volume group requires advanced system administration experience. If not done correctly, you can cause your system to be unbootable.
2. Mirrored dump devices are supported in AIX 4.3.3 or later.

In the following scenario, the `rootvg` is contained on `hdisk01`, and the mirror is being made to a disk called `hdisk11`:

1. Check that `hdisk11` is supported by AIX as a boot device:

```
bootinfo -B hdisk11
```

If this command returns a value of 1, the selected disk is bootable by AIX. Any other value indicates that `hdisk11` is not a candidate for rootvg mirroring.

2. Extend rootvg to include `hdisk11`, using the following command:

```
extendvg rootvg hdisk11
```

If you receive the following error messages:

```
0516-050 Not enough descriptor space left in this volume group, Either try
adding a smaller PV or use another volume group.
```

or a message similar to:

```
0516-1162 extendvg: Warning, The Physical Partition size of 16 requires the
creation of 1084 partitions for hdisk11. The limitation for volume group
rootvg is 1016 physical partitions per physical volume. Use chvg command with
the -t option to attempt to change the maximum physical partitions per Physical
Volume for this volume group.
```

You have the following options:

- Mirror the rootvg to an empty disk that already belongs to the rootvg.
- Use a smaller disk.
- Change the maximum number of partitions supported by the rootvg, using the following procedure:
 - a. Check the message for the number of physical partitions needed for the destination disk and the maximum number currently supported by rootvg.
 - b. Use the **chvg -t** command to multiply the maximum number of partitions currently allowed in rootvg (in the above example, 1016) to a number that is larger than the physical partitions needed for the destination disk (in the above example, 1084). For example:

```
chvg -t 2 rootvg
```
 - c. Reissue the **extendvg** command at the beginning of step 2.

3. Mirror the rootvg, using the exact mapping option, as shown in the following command:

```
mirrorvg -m rootvg hdisk11
```

This command will turn off quorum when the volume group is rootvg. If you do not use the exact mapping option, you must verify that the new copy of the boot logical volume, `hd5`, is made of contiguous partitions.

4. Initialize all boot records and devices, using the following command:

```
bosboot -a
```

5. Initialize the boot list with the following command:

```
bootlist -m normal hdisk01 hdisk11
```

Note:

- a. Even though the **bootlist** command identifies `hdisk11` as an alternate boot disk, it cannot guarantee the system will use `hdisk11` as the boot device if `hdisk01` fails. In such case, you might have to boot from the product media, select **maintenance**, and reissue the **bootlist** command without naming the failed disk.
- b. If your hardware model does not support the **bootlist** command, you can still mirror the rootvg, but you must actively select the alternate boot disk when the original disk is unavailable.

Unmirroring the root volume group

You can unmirror the root volume group.

Attention: Unmirroring the root volume group requires advanced system administration experience. If not done correctly, your system can become unbootable.

In the following scenario, the root volume group is on `hdisk01` and mirrored onto `hdisk11`. This example removes the mirror on `hdisk11`. The procedure is the same, regardless of which disk you booted to last.

1. Use the following command to unmirror the root volume group on `hdisk11`:

```
unmirrorvg rootvg hdisk11
```

The **unmirrorvg** command turns quorum back on for the root volume group.

2. Use the following command to reduce the disk out of the root volume group:

```
reducevg rootvg hdisk11
```

3. Use the following command to reinitialize the boot record of the remaining disk:

```
bosboot -a -d /dev/hdisk01
```

4. Use the following command to modify the boot list in order to remove the unmirrored disk from the list:

```
bootlist -m normal hdisk01
```

The disk is unmirrored.

Disk removal while the system remains available

The following procedure describes how to remove a disk using the hot-removability feature, which lets you remove the disk without turning the system off. This feature is only available on certain systems.

Hot removability is useful when you want to:

- Remove a disk that contains data in a separate non-rootvg volume group for security or maintenance purposes.
- Permanently remove a disk from a volume group.
- Correct a disk failure.

Removing a disk with data

Use this procedure to remove a disk that contains data without turning the system off.

The disk you are removing must be in a separate non-rootvg volume group. Use this procedure when you want to move a disk to another system.

1. To list the volume group associated with the disk you want to remove, type:

```
smit lspv
```

Your output looks similar to the following:

```
PHYSICAL VOLUME:   hdisk2                VOLUME GROUP:   imagesvg
PV IDENTIFIER:    00083772caa7896e VG IDENTIFIER    0004234500004c00000000e9b5cac262
PV STATE:        active
STALE PARTITIONS: 0                ALLOCATABLE:    yes
PP SIZE:         16 megabyte(s)          LOGICAL VOLUMES: 5
TOTAL PPs:       542 (8672 megabytes)    VG DESCRIPTORS: 2
FREE PPs:        19 (304 megabytes)      HOT SPARE:      no
USED PPs:        523 (8368 megabytes)
FREE DISTRIBUTION: 00..00..00..00..19
USED DISTRIBUTION: 109..108..108..108..90
```

The name of the volume group is listed in the **VOLUME GROUP** field. In this example, the volume group is `imagesvg`.

2. To verify that the disk is in a separate non-rootvg volume group, type:

```
smit lsvg
```

Then select the volume group associated with your disk (in this example, imagesvg). Your output looks similar to the following:

VOLUME GROUP:	imagesvg	VG IDENTIFIER:	0004234500004c00000000e9b5cac262
VG STATE:	active	PP SIZE:	16 megabyte(s)
VG PERMISSION:	read/write	TOTAL PPs:	542 (8672 megabytes)
MAX LVs:	256	FREE PPs:	19 (304 megabytes)
LVs:	5	USED PPs:	523 (8368 megabytes)
OPEN LVs:	4	QUORUM:	2
TOTAL PVs:	1	VG DESCRIPTORS:	2
STALE PVs:	0	STALE PPs:	0
ACTIVE PVs:	1	AUTO ON:	yes
MAX PPs per PV:	1016	MAX PVs:	32
LTG size:	128 kilobyte(s)	AUTO SYNC:	no
HOT SPARE:	no		

In this example, the TOTAL PVs field indicates there is only one physical volume associated with imagesvg. Because all data in this volume group is contained on hdisk2, hdisk2 can be removed using this procedure.

3. To unmount any file systems on the logical volumes on the disk, type:

```
smit umountfs
```

4. To deactivate and export the volume group in which the disk resides, unconfigure the disk and turn it off, type:

```
smit exportvgrds
```

When the procedure completes, the system displays a message indicating the cabinet number and disk number of the disk to be removed. If the disk is placed at the front side of the cabinet, the disk shutter automatically opens.

5. Look at the LED display for the disk you want to remove. Ensure the yellow LED is off (not lit).
6. Physically remove the disk. For more information about the removal procedure, see the service guide for your machine.

At this point, the disk is physically and logically removed from your system. If you are permanently removing this disk, this procedure is completed. You can also do one of the following:

- Import the removed disk to another system. See “Importing or exporting a volume group” on page 37.
- Replace the removed disk with a new one. See “Adding disks while the system remains available” on page 32.

Removing a disk without data

The following procedure describes how to remove a disk that contains either no data or no data that you want to keep.

Attention: The following procedure erases any data that resides on the disk.

1. To unmount any file systems on the logical volumes on the disk, type:

```
smit umountfs
```

2. To deactivate and export any volume group in which the disk resides, unconfigure the disk and turn it off, type:

```
smit exportvgrds
```

When the procedure completes, the system displays a message indicating the cabinet number and disk number of the disk to be removed. If the disk is placed at the front side of the cabinet, the disk shutter automatically opens.

3. Look at the LED display for the disk you want to remove. Ensure the yellow LED is off (not lit).

4. Physically remove the disk. For more information about the removal procedure, see the service guide for your machine.

At this point, the disk is physically and logically removed from your system. If you are permanently removing this disk, this procedure is completed. If you want to replace the removed disk with a new one, see “Adding disks while the system remains available” on page 32.

Removing a logical volume by removing the file system

The following procedure explains how to remove a JFS or JFS2 file system, its associated logical volume, its associated stanza in the `/etc/filesystems` file, and, optionally, the mount point (directory) where the file system is mounted.

Attention: When you remove a file system, you destroy all data in the specified file systems and logical volume.

If you want to remove a logical volume with a different type of file system mounted on it or a logical volume that does not contain a file system, refer to “Removing a logical volume only.”

To remove a journaled file system through Web-based System Manager, use the following procedure:

1. If Web-based System Manager is not already running, with root authority type `wsm` on the command line.
2. Select a host name.
3. Select the **File Systems** container.
4. Select the **Journaled File Systems** container.
5. Select the file system you want to remove.
6. From the **Selected** menu, select **Unmount**.
7. From the **Selected** menu, select **Delete**.

To remove a journaled file system through SMIT, use the following procedure:

1. Unmount the file system that resides on the logical volume with a command similar to the following example:

```
umount /adam/usr/local
```

Note: You cannot use the `umount` command on a device in use. A device is in use if any file is open for any reason or if a user’s current directory is on that device.

2. To remove the file system, type the following fast path:

```
smit rmfs
```
3. Select the name of the file system you want to remove.
4. Go to the **Remove Mount Point** field and toggle to your preference. If you select **yes**, the underlying command will also remove the mount point (directory) where the file system is mounted (if the directory is empty).
5. Press Enter to remove the file system. SMIT prompts you to confirm whether you want to remove the file system.
6. Confirm you want to remove the file system. SMIT displays a message when the file system has been removed successfully.

At this point, the file system, its data, and its associated logical volume are completely removed from your system.

Removing a logical volume only

Use this procedure to remove a logical volume with a different type of file system mounted on it or a logical volume that does not contain a file system.

Attention: Removing a logical volume destroys all data in the specified file systems and logical volume.

The following procedures explain how to remove a logical volume and any associated file system. You can use this procedure to remove a non-JFS file system or a logical volume that does not contain a file system. After the following procedures describe how to remove a logical volume, they describe how to remove any non-JFS file system's stanza in the `/etc/filesystems` file.

To remove a logical volume through Web-based System Manager, use the following procedure:

1. If Web-based System Manager is not already running, with root authority, type `wsm` on the command line.
2. Select a host name.
3. If the logical volume does not contain a file system, skip to step 10.
4. Select the **File Systems** container.
5. Select the container for the appropriate file system type.
6. Select the file system you want to unmount.
7. From the **Selected** menu, select **Unmount**.
8. Select the appropriate file system container in the navigation area to list its file systems.
9. Note the logical volume name of the system you want to remove.
10. Select the **Volumes** container.
11. Select the **Logical Volumes** container.
12. Select the logical volume you want to remove.
13. From the Selected menu, select **Delete**.

To remove a logical volume through SMIT, use the following procedure:

1. If the logical volume does not contain a file system, skip to step 4.
2. Unmount all file systems associated with the logical volume by typing:

```
umount /FSname
```

Where `/FSname` is the full path name of a file system.

Note:

- a. The **umount** command fails if the file system you are trying to **umount** is currently being used. The **umount** command executes only if none of the file system's files are open and no user's current directory is on that device.
 - b. Another name for the **umount** command is **umount**. The names are interchangeable.
3. To list information you need to know about your file systems, type the following fast path:

```
smit lsfs
```

The following is a partial listing:

Name	Nodename	Mount Pt	...
/dev/hd3	--	/tmp	...
/dev/locallv	--	/adam/usr/local	...

Assuming standard naming conventions for the second listed item, the file system is named **/adam/usr/local** and the logical volume is `locallv`. To verify this, type the following fast path:

```
smit ls1v2
```

The following is a partial listing:

```

imagesvg:
LV NAME          TYPE      LPs   PPs   PVs   LV STATE   MOUNT POINT
hd3              jfs      4     4     1     open/syncd /tmp
locallv         mine     4     4     1     closed/syncd /adam/usr/local

```

- To remove the logical volume, type the following fast path on the command line:

```
smit rmlv
```

- Select the name of the logical volume you want to remove.
- Go to the **Remove Mount Point** field and toggle to your preference. If you select **yes**, the underlying command will also remove the mount point (directory) where the file system is mounted (if any, and if that directory is empty).
- Press Enter to remove the logical volume. SMIT prompts you to confirm whether you want to remove the logical volume.
- Confirm you want to remove the logical volume. SMIT displays a message when the logical volume has been removed successfully.
- If the logical volume had a non-JFS file system mounted on it, remove the file system and its associated stanza in the **/etc/filesystems** file, as shown in the following example:

```
rmfs /adam/usr/local
```

Or, you can use the file system name as follows:

```
rmfs /dev/locallv
```

At this point, the logical volume is removed. If the logical volume contained a non-JFS file system, that system's stanza has also been removed from the **/etc/filesystems** file.

Resizing a RAID volume group

In AIX 5.2 and later versions, on systems that use a redundant array of independent disks (RAID), **chvg** and **chpv** command options provide the ability to add a disk to the RAID group and grow the size of the physical volume that LVM uses without interruptions to the use or availability of the system.

Note:

- This feature is not available while the volume group is activated in classic or in enhanced concurrent mode.
- The rootvg volume group cannot be resized using the following procedure.
- A volume group with an active paging space cannot be resized using the following procedure.

The size of all disks in a volume group is automatically examined when the volume group is activated (varyon). If growth is detected, the system generates an informational message.

The following procedure describes how to grow disks in a RAID environment:

- To check for disk growth and resize if needed, type the following command:

```
chvg -g VGname
```

Where *VGname* is the name of your volume group. This command examines all disks in the volume group. If any have grown in size, it attempts to add physical partitions to the physical volume. If necessary, it will determine the appropriate 1016 limit multiplier and convert the volume group to a big volume group.

- To turn off LVM bad block relocation for the volume group, type the following command:

```
chvg -b ny VGname
```

Where *VGname* is the name of your volume group.

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 “Logical volume strategy” on page 49 for more information. Volume group size can also play a part.

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

When to create separate volume groups

There are several reasons why you might want to organize physical volumes into volume groups separate from rootvg.

- For safer and easier maintenance.
 - Operating system updates, reinstallations, and crash recoveries are safer because you can separate user file systems from the operating system so that user files are not jeopardized during these operations.
 - Maintenance is easier because you can update or reinstall the operating system without having to restore user data. For example, before updating, you can remove a user-defined volume group from the system by unmounting its file systems. Deactivate it using the **varyoffvg** command, then export the group using the **exportvg** command. After updating the system software, you can reintroduce the user-defined volume group using the **importvg** command, then remount its file systems.
- For different physical-partition sizes. All physical volumes within the same volume group must have the same physical partition size. To have physical volumes with different physical partition sizes, place each size in a separate volume group.
- When different quorum characteristics are required. If you have a file system for which you want to create a nonquorum volume group, maintain a separate volume group for that data; all of the other file systems should remain in volume groups operating under a quorum.
- 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).

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 requirements, see “Vary-on process” on page 73. 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 “Conversion of a volume group to nonquorum status” on page 74. 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 must also plan for room to mirror the data. Keep in mind that you can only mirror and move data between disks that are in the same volume group. If the site uses large file systems, finding disk space on which to mirror could become a problem at a later time. Be aware of the implications on availability of inter-disk settings for logical volume copies (see “Inter-disk settings for logical volume copies” on page 54) and intra-disk allocation (see “Intra-disk allocation policies for each logical volume” on page 55) for a logical volume.

High availability in case of adapter or power supply failure

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

- Use two adapters, located in the same or different chassis. Locating the adapters in different chassis protects against losing both adapters if there is a power supply failure in one chassis.
- Use two adapters, attaching at least one disk to each adapter. This protects against a failure at either adapter (or power supply if adapters are in separate cabinets) by still maintaining a quorum in the volume group, assuming *cross-mirroring* (copies for a logical partition cannot share the same physical volume) between the logical volumes on disk A (adapter A) and the logical volumes on disk B (adapter B). This means that you copy the logical volumes that reside on the disks attached to adapter A to the disks that reside on adapter B and also that you copy the logical volumes that reside on the disks attached to adapter B to the disks that reside on adapter A as well.
- Configure all disks from both adapters into the same volume group. This ensures that at least one logical volume copy remains intact in case an adapter fails, or, if cabinets are separate, in case a power supply fails.
- Make the volume group a nonquorum volume group. This allows the volume group to remain active as long as one Volume Group Descriptor Area (VGDA) is accessible on any disk in the volume group. See “Conversion of a volume group to nonquorum status” on page 74 for more information.
- If there are two disks in the volume group, implement cross-mirroring between the adapters. If more than one disk is available on each adapter, implement double-mirroring. In that case, you create a mirrored copy on a disk that uses the same adapter and one on a disk using a different adapter.

Logical volume strategy

The policies described here 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 access data even if its associated disk has failed or is inaccessible. The data might remain accessible through copies of the data that are made and maintained on separate disks and adapters during normal system operation. Techniques such as mirroring and the use of hot spare disks can help ensure data availability.

Performance is the average speed at which data is accessed. Policies such as write-verify and mirroring enhance availability but add to the system processing load, and thus degrade performance. Mirroring doubles or triples the size of the logical volume. In general, increasing availability degrades performance. Disk striping can increase performance. Beginning with AIX 4.3.3, disk striping is allowed with mirroring. Beginning with AIX 5.1, you can detect and remedy hot-spot problems that occur when some of the logical partitions on your disk have so much disk I/O that your system performance noticeably suffers.

By controlling the allocation of data on the disk and between disks, you can tune the storage system for the highest possible performance. See *Performance management* for detailed information on how to maximize storage-system performance.

Use the following sections to evaluate the trade-offs among performance, availability, and cost. Remember that increased availability often decreases performance, and vice versa. Mirroring may increase performance, however, 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 disk backups.

Requirements for mirroring or striping

Determine whether the data that is stored in the logical volume is valuable enough to warrant the processing and disk-space costs of mirroring. If you have a large sequential-access file system that is performance-sensitive, you may want to consider disk striping.

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

AIX LVM supports the following RAID options:

Table 2. Logical Volume Manager support for RAID

RAID 0	Striping
RAID 1	Mirroring
RAID 10 or 0+1	Mirroring and striping

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

You can mirror the rootvg, but if you do, create a separate dump logical volume. Dumping to a mirrored logical volume can result in an inconsistent dump. Also, because the default dump device is the primary paging logical volume, create a separate dump logical volume if you mirror your paging logical volumes.

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.

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

Scheduling policies 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, the following scheduling policies for writing to disk can be set for a logical volume with multiple copies:

Sequential-scheduling policy

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

Parallel-scheduling policy

Simultaneously starts the Write operation for all the physical partitions in a logical partition. When the Write operation to the physical partition that takes the longest to complete finishes, the Write operation is completed. Specifying mirrored logical volumes with a parallel-scheduling policy might improve I/O read-operation performance, because multiple copies allow the system to direct the read operation to the least busy disk for this logical volume.

Parallel write with sequential read-scheduling policy

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

Parallel write with round robin read-scheduling policy

Simultaneously starts the Write operation for all the physical partitions in a logical partition. Reads are switched back and forth between the mirrored copies.

Bad block policy

Indicates whether the volume group is enabled for bad block relocation. The default value is *yes*. When the value is set to *yes* for the volume group, the bad blocks can be relocated. When the value is set to *no*, the policy overrides the logical volume settings. When the value is changed, all logical volumes continue with the previous setting. The value indicates whether or not a requested I/O must be directed to a relocated block. If the value is set to *yes*, the volume group allows bad block relocation. If the value is set to *no*, bad block allocation is not completed. The LVM performs software relocation only when hardware relocation fails. Otherwise, the LVM bad block relocation (BBR) flag has no effect.

Note: Bad block relocation is disabled unless the bad block policy settings for volume group and logical volume are both set to *yes*.

Mirror Write Consistency policy for a logical volume

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

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

When active MWC is being used, system performance can be adversely affected. Adverse effects are caused by the overhead of logging or journaling a Write request in which a Logical Track Group (LTG) is active. The allowed LTG sizes for a volume group are 128 K, 256 K, 512 K, 1024 K, 2 MB, 4 MB, 8 MB, and 16 MB.

Note: To have an LTG size greater than 128 K, the disks contained in the volume group must support I/O requests of this size from the disk's strategy routines. The LTG is a contiguous block contained within the logical volume and is aligned on the size of the LTG. This overhead is for mirrored Writes only.

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

When MWC is set to passive, the volume group logs that the logical volume has been opened. After a crash when the volume group is varied on, an automatic force sync of the logical volume is started. Consistency is maintained while the force sync is in progress by using a copy of the read recovery policy that propagates the blocks being read to the other mirrors in the logical volume. This policy is only supported on the BIG volume group type.

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

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

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

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

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

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

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

Inter-disk allocation policies

The inter-disk allocation policy specifies the number of disks on which the physical partitions of a logical volume are located.

The physical partitions for a logical volume might be located on a single disk or spread across all the disks in a volume group. The following options are used with the **mkiv** and **chiv** commands 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 **mkiv** operation succeeds if two or more copies must occupy the same physical volume.
- The **Super Strict** option specifies that the partitions allocated for one mirror cannot share a physical volume with the partitions from another mirror.
- Striped logical volumes can only have a maximum range and a super 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 “Inter-disk settings for logical volume copies” on page 54.

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

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

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

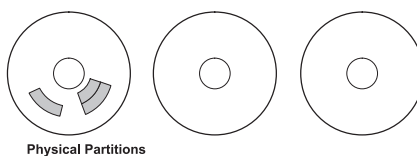


Figure 3. Minimum Inter-Disk Allocation Policy

This illustration shows three disks. One disk contains three physical partitions; the others have no physical partitions.

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

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

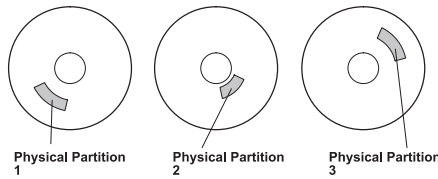


Figure 4. Maximum Inter-Disk Allocation Policy

This illustration shows three disks, each containing one physical partition.

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

Inter-disk settings for logical volume copies

The allocation of a single copy of a logical volume on disk is fairly straightforward.

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

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

The setting `Strict = y` means that each copy of the logical partition is placed on a different physical volume. The setting `Strict = n` means that the copies are not restricted to different physical volumes. By comparison, the Super Strict option would not allow any physical partition from one mirror to be on the same disk as a physical partition from another mirror of the same logical volume.

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

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

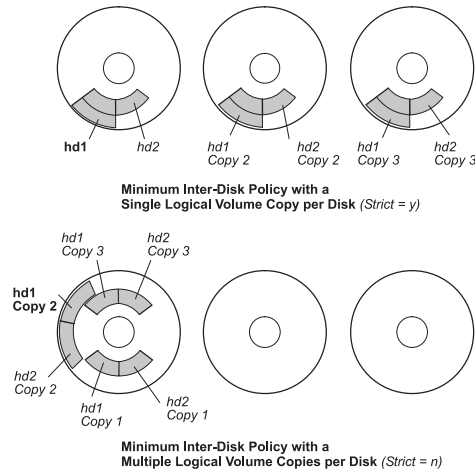


Figure 5. Minimum Inter-Disk Policy/Strict. This illustration shows that if the Strict option is equal to Yes, each copy of the logical partition is on a different physical volume. If Strict is equal to No, all copies of the logical partitions are on a single physical volume.

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

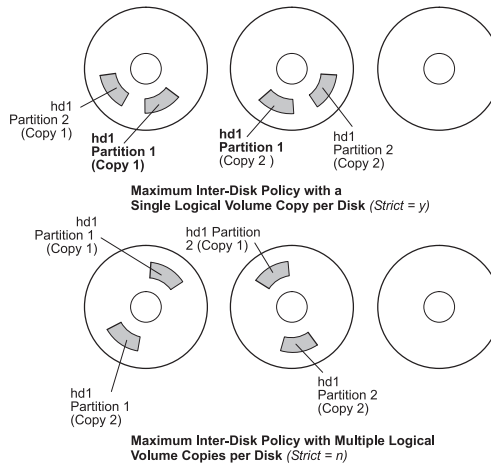


Figure 6. Maximum Inter-Disk Policy/Strict. This illustration shows that if the Strict option is equal to Yes, each copy of a partition is on a separate physical volume. If Strict is equal to No, all copies are on a single physical volume.

Intra-disk allocation policies for each logical volume

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

The closer a given physical partition is to the center of a physical volume, the lower the average seek time is 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 so frequently used by the operating system. At the other extreme, the boot logical volume is used infrequently and therefore is allocated at the edge or middle of the physical volume.

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

This rule has one important exception: Mirrored logical volumes with Mirror Write Consistency (MWC) set to On are at the outer edge because that is where the system writes MWC data. If mirroring is not in effect, MWC does not apply and does not affect performance.

The five regions where physical partitions can be located are as follows:

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

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

Combining allocation policies

If you select inter-disk and intra-disk policies that are not compatible, you might get unpredictable results.

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

Using map files for precise allocation

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

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

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. To verify that the physical partitions you plan to use are free to be allocated, type:

```
lspv -p hdisk1
```

2. Create a file, such as /tmp/mymap1, containing:

```
hdisk1:1-3  
hdisk1:41-45  
hdisk1:50-60
```

The **mklv** command allocates the physical partitions in the order that they appear in the map file. Be sure that there are sufficient physical partitions in the map file to allocate the entire logical volume that you specify with the **mklv** command. (You can list more than you need.)

3. Type the command:

```
mklv -t jfs -y lv06 -m /tmp/mymap1 rootvg 10
```

Developing striped logical volume strategies

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

Note: A dump space or boot logical volume cannot be striped. The boot logical volume must be contiguous physical partitions.

To create a 12-partition striped logical volume called 1v07 in VGName with a strip size (the strip size multiplied by the number of disks in an array equals the stripe size) of 16 KB across hdisk1, hdisk2, and hdisk3, type:

```
mk1v -y 1v07 -S 16K VGName 12 hdisk1 hdisk2 hdisk3
```

To create a 12-partition striped logical volume called 1v08 in VGName with a strip size of 8 KB across any three disks within VGName, type:

```
mk1v -y 1v08 -S 8K -u 3 VGName 12
```

For more information on how to improve performance by using disk striping, see *Performance management*.

Write-verify policies

Using the write-verify option causes all Write operations to be verified by an immediate follow-up Read operation to check the success of the Write.

If the Write operation is not successful, you get an error message. This policy enhances availability but degrades performance because of the extra time needed for the Read. You can specify the use of a write-verify policy on a logical volume either when you create it using the **mk1v** command, or later by changing it with the **chlv** command.

Hot-spare disk policies

Beginning with AIX 5.1, you can designate disks as hot-spare disks for a volume group with mirrored logical volumes.

When you designate which disks to use as hot-spare disks, you can specify a policy to be used if a disk or disks start to fail and you can specify synchronization characteristics. This support complements but does not replace the sparing support available with serial storage architecture (SSA) disks. You can also use hot-spare disks with SSA disks when you add one to your volume group.

If you add a physical volume to a volume group (to mark it as a hot-spare disk), the disk must have at least the same capacity as the smallest disk already in the volume group. When this feature is implemented, data will be migrated to a hot-spare disk when Mirror Write Consistency (MWC) write failures mark a physical volume missing.

The commands to enable hot-spare disk support, **chvg** and **chpv**, provide several options in how you implement the feature at your site, as shown by the following syntax:

```
chvg -hhotsparepolicy -ssyncpolicy VolumeGroup
```

Where *hotsparepolicy* determines which of the following policies you want to use when a disk is failing:

- y** Automatically migrates partitions from one failing disk to one spare disk. From the pool of hot spare disks, the smallest one that is big enough to substitute for the failing disk will be used.
- Y** Automatically migrates partitions from a failing disk, but might use the complete pool of hot-spare disks.
- n** Does not migrate automatically (default).
- r** Removes all disks from the pool of hot-spare disks for this volume group.

The *syncpolicy* argument determines whether you want to synchronize any stale partitions automatically:

- y** Automatically tries to synchronize stale partitions.
- n** Does not automatically try to synchronize stale partitions. (This option is the default.)

The *VolumeGroup* argument specifies the name of the associated mirrored volume group.

Hot spot management in logical volumes

Beginning with AIX 5.1, you can identify *hot spot* problems with your logical volumes and remedy those problems without interrupting the use of your system.

A hot-spot problem occurs when some of the logical partitions on your disk have so much disk I/O that your system performance noticeably suffers.

The first step toward solving the problem is to identify it. By default, the system does not collect statistics for logical volume use. After you enable the gathering of these statistics, the first time you enter the **lvmstat** command, the system displays the counter values since the previous system reboot. Thereafter, each time you enter the **lvmstat** command, the system displays the difference since the previous **lvmstat** command.

By interpreting the output of the **lvmstat** command, you can identify the logical partitions with the heaviest traffic. If you have several logical partitions with heavy usage on one physical disk and want to balance these across the available disks, you can use the **migratelp** command to move these logical partitions to other physical disks.

In the following example, the gathering of statistics is enabled and the **lvmstat** command is used repeatedly to gather a baseline of statistics:

```
# lvmstat -v rootvg -e
# lvmstat -v rootvg -C
# lvmstat -v rootvg
```

The output is similar to the following:

Logical Volume	iocnt	Kb_read	Kb_wrtn	Kbps
hd8	4	0	16	0.00
paging01	0	0	0	0.00
lv01	0	0	0	0.00
hd1	0	0	0	0.00
hd3	0	0	0	0.00
hd9var	0	0	0	0.00
hd2	0	0	0	0.00
hd4	0	0	0	0.00
hd6	0	0	0	0.00
hd5	0	0	0	0.00

The previous output shows that all counters have been reset to zero. In the following example, data is first copied from the **/unix** directory to the **/tmp** directory. The **lvmstat** command output reflects the activity for the rootvg:

```
# cp -p /unix /tmp
# lvmstat -v rootvg
```

Logical Volume	iocnt	Kb_read	Kb_wrtn	Kbps
hd3	296	0	6916	0.04
hd8	47	0	188	0.00
hd4	29	0	128	0.00
hd2	16	0	72	0.00
paging01	0	0	0	0.00
lv01	0	0	0	0.00
hd1	0	0	0	0.00
hd9var	0	0	0	0.00
hd6	0	0	0	0.00
hd5	0	0	0	0.00

The output shows activity on the **hd3** logical volume, which is mounted in the **/tmp** directory, on **hd8**, which is the JFS log logical volume, on **hd4**, which is **/** (root), on **hd2**, which is the **/usr** directory, and on **hd9var**, which is the **/var** directory. The following output provides details for **hd3** and **hd2**:

```
# lvmstat -l hd3
Log_part  mirror#  iocnt  Kb_read  Kb_wrtn  Kbps
      1      1      299      0      6896    0.04
      3      1       4       0       52    0.00
      2      1       0       0       0     0.00
      4      1       0       0       0     0.00
# lvmstat -l hd2
Log_part  mirror#  iocnt  Kb_read  Kb_wrtn  Kbps
      2      1       9       0       52    0.00
      3      1       9       0       36    0.00
      7      1       9       0       36    0.00
      4      1       4       0       16    0.00
      9      1       1       0        4    0.00
     14      1       1       0        4    0.00
      1      1       0       0        0    0.00
```

The output for a volume group provides a summary for all the I/O activity of a logical volume. It is separated into the number of I/O requests (**iocnt**), the kilobytes read and written (**Kb_read** and **Kb_wrtn**, respectively), and the transferred data in KB/s (**Kbps**). If you request the information for a logical volume, you receive the same information, but for each logical partition separately. If you have mirrored logical volumes, you receive statistics for each of the mirror volumes. In the previous sample output, several lines for logical partitions without any activity were omitted. The output is always sorted in decreasing order on the **iocnt** column.

The **migratelp** command uses, as parameters, the name of the logical volume, the number of the logical partition (as it is displayed in the **lvmstat** output), and an optional number for a specific mirror copy. If information is omitted, the first mirror copy is used. You must specify the target physical volume for the move; in addition, you can specify a target physical partition number. If successful, the output is similar to the following:

```
# migratelp hd3/1 hdisk1/109
migratelp: Mirror copy 1 of logical partition 1 of logical volume
          hd3 migrated to physical partition 109 of hdisk1.
```

After the hot spot feature is enabled, either for a logical volume or a volume group, you can define your reporting and statistics, display your statistics, select logical partitions to migrate, specify the destination physical partition, and verify the information before committing your changes. The Web-based System Manager helps you configure hot-spot reporting and manage the result. For instructions on enabling hot spot reporting, see “Enablement and configuration of hot spot reporting” on page 36.

Volume group policy implementation

After you have decided which volume group policies you want to use, analyze your current configuration by typing the **lspv** command on the command line.

The standard configuration provides a single volume group that includes multiple physical volumes attached to the same disk adapter and other supporting hardware. 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. To implement your volume group policy changes, do the following:

1. Use the **lspv** command output to check your allocated and free physical volumes.
2. Ensure a quorum by adding one or more physical volumes. For instructions, see “Adding disks while the system remains available” on page 32, Adding a Fixed Disk Without Data to an Existing Volume Group, or Adding a Fixed Disk Without Data to a New Volume Group.

3. To change a volume group to nonquorum status, see “Conversion of a volume group to nonquorum status” on page 74.
4. Reconfigure the hardware only if necessary to ensure high availability. For instructions, see the service guide for your system.

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 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 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.

For more information about how LVM works, see *Understanding the Logical Volume Device Driver in AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts* and *Logical Volume Programming Overview in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

Logical Volume Manager configuration tasks

The Logical Volume Manager (LVM) is installed with the base operating system and needs no further configuration. However, disks must be configured and defined as a physical volume before the LVM can use them.

If you want to set up raw logical volumes for use by an application, see “Defining a raw logical volume for an application” on page 40.

Configuring a disk

You can configure a new disk by various methods.

You can configure a new disk in any of the following ways.

- If you can shut down and power off the system, use Method 1. Whenever possible, it is always preferable to shut down and power off any system when you are attaching a physical disk to it.
- If you cannot shut down your system and you know details about the new disk, such as the subclass, type, parent name, and where it is connected, use Method 2.
- If you cannot shut down your system and you only know the location of the disk, use Method 3.

After a disk is configured, although it is generally available for use, the Logical Volume Manager requires that it is further identified as a physical volume.

Method 1

Use the following method when you can shut down and power off the system before attaching the disk:

1. Physically connect the new disk to the system and then power on the disk and system according to the documentation that came with your system.
2. During system boot, let the Configuration Manager (**cfgmgr**) automatically configure the disk.

3. After system boot, with root authority, type the **lspv** command at the command line to look for the new disk's name. The system returns an entry similar to one of the following:

```
hdisk1 none none
```

or:

```
hdisk1 00005264d21adb2e none
```

The first field identifies the system-assigned name of the disk. The second field displays the physical volume ID (PVID), if any. If the new disk does not appear in the **lspv** output, refer to the *Installation and migration*.

At this point, the disk is usable by the system but it needs a PVID for use by the LVM. If the new disk does not have a PVID, then see “Making an available disk a physical volume” on page 62.

Method 2

Use the following method when you cannot shut down your system and you know the following information about the new disk:

- How the disk is attached (subclass)
- The type of the disk (type)
- Which system attachment the disk is connected to (parent name)
- The logical address of the disk (where connected).

Do the following:

1. Physically connect the new disk to the system and then power on the disk and system according to the documentation that came with your system.
2. To configure the disk and ensure that it is available as a physical volume, use the **mkdev** command with the flags shown, as in the following example:

```
mkdev -c disk -s scsi -t 2200mb -p scsi3 \  
-w 6,0 -a pv=yes
```

This example adds a 2.2 GB disk with a SCSI ID of 6 and logical unit number of 0 to the scsi3 SCSI bus. The **-c** flag defines the class of the device. The **-s** flag defines the subclass. The **-t** flag defines the type of device. The **-p** flag defines the parent device name that you want to assign. The **-w** flag designates the disk's location by SCSI ID and logical unit number. The **-a** flag specifies the device attribute-value pair, **pv=yes**, which makes the disk a physical volume and writes a boot record with a unique physical volume identifier onto the disk (if it does not already have one).

At this point, the disk is defined both as an available device and as a physical volume. You can type the **lspv** command on the command line to list the new disk entry. If the new disk does not appear in the **lspv** output, refer to the *Installation and migration*.

Method 3

Use the following method when you cannot shut down your system and you know only the location of the disk:

1. Physically connect the new disk to the system and then power on the disk and system according to the documentation that came with your system.
2. To check which physical disks are already configured on the system, type the **lspv** command on the command line. For more information about the **lspv** command, see [../././com.ibm.aix.cmds/doc/aixcmds3/lspv.htm](#). The output looks similar to the following:

```
hdisk0 000005265ac63976 rootvg
```

3. Type **cfgmgr** on the command line to enter the Configuration Manager. The Configuration Manager automatically detects and configures all newly connected devices on the system, including the new disk. For more information about the **cfgmgr** command, see **cfgmgr**.
4. To confirm that the new disk was configured, type the **lspv** command again. The output looks similar to one of the following:

```
hdisk1    none                none
```

or

```
hdisk1    00005264d21adb2e    none
```

The first field identifies the system-assigned name of the disk. The second field displays the physical volume ID (PVID), if any. If the new disk does not appear in the **lspv** output, refer to the *Installation and migration*.

At this point, the disk is usable by the system but it needs a PVID for use by the LVM. If the new disk does not have a PVID, then see “Making an available disk a physical volume.”

Making an available disk a physical volume

A disk must be configured as a physical volume before it can be assigned to volume groups and used by the LVM.

Use the following instructions to configure a physical volume:

1. Ensure the disk is known to the operating system, is available, and is not being used by the operating system or any applications. Type the **lspv** command on the command line. The output looks similar to the following:

```
hdisk1    none                none
```

Check the output for the following:

- If the new disk’s name does not appear in command output, refer to “Configuring a disk” on page 60.
- If the second field of the output shows a system-generated physical volume identifier (PVID) (for example, 00005264d21adb2e), the disk is already configured as a physical volume and you do not have to complete this procedure.
- If the third field of the output shows a volume group name (for example, rootvg), the disk is currently being used and is not an appropriate choice for this procedure.

If the new disk has no PVID and is not in use, continue with the next step.

2. To change an available disk to a physical volume, type the **chdev** command on the command line. For example:

```
chdev -l hdisk3 -a pv=yes
```

The **-l** flag specifies the device name of the disk. The **-a** flag specifies the device attribute-value pair, **pv=yes**, which makes the disk a physical volume and writes a boot record with a unique physical volume identifier onto the disk (if it does not already have one).

At this point, the disk is defined as a physical volume. You can type the **lspv** command on the command line to list the new disk entry.

Troubleshooting LVM

There are several common types of problems with LVM that you can troubleshoot.

Troubleshooting disk drive problems

This information tells how to diagnose and fix disk drive problems.

If your disk drive is running out of available space, see “Disk drive space.” If you suspect a disk drive is mechanically failing or has failed, run diagnostics on the disk use the following procedure:

1. With root authority, type the following SMIT fast path on the command line:
`smit diag`
2. Select **Current Shell Diagnostics** to enter the AIX Diagnostics tool.
3. After you read the Diagnostics Operating Instructions screen, press Enter.
4. Select **Diagnostics Routines**.
5. Select **System Verification**.
6. Scroll down through the list to find and select the drive you want to test.
7. Select **Commit**.

Based on the diagnostics results, you should be able to determine the condition of the disk:

- If you detect the disk drive is failing or has failed, of primary importance is recovering the data from that disk. If the disk is still accessible, try completing the procedure in “Migrating the contents of a physical volume” on page 38. Migration is the preferred way to recover data from a failing disk. The following procedures describe how to recover or restore data in logical volumes if migration cannot complete successfully.
- If your drive is failing and you can repair the drive without reformatting it, no data will be lost. See “Disk drive recovery without reformatting” on page 64.
- If the disk drive must be reformatted or replaced, make a backup, if possible, and remove the disk drive from its volume group and system configuration before replacing it. Some data from single-copy file systems might be lost. See “Recovering using a reformatted or replacement disk drive” on page 64.
- If your system supports the hot removability feature, see “Recovering from disk failure while the system remains available” on page 69.

Disk drive space

If you run out of space on a disk drive, there are several ways to remedy the problem. You can automatically track and remove unwanted files, restrict users from certain directories, or mount space from another disk drive.

You must have root user, system group, or administrative group authority to execute these tasks.

Command for cleaning up file systems automatically:

Use the **skulker** command to clean up file systems by removing unwanted files.

Type the following from the command line:

```
skulker -p
```

The **skulker** command is used to periodically purge obsolete or unneeded files from file systems. Candidates include files in the **/tmp** directory, files older than a specified age, **a.out** files, core files, or **ed.hup** files. For more information about the **skulker** command, see **skulker**.

The **skulker** command is typically run daily, as part of an accounting procedure run by the **cron** command during off-peak hours. For more information about using the **skulker** command in a **cron** process, see “Disk overflows” on page 108.

For information on typical **cron** entries, see “Setting up an accounting system” on page 245.

Restricting users from certain directories:

You can release disk space and possibly keep it free by restricting access to directories and monitoring disk usage.

1. Restrict users from certain directories by typing:

```
chmod 655 DirName
```

This command sets read and write permissions for the owner (root) and sets read-only permissions for the group and others. *DirName* is the full path name of the directory you want to restrict.

2. Monitor the disk usage of individual users. One way to do this is to add the following line to the **/var/spool/cron/crontabs/adm** file:

```
0 2 * * 4 /usr/sbin/acct/dodisk
```

This line executes the **dodisk** command at 2 a.m. (0 2) each Thursday (4). The **dodisk** command initiates disk-usage accounting. This command is usually run as part of an accounting procedure run by the **cron** command during off-peak hours. See “Setting up an accounting system” on page 245 for more information on typical **cron** entries.

Mounting space from another disk drive:

You can get more space on a disk drive by mounting space from another drive.

You can mount space from one disk drive to another in the following ways:

- Use the **smit mountfs** fast path.
- Use the **mount** command. For example:

```
mount -n nodeA -vnfs /usr/spool /usr/myspool
```

The **mount** command makes a file system available for use at a specific location.

For more information about mounting file systems, see Mount a JFS or JFS2.

Disk drive recovery without reformatting

If you repair a bad disk and place it back in the system without reformatting it, you can let the system automatically activate and resynchronize the stale physical partitions on the drive at boot time. A *stale physical partition* contains data your system cannot use.

If you suspect a stale physical partition, type the following on the command line:

```
lspv -M PhysVolName
```

Where *PhysVolName* is the name of your physical volume. The **lspv** command output will list all partitions on your physical volume. The following is an excerpt from example output:

hdisk16:112	lv01:4:2	stale
hdisk16:113	lv01:5:2	stale
hdisk16:114	lv01:6:2	stale
hdisk16:115	lv01:7:2	stale
hdisk16:116	lv01:8:2	stale
hdisk16:117	lv01:9:2	stale
hdisk16:118	lv01:10:2	stale

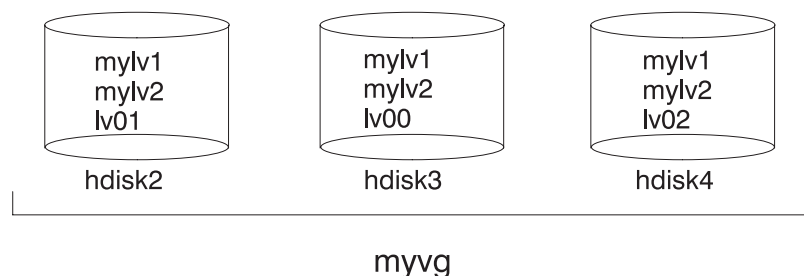
The first column displays the physical partitions and the second column displays the logical partitions. Any stale physical partitions are noted in the third column.

Recovering using a reformatted or replacement disk drive

You can recover data from a failed disk drive when you must reformat or replace the failed disk.

Attention: Before you reformat or replace a disk drive, remove all references to nonmirrored file systems from the failing disk and remove the disk from the volume group and system configuration. If you do not, you create problems in the ODM (object data manager) and system configuration databases. Instructions for these essential steps are included in the following procedure, under Before replacing or reformatting your failed or failing disk.

The following procedure uses a scenario in which the volume group called **myvg** contains three disk drives called **hdisk2**, **hdisk3**, and **hdisk4**. In this scenario, **hdisk3** goes bad. The nonmirrored logical volume **lv01** and a copy of the **mylv** logical volume is contained on **hdisk2**. The **mylv** logical volume is mirrored and has three copies, each of which takes up two physical partitions on its disk. The failing **hdisk3** contains another copy of **mylv**, and the nonmirrored logical volume called **lv00**. Finally, **hdisk4** contains a third copy of **mylv** as well as a logical volume called **lv02**. The following figure shows this scenario.



This procedure is divided into the following key segments:

- The things you do to protect data before replacing or reformatting your failing disk
- The procedure you follow to reformat or replace the disk
- The things you do to recover the data after the disk is reformatted or replaced

Before replacing or reformatting your failed or failing disk:

1. Log in with root authority.
2. If you are not familiar with the logical volumes that are on the failing drive, use an operational disk to view the contents of the failing disk. For example, to use **hdisk4** to look at **hdisk3**, type the following on the command line:

```
lspv -M -n hdisk4 hdisk3
```

The **lspv** command displays information about a physical volume within a volume group. The output looks similar to the following:

```
hdisk3:1      mylv:1
hdisk3:2      mylv:2
hdisk3:3      lv00:1
hdisk3:4-50
```

The first column displays the physical partitions, and the second column displays the logical partitions. Partitions 4 through 50 are free.

3. Back up all single-copy logical volumes on the failing device, if possible. For instructions, see “Backing up user files or file systems” on page 149.
4. If you have single-copy file systems, unmount them from the disk. (You can identify single-copy file systems from the output of the **lspv** command. Single-copy file systems have the same number of logical partitions as physical partitions on the output.) Mirrored file systems do not have to be unmounted.

In the scenario, **lv00** on the failing disk **hdisk3** is a single-copy file system. To unmount it, type the following:

```
umount /dev/lv00
```

If you do not know the name of the file system, assuming the **/etc/filesystems** file is not solely located on the failed disk, type **mount** on the command line to list all mounted file systems and find the name associated with your logical volume. You can also use the **grep** command on the **/etc/filesystems** file to list only the file system names, if any, associated with your logical volume. For example:

```
grep lv00 /etc/filesystems
```

The output looks similar to the following:

```
dev          = /dev/lv00
log         = /dev/loglv00
```

Note:

- a. The **umount** command fails if the file system you are trying to unmount is currently being used. The **umount** command executes only if none of the file system's files are open and no user's current directory is on that device.
 - b. Another name for the **umount** command is **umount**. The names are interchangeable.
5. Remove all single-copy file systems from the failed physical volume by typing the **rmfs** command:

```
rmfs /FSname
```
 6. Remove all mirrored logical volumes located on the failing disk.

Note: You cannot use **rmlvcopy** on the hd5 and hd7 logical volumes from physical volumes in the rootvg volume group. The system does not allow you to remove these logical volumes because there is only one copy of these.

The **rmlvcopy** command removes copies from each logical partition. For example, type:

```
rmlvcopy mylv 2 hdisk3
```

By removing the copy on hdisk3, you reduce the number of copies of each logical partition belonging to the **mylv** logical volume from three to two (one on hdisk4 and one on hdisk2).

7. If the failing disk was part of the root volume group and contained logical volume hd7, remove the primary dump device (hd7) by typing the following on the command line:

```
sysdumpdev -P -p /dev/sysdumpnul1
```

The **sysdumpdev** command changes the primary or secondary dump device location for a running system. When you reboot, the dump device returns to its original location.

Note: Beginning with AIX 5.3, you can choose to dump to a DVD device. For more information on how to configure a DVD to be the dump device, see **sysdumpdev** in *AIX 5L Version 5.3 Commands Reference, Volume 5*

8. Remove any paging space located on the disk using the following command:

```
rmpps PSname
```

Where *PSname* is the name of the paging space to be removed, which is actually the name of the logical volume on which the paging space resides.

If the **rmpps** command is not successful, you must use the **smit chpps** fast path to deactivate the primary paging space and reboot before continuing with this procedure. The **reducevg** command in step 10 can fail if there are active paging spaces.

9. Remove any other logical volumes from the volume group, such as those that do not contain a file system, using the **rmlv** command. For example, type:

```
rmlv -f lv00
```
10. Remove the failed disk from the volume group using the **reducevg** command. For example, type:

```
reducevg -df myvg hdisk3
```

If you cannot execute the **reducevg** command or if the command is unsuccessful, the procedure in step 13 can help clean up the VGDA/ODM information after you reformat or replace the drive.

Replacing or reformatting your failed or failing disk:

11. The next step depends on whether you want to reformat or replace your disk and on what type of hardware you are using:

- If you want to reformat the disk drive, use the following procedure:
 - a. With root authority, type the following SMIT fast path on the command line:


```
smit diag
```
 - b. Select **Current Shell Diagnostics** to enter the AIX Diagnostics tool.
 - c. After you read the **Diagnostics Operating Instructions** screen, press Enter.
 - d. Select **Task Selection**.
 - e. Scroll down through the task list to find and select **Format Media**.
 - f. Select the disk you want to reformat. After you confirm that you want to reformat the disk, all content on the disk will be erased.

After the disk is reformatted, continue with step 12.

- If your system supports hot swap disks, use the procedure in “Recovering from disk failure while the system remains available” on page 69, then continue with step 13.
- If your system does not support hot swap disks, do the following:
 - Power off the old drive using the SMIT fast path **smit rmvdsk**. Change the KEEP definition in database field to No.
 - Contact your next level of system support to replace the disk drive.

After replacing or reformatting your failed or failing disk:

12. Follow the instructions in “Configuring a disk” on page 60 and “Making an available disk a physical volume” on page 62.
13. If you could not use the **reducevg** command on the disk from the old volume group before the disk was formatted (step 10), the following procedure can help clean up the VGDA/ODM information.
 - a. If the volume group consisted of only one disk that was reformatted, type:

```
exportvg VGName
```

Where *VGName* is the name of your volume group.

- b. If the volume group consists of more than one disk, type the following on the command line:

```
varyonvg VGName
```

The system displays a message about a missing or unavailable disk, and the new (or reformatted) disk is listed. Note the physical volume identifier (PVID) of the new disk, which is listed in the **varyonvg** message. It is the 16-character string between the name of the missing disk and the label PVNOTFND. For example:

```
hdisk3 00083772caa7896e PVNOTFND
```

Type:

```
varyonvg -f VGName
```

The missing disk is now displayed with the PVREMOVED label. For example:

```
hdisk3 00083772caa7896e PVREMOVED
```

Then, type the command:

```
reducevg -df VGName PVID
```

Where PVID is the physical volume identifier (in this scenario, 00083772caa7896e).

14. To add the new disk drive to the volume group, use the **extendvg** command. For example, type:

```
extendvg myvg hdisk3
```

15. To re-create the single-copy logical volumes on the new (or reformatted) disk drive, use the **mklv** command. For example, type:

```
mklv -y 1v00 myvg 1 hdisk3
```

This example recreates the lv00 logical volume on the hdisk3 drive. The 1 means that this logical volume is not mirrored.

16. To re-create the file systems on the logical volume, use the **crfs** command. For example, type:

```
crfs -v jfs -d lv00 -m /dev/lv00
```

17. To restore single-copy file system data from backup media, see “Restoring user files from a backup image” on page 166.

18. To re-create the mirrored copies of logical volumes, use the **mklvcopy** command. For example, type:

```
mklvcopy mylv 3 hdisk3
```

This example creates a mirrored third partition of the **mylv** logical volume on hdisk3.

19. To synchronize the new mirror with the data on the other mirrors (in this example, hdisk2 and hdisk4), use the **syncvg** command. For example, type:

```
syncvg -p hdisk3
```

At this point, all mirrored file systems should be restored and up-to-date. If you were able to back up your single-copy file systems, they will also be ready to use. You should be able to proceed with normal system use.

Example of recovering from a failed disk drive:

To recover from a failed disk drive, back out the way you came in; that is, list the steps you went through to create the volume group, and then go backwards.

The following example is an illustration of this technique. It shows how a mirrored logical volume was created and then how it was altered, backing out one step at a time, when a disk failed.

Note: The following example illustrates a specific instance. It is not intended as a general prototype on which to base any general recovery procedures.

1. The system manager, Jane, created a volume group called **workvg** on hdisk1, by typing:

```
mkvg -y workvg hdisk1
```

2. She then created two more disks for this volume group, by typing:

```
extendvg workvg hdisk2
```

```
extendvg workvg hdisk3
```

3. Jane created a logical volume of 40 MB that has three copies. Each copy is on one of each of the three disks that comprise the **workvg** volume group. She used the following commands:

```
mklv -y testlv workvg 10
```

```
mklvcopy testlv 3
```

After Jane created the mirrored workvg volume group, hdisk2 failed. Therefore, she took the following steps to recover:

1. She removed the logical volume copy from hdisk2 by typing:

```
rmllvcopy testlv 2 hdisk2
```

2. She detached hdisk2 from the system so that the ODM and VGDA are updated, by typing:

```
reducevg workvg hdisk2
```

3. She removed hdisk2 from the system configuration to prepare for replacement by typing:

```
rmdev -l hdisk2 -d
```

4. She chose to shut down the system, by typing:

```
shutdown -F
```

5. She replaced the disk. The new disk did not have the same SCSI ID as the former hdisk2.

6. She rebooted the system.

Because you have a new disk (the system sees that there is a new PVID on this disk), the system chooses the first *open* hdisk name. Because the **-d** flag was used in step 3, the name hdisk2 was released, so the system chose hdisk2 as the name of the new disk. If the **-d** flag had not been used, hdisk4 would have been chosen as the new name.

7. Jane added this disk into the **workvg** volume group by typing:

```
extendvg workvg hdisk2
```

8. She created two mirrored copies of the logical volume by typing:

```
mk1vcopy testlv 3
```

The Logical Volume Manager automatically placed the third logical volume copy on the new hdisk2.

Recovering from disk failure while the system remains available

You can recover from disk failure using the hot removability feature.

The procedure to recover from disk failure using the hot removability feature is, for the most part, the same as described in “Disk drive recovery without reformatting” on page 64, with the following exceptions:

1. To unmount file systems on a disk, use the procedure Mount a JFS or JFS2.
2. To remove the disk from its volume group and from the operating system, use the procedure “Removing a disk without data” on page 44.
3. To replace the failed disk with a new one, you do not need to shut down the system. Use the following sequence of procedures:
 - a. Chapter 3, “Logical volume storage,” on page 25
 - b. “Configuring a disk” on page 60
 - c. Continue with step 13 of “Recovering using a reformatted or replacement disk drive” on page 64.

Replacing a disk when the volume group consists of one disk

Use one of these procedures if you can access a disk that is going bad as part of a volume group.

- Add fixed disk without data to existing volume group
- Add fixed disk without data to new volume group
- “Migrating the contents of a physical volume” on page 38

If the disk is bad and cannot be accessed, follow these steps:

1. Export the volume group.
2. Replace the drive.
3. Re-create the data from backup media that exists.

Physical and logical volume errors

There are several common errors with physical and logical volumes that you can troubleshoot.

Hot spot problems

If you notice performance degradation when accessing logical volumes, you might have hot spots in your logical volumes that are experiencing too much disk I/O.

For more information, see “Hot spot management in logical volumes” on page 58 and “Enablement and configuration of hot spot reporting” on page 36.

LVCB warnings

A warning results if the LVCB contains invalid information.

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 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 you might receive the following message:

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

Most of the time, this message results when database programs bypass the JFS and access raw logical volumes as storage media. When this occurs, the information for the database is literally written over the LVCB. For raw logical volumes, this is not fatal. After the LVCB is overwritten, the user can still:

- Expand a logical volume
- Create mirrored copies of the logical volume
- Remove the logical volume
- Create a journaled file system to mount the logical volume

There are limitations to deleting LVCBs. A logical volumes with a deleted LVCB might not import successfully to other systems. During an importation, the LVM **importvg** command scans the LVCBs of all defined logical volumes in a volume group for information concerning the logical volumes. If the LVCB does not exist, the imported volume group still defines the logical volume to the new system that is accessing this volume group, and the user can still access the raw logical volume. However, the following typically happens:

- Any JFS information is lost and the associated mount point is not imported to the new system. In this case, you must create new mount points, and the availability of previous data stored in the file system is not ensured.
- Some non-JFS information concerning the logical volume cannot be found. When this occurs, the system uses default logical volume information to populate the ODM information. Thus, some output from the **lslv** command might be inconsistent with the real logical volume. If any logical volume copies still exist on the original disks, the information is not be correctly reflected in the ODM database. Use the **rmivcopy** and **mkivcopy** commands to rebuild any logical volume copies and synchronize the ODM.

Physical partition limits

In the design of Logical Volume Manager (LVM), each logical partition maps to one physical partition (PP). And, each physical partition maps to a number of disk sectors. The design of LVM limits the number of physical partitions that LVM can track per disk to 1016. In most cases, not all of the 1016 tracking partitions are used by a disk.

When this limit is exceeded, you might see a message similar to the following:

```
0516-1162 extndvlg: Warning, The Physical Partition Size of PPsize requires the
creation of TotalPPs partitions for PVname. The limitation for volume group
VGname is LIMIT physical partitions per physical volume. Use chvg command
with -t option to attempt to change the maximum Physical Partitions per
Physical volume for this volume group.
```

Where:

PPsize

Is 1 MB to 1 GB in powers of 2.

Total PPs

Is the total number of physical partitions on this disk, given the *PPsize*.

PVname

Is the name of the physical volume, for example, hdisk3.

VGname

Is the name of the volume group.

LIMIT Is 1016 or a multiple of 1016.

This limitation is enforced in the following instances:

1. When creating a volume group using the **mkvg** command, you specified a number of physical partitions on a disk in the volume group that exceeded 1016. To avoid this limitation, you can select from the physical partition size ranges of 1, 2, 4 (the default), 8, 16, 32, 64, 128, 256, 512 or 1024 MB and use the **mkvg -s** command to create the volume group. Alternatively, you can use a suitable factor that allows multiples of 1016 partitions per disk, and use the **mkvg -t** command to create the volume group.
2. When adding a disk to a pre-existing volume group with the **extendvg** command, the new disk caused the 1016 limitation violation. To resolve this situation, convert the existing volume group to hold multiples of 1016 partitions per disk using the **chvg -t** command. Alternatively, you can re-create the volume group with a larger partition size that allows the new disk, or you can create a standalone volume group consisting of a larger physical size for the new disk.

Partition limitations and the rootvg

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

Partition limitations and RAID systems

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

Device configuration database synchronization

A system malfunction can cause the device configuration database to become inconsistent with the LVM. You can synchronize the device configuration database with the LVM information.

When the device configuration database becomes inconsistent with the LVM, a logical volume command generates such error messages as:

```
0516-322 The Device Configuration Database is inconsistent ...
```

OR

```
0516-306 Unable to find logical volume LVname in the Device  
Configuration Database.
```

(where the logical volume called *LVname* is normally available).

Attention: Do not remove the **/dev** entries for volume groups or logical volumes. Do not change the database entries for volume groups or logical volumes using the Object Data Manager.

To synchronize the device configuration database with the LVM information, with root authority, type the following on the command line:

```
sync1vodm -v VGName
```

Where *VGName* is the name of the volume group you want to synchronize.

Fixing volume group errors

Use these methods to fix volume group errors.

If the **importvg** command is not working correctly, try refreshing the device configuration database. See “Device configuration database synchronization.”

Overriding a vary-on failure

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

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

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

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

Use the following procedure to avoid losing quorum when one disk is missing or might soon fail and requires repair:

1. To temporarily remove the volume from the volume group, type:

```
chpv -vr PVname
```

When this command completes, the physical volume *PVname* is no longer factored in quorum checking. However, in a two-disk volume group, this command fails if you try the **chpv** command on the disk that contains the two VGDA/VGSAs. The command does not allow you to cause quorum to be lost.

2. If you need to remove the disk for repair, power off the system, and remove the disk. (For instructions, see “Troubleshooting disk drive problems” on page 62.) After fixing the disk and returning the disk to the system, continue with the next step.
3. To make the disk available again to the volume group for quorum checking, type:

```
chpv -v PVname
```

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.

VGDA warnings

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

```
0516-1163 extndvlg: VGname already has maximum physical volumes. With the maximum
number of physical partitions per physical volume being LIMIT, the maximum
number of physical volumes for volume group VGname is MaxDisks.
```

Where:

VGname

Is the name of the volume group.

LIMIT Is 1016 or a multiple of 1016.

MaxDisks

Is the maximum number of disks in a volume group. For example, if there are 1016 physical partitions (PPs) per disk, then *MaxDisk* is 32; if there are 2032, then *MaxDisk* is 16.

You can modify the **image.data** file and then use alternate disk installation, or restore the system using the **mksysb** command to re-create the volume group as a big volume group. For more information, see the *Installation and migration*.

On older AIX versions when the limit was smaller than 32 disks, the exception to this description of the maximum VGDA was the **rootvg**. To provide users with more free disk space, when rootvg was created, the **mkvg -d** command used the number of disks selected in the installation menu as the reference number. This **-d** number is 7 for one disk and one more for each additional disk selected. For example, if two disks are selected, the number is 8 and if three disks are selected, the number is 9, and so on.

Vary-on process

The vary-on process is one of the mechanisms that the LVM uses to ensure that a volume group is ready to use and contains the most up-to-date data.

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, 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 vary-off this volume group.

Quorum

The quorum is one of the mechanisms that the LVM uses to ensure that a volume group is ready to use and contains the most up-to-date data.

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 LVM. This prevents further disk I/O to that volume group so that data is not lost or assumed to be written

when physical problems occur. Additionally, as a result of the vary-off, the user is notified in the error log that a hardware error has occurred and service must be performed.

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

Nonquorum volume groups

The Logical Volume Manager (LVM) automatically deactivates the volume group when it lacks a quorum of Volume Group Descriptor Areas (VGDA) or Volume Group Status Areas (VGSAs). However, you can choose an option that allows the group to stay online as long as there is one VGDA/VGSA pair intact. This option produces a *nonquorum volume group*.

The LVM requires access to all of the disks in nonquorum volume groups before allowing reactivation. This ensures that the VGDA and VGSA are up-to-date.

You might want to produce a nonquorum volume group 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 active 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.

Even when you are using nonquorum volume groups, it is possible to lose quorum and see the following message in the **errpt** command output:

```
QUORUM LOST, VOLUME GROUP CLOSING LVM.
```

This message occurs when all physical volumes are in the `missing` state and the LVM automatically varies off the volume group.

The message says QUORUM LOST because disabling quorum on a volume group reduces the quorum requirement to 1. You can use the **lsvg vgrname** command to display the quorum value which is in the `QUORUM:` field. In the case where all physical volumes are `missing`, even this minimum quorum requirement is violated, resulting in the lost quorum message and an automatic vary off of the volume group.

Note: For more information on nonquorum volume groups, see the *AIX Logical Volume Manager, from A to Z: Introduction and Concepts*, an IBM Redbooks publication, available at www.redbooks.ibm.com.

Conversion of a volume group to nonquorum status

You can change a volume group to nonquorum status to have data continuously available even when there is no quorum.

This procedure is often used for systems with the following configurations:

- A two-disk volume group in which the logical volumes are mirrored
- A three-disk volume group in which the logical volumes are mirrored either once or twice

When a volume group under these circumstances can operate in nonquorum status, then even if a disk failure occurs, the volume group remains active as long as at least one disk in the volume group is active. For conceptual information about quorums, refer to “Quorum” on page 73.

To make recovery of nonquorum groups possible, ensure the following:

- If your system uses JFS or JFS2 file systems, mirror the JFS log logical volume.
- Place mirrored copies on separate disks. If you are unsure of the configuration, type the following command to check the physical location (PV1, PV2, and PV3) of each logical partition. (To place the copies on separate disks, the PV1, PV2, and PV3 columns must contain different hdisk numbers.)

```
lslv -m LVName
```

If a logical volume has its only copies residing on the same disk, and that disk becomes unavailable, the volume will not be available to the user regardless of the quorum or nonquorum status of its volume group.

Both user-defined and rootvg volume groups can operate in nonquorum status, but their configuration and recovery methods are different.

To activate a nonquorum user-defined volume group, all of the volume group’s physical volumes must be accessible or the activation fails. Because nonquorum volume groups stay online until the last disk becomes inaccessible, it is necessary to have each disk accessible at activation time.

Attention: When a disk associated with the rootvg volume group is missing, avoid powering on the system unless the missing disk cannot possibly be repaired. The Logical Volume Manager (LVM) always uses the **-f** flag to forcibly activate (vary on) a nonquorum rootvg; this operation involves risk. LVM must force the activation because the operating system cannot be brought up unless rootvg is activated. In other words, LVM makes a final attempt to activate (vary on) a nonquorum rootvg even if only a single disk is accessible.

Replacing a failed physical volume in a mirrored volume group

The following procedures replace a failed physical volume (PV) within a mirrored volume group. The **replacepv** command provides a method for replacing a failed PV in most configurations. An alternative procedure is also provided for configurations where the **replacepv** command cannot be used.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.
--

Prerequisites

- All logical volumes using the failed PV have valid copies on other available PVs (with the possible exception of a dedicated dump logical volume).

Replacing a failed PV using the **replacepv** command

Prerequisites

If any of the prerequisites listed below cannot be met, see the alternate procedure.

- The volume group containing the failed PV is not rootvg.
- The replacement PV can be added to the volume group containing the failed PV (this might not be possible depending on the PV size and volume group characteristics, such as MAX PPs per PV).
- The replacement PV must be able to be configured into the system at the same time as the failing PV.
- The replacement PV’s name can differ from the failed PV’s name.
- The size of the replacement PV must be at least the size of the failed PV.

- The volume group containing the failed PV must not be a snapshot volume group or have a snapshot volume group.

Complete the following steps, assuming that the failed PV is `hdisk2` and the replacement PV is `hdisk10`:

1. If the replacement PV is not yet installed on the system, perform the steps necessary to install it. To use the configuration manager to define a new PV, run the following command:

```
cfgmgr
```

Use the **lspv** command to determine the name assigned to the PV. For this example, assume that the new PV is named `hdisk10`.

2. To replace the failed PV with the one defined in Step 1, run the following command:

```
replacepv hdisk2 hdisk10
```

When the command runs, `hdisk2` is replaced by `hdisk10`, and `hdisk2` is no longer assigned to a volume group.

3. To undefine the failed PV, run the following command:

```
rmdev -d1 hdisk2
```

4. Physically remove the failed disk from the system.

5. Verify that the procedure was successful by completing the following steps:

- To check that all logical volumes are mirrored to the new PV as desired, run the following command:

```
lslv lvname
```

Check the **COPIES** attribute of each logical volume affected by the failed PV to ensure that the desired number of copies now exist. If the number of copies of the logical volume is below the desired number, use the **mklvcopy** command to create additional copies.

- To verify that all logical volume partitions are synchronized and there are no stale partitions, run the following command:

```
lspv hdisk10
```

Check the **STALE PARTITIONS** attribute of the replaced PV to ensure that the count is zero. If there are stale partitions use the **syncvg** command to synchronize the partitions.

Step 5 completes the replacement procedure for a failed PV.

Replacing a failed PV when the configuration does not allow the use of the `replacepv` command

Assume that the failed physical volume, `hdisk0`, and its mirror, `hdisk1`, are part of the `yourvg` volume group.

1. To remove mirror copies from the failed PV, run the following command:

```
unmirrorvg yourvg hdisk0
```

2. If the PV failure occurred on `rootvg`, remove `hdisk0` from the boot list by running the following command:

Note: If your configuration uses boot devices other than `hdisk0` and `hdisk1`, add them to the command syntax.

```
bootlist -om normal hdisk1
```

This step requires that `hdisk1` remains a bootable device in `rootvg`. After completing this step, ensure that `hdisk0` does not appear in output.

3. If the PV failure occurred on `rootvg`, recreate any dedicated dump devices from the failed PV.

If you have a dedicated dump device that was on the failed PV, you can use the **mklv** command to create a new logical volume on an existing PV. Use the **sysdumpdev** command to set the new logical volume as the primary dump device.

4. To undefine the failed PV, run the following command:

Note: Removing the disk device entry will also remove the `/dev/ipldevice` hard link if the failed PV is the PV used to boot the system.

```
reducevg yourvg hdisk0
rmdev -dl hdisk0
```

5. If the failed PV is the most recently used boot device, recreate the `/dev/ipldevice` hard link that was removed in Step 4 by running the following command:

```
ln /dev/rhdisk1 /dev/ipldevice
```

Note the `r` prefixed to the PV name.

To verify that your `/dev/ipldevice` hard link has been recreated, run the following command:

```
ls /dev/ipldevice
```

6. Replace the failed disk.
7. To define the new PV, run the following command:

```
cfgmgr
```

The **cfgmgr** command assigns a PV name to the replacement PV. The assigned PV name is likely to be the same as the PV name previously assigned to the failed PV. In this example, assume that the device `hdisk0` is assigned to the replacement PV.

8. To add the new PV to the volume group, run the following command:

```
extendvg yourvg hdisk0
```

You might encounter the following error message:

```
0516-050 Not enough descriptor space left in this volume group.
Either try adding a smaller PV or use another volume group.
```

If you encounter this error and cannot add the PV to the volume group, you can try to mirror logical volumes to another PV that already exists in the volume group or add a smaller PV. If neither option is possible, you can try to bypass this limitation by upgrading the volume group to a Big-type or Scalable-type volume group using the **chvg** command.

9. Mirror the volume group.

Note: The **mirrorvg** command cannot be used if all of the following conditions exist:

- The target system is a logical partition (LPAR).
- A copy of the boot logical volume (by default, `hd5`) resides on the failed PV.
- The replacement PV's adapter was dynamically configured into the LPAR since the last cold boot.

If all of the above conditions exist, use the **mklvcopy** command to recreate mirror copies for each logical volume as follows:

- a. Create copies of the boot logical volume to ensure that it is allocated to a contiguous series of physical partitions.
- b. Create copies of the remaining logical volumes, and synchronize the copies using the **syncvg** command.
- c. Make the disk bootable by shutting down the LPAR and activating it instead of rebooting using the `shutdown` or `reboot` commands. This shutdown does not have to be done immediately, but it is necessary for the system to boot from the new PV.

Otherwise, create new copies of logical volumes in the volume group using the new PV with the following command:

Note: The **mirrorvg** command disables quorum by default. For rootvg, you will want to use the **-m** option to ensure that the new logical volume copies are mapped to hdisk0 in the same way as the working disk.

```
mirrorvg yourvg hdisk0
```

10. If your configuration holds third copies of some logical volumes, you might need to recreate those copies with the following command:

```
mklvcopy -k
```

11. If the PV failure occurred on rootvg, initialize the boot record by running the following command:

```
bosboot -a
```

12. If the PV failure occurred on rootvg, update the boot list by running the following command:

Note: If your configuration uses boot devices other than hdisk0 and hdisk1, add them to the command.

```
bootlist -om normal hdisk0 hdisk1
```

13. Verify that the procedure was successful.

- To verify that all logical volumes are mirrored to the new PV, run the following command:

```
lslv lvname
```

Check the COPIES attribute of each logical volume affected by the failed PV to ensure that the desired number of copies now exist. If the number of copies of the logical volume is below the desired number, use the **mklvcopy** command to create additional copies.

- To verify that all the logical volume partitions are synchronized, check that there are no stale partitions by running the following command:

```
lspv hdisk0
```

Check the STALE PARTITIONS attribute of the replaced PV to ensure that the count is zero. If there are stale partitions use the **syncvg** command to synchronize the partitions.

If the PV failure occurred on rootvg, use the following steps to verify other aspects of this procedure:

- To verify the boot list, run the following command:

```
bootlist -om normal
```

- To verify the dump device, run the following command:

```
sysdumpdev -l
```

- To verify the list of bootable PVs, run the following command:

```
ipl_varyon -i
```

- To verify the /dev/ipl_device, run the following command:

```
ls -i /dev/rhdisk1 /dev/ipldevice
```

Ensure the output of the **ls** command has the same i-node number for both entries.

This step completes the procedure.

For more information

- *AIX Logical Volume Manager from A to Z: Introduction and Concepts*, available at the Redbooks Web site at www.redbooks.ibm.com.
- *AIX Storage Management*, available at the Redbooks Web site at www.redbooks.ibm.com.

Notifying the administrator when a physical volume is missing

Although AIX logs an error when a physical volume becomes inaccessible, there are circumstances in which an error can go undetected.

For example, when the physical volume is part of a mirrored volume group, users do not notice a problem because a good copy of the data remains accessible. In such cases, automatic notification can alert the administrator to the problem before the users notice any disruption to their work.

The following procedure describes how to set up automatic notification when a physical volume is declared missing. By modifying the following procedure, you can track other errors that are significant to you.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. With root authority, make a backup copy of the `/etc/objrepos/errnotify` ODM file. You can name the backup copy anything you choose. In the following example, the backup copy appends the `errnotify` file name with the current date:

```
cd /etc/objrepos
cp errnotify errnotifycurrent_date
```

2. Use your favorite editor to create a file named `/tmp/pvmiss.add` that contains the following stanza:

```
errnotify:
en_pid = 0
en_name = "LVM_SA_PVMISS"
en_persistenceflg = 1
en_label = "LVM_SA_PVMISS"
en_crcid = 0
en_type = "UNKN"
en_alertflg = ""
en_resource = "LVDD"
en_rtype = "NONE"
en_rclass = "NONE"
en_method = "/usr/lib/ras/pvmiss.notify $1 $2 $3 $4 $5 $6 $7 $8 $9"
```

After you complete all the steps in this article, the error notification daemon will automatically expand the \$1 through \$9 in this script with detailed information from the error log entry within the notification message.

3. Use your favorite editor to create a file named `/usr/lib/ras/pvmiss.notify` with the following contents:

```
#!/bin/ksh
exec 3>/dev/console
print -u3 "?"
print -u3 - "-----"
print -u3 "ALERT! ALERT! ALERT! ALERT! ALERT! ALERT!"
print -u3 ""
print -u3 "Desc: PHYSICAL VOLUME IS MISSING. SEE ERRPT."
print -u3 ""
print -u3 "Error label: $9"
print -u3 "Sequence number: $1"
print -u3 "Error ID: $2"
print -u3 "Error class: $3"
print -u3 "Error type: $4"
print -u3 "Resource name: $6"
print -u3 "Resource type: $7"
print -u3 "Resource class: $8"
print -u3 - "-----"
print -u3 "?"
mail - "PHYSICAL VOLUME DECLARED MISSING" root <<-EOF
-----
ALERT! ALERT! ALERT! ALERT! ALERT! ALERT!
Desc: PHYSICAL VOLUME IS MISSING. SEE ERRPT.
Error label: $9
```

```
Sequence number: $1
Error ID: $2
Error class: $3
Error type: $4
Resource name: $6
Resource type: $7
Resource class: $8
```

EOF

4. Save your file and exit the editor.
5. Set the appropriate permissions on the file you just created. For example:

```
chmod 755 /usr/lib/ras/pvmiss.notify
```
6. Type the following command to add the LVM_SA_PVMISS definition that you created in step 2 to the ODM:

```
odmadd /tmp/pvmiss.add
```

At this point, the system runs the `/usr/lib/ras/pvmiss.notify` script whenever an LVM_SA_PVMISS error occurs. This script sends a message to the console and also sends mail to the root user.

For more information

- The **odmadd** command description in *AIX 5L Version 5.3 Commands Reference, Volume 4*
- "Logical volume storage"

Splitting a mirrored disk from a volume group

Beginning with AIX 5.2, *snapshot* support helps you protect the consistency of your mirrored volume groups from potential disk failure.

Using the snapshot feature, you can split off a mirrored disk or disks to use as a reliable (from the standpoint of the LVM metadata) point-in-time backup of a volume group, and, when needed, reliably reintegrate the split disks into the volume group. In the following procedure, you first split off a mirrored disk from a volume group and then you merge the split-off disk into the original volume group. To further ensure the reliability of your snapshot, file systems must be unmounted and applications that use raw logical volumes must be in a known state (a state from which the application can recover if you need to use the backup).

A volume group cannot be split if any one of the following is true:

- A disk is already missing.
- The last non-stale partition would be on the split-off volume group.
- Any stale partitions exist in the volume group, unless you use the force flag (**-f**) with the **splitvg** command.

Furthermore, the snapshot feature (specifically, the **splitvg** command) cannot be used in enhanced or classic concurrent mode. The split-off volume group cannot be made concurrent or enhanced concurrent and there are limitations to the changes allowed for both the split-off and the original volume group. For details, read the **chvg** command description in *AIX 5L Version 5.3 Commands Reference, Volume 1*.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Ensure that the volume group is fully mirrored and that the mirror exists on a disk or set of disks that contains only this set of mirrors.
2. To enable snapshot support, split off the original volume group (`origVG`) to another disk or set of disks, using the following command:

```
splitvg origVG
```

At this point, you now have a reliable point-in-time backup of the original volume group. Be aware, however, that you cannot change the allocation on the split-off volume group.

3. Reactivate the split-off disk and merge it into the original volume group using the following command:

```
joinvg origVG
```

At this point, the split-off volume group is now reintegrated with the original volume group.

For more information

- Logical volume storage
- The **chvg**, **joinvg**, **recreatevg**, and **splitvg** command descriptions in the *AIX 5L Version 5.3 Commands Reference*.
- "Chapter 2. Mirroring" in *AIX Logical Volume Manager from A to Z: Introduction and Concepts*, an IBM Redbooks publication
- *AIX Storage Management*, an IBM Redbooks publication

Chapter 4. Paging space and virtual memory

AIX uses virtual memory to address more memory than is physically available in the system.

The management of memory pages in RAM or on disk is handled by the Virtual Memory Manager (VMM). Virtual-memory segments are partitioned in units called *pages*. A *paging space* is a type of logical volume with allocated disk space that stores information which is resident in virtual memory but is not currently being accessed. This logical volume has an attribute type equal to paging, and is usually simply referred to as paging space or *swap space*. When the amount of free RAM in the system is low, programs or data that have not been used recently are moved from memory to paging space to release memory for other activities.

Paging space addition and activation

To make paging space available to your system, you must add and activate the paging space.

The total amount of paging space is often determined by trial and error. One commonly used guideline is to double the RAM size and use that figure as a paging space target. To use the Web-based System Manager wizard to increase paging space, select the **Volumes** container, then the **Paging Space** container. From the Selected menu, choose **Increase Paging Space** → **Wizard**.

If you prefer to use SMIT, type one of the following fast paths on the command line:

- To list your current paging space, type: `smit lsp`
- To add paging space, type: `smit mkps`
- To activate paging space, type: `smit swapon`

Paging performance improvement

To improve paging performance, use multiple paging spaces and locate them on separate physical volumes whenever possible.

However, more than one paging space can be located on the same physical volume. Although you can use multiple physical volumes, it is a good idea to select only those disks within the rootvg volume group unless you are thoroughly familiar with your system.

Setting the PSALLOC environment variable for early allocation mode

The operating system uses the *PSALLOC* environment variable to determine the mechanism used for memory and paging space allocation.

The default setting is `late`. For a description of the early and late alternatives for the *PSALLOC* environment variable, see “Comparisons of late and early paging space allocation” on page 89.

The following examples show different ways to change the *PSALLOC* environment variable to `early`. The method you choose depends on how broadly you want to apply the change.

- Type the following command on a shell command line:

```
PSALLOC=early;export PSALLOC
```

This command causes all subsequent commands run from that shell session to run in early allocation mode.

- Add the following command in a shell resource file (`.shrc` or `.kshrc`):

```
PSALLOC=early;export PSALLOC
```

This entry causes all processes in your login session, with the exception of the login shell, to run under early allocation mode. This method also protects the processes from the **SIGKILL** signal mechanism.

- Insert the **putenv** subroutine inside a program to set the *PSALLOC* environment variable to *early*. Using this method, the early allocation behavior takes effect at the next call to the **exec** subroutine.

Changing or removing a paging space

Changing a paging space is easily done with Web-based System Manager or SMIT, but removing a paging space is more risky.

Changing the characteristics of a paging space can be done with Web-based System Manager, or you can type the following SMIT fast path on the command line: `smit chps`.

The procedure to remove a paging space is more risky, especially if the paging space you want to remove is a default paging space, such as `hd6`. A special procedure is required for removing the default paging spaces, because they are activated during boot time by shell scripts that configure the system. To remove one of the default paging spaces, these scripts must be altered and a new boot image must be created.

Attention: Removing default paging spaces incorrectly can prevent the system from restarting. The following procedure is for experienced system managers only.

To remove an existing paging space, use the following procedure:

1. With root authority, deactivate the paging space by typing the following SMIT fast path on the command line:
`smit swapoff`
2. If the paging space you are removing is the default dump device, you must change the default dump device to another paging space or logical volume before removing the paging space. To change the default dump device, type the following command:
`sysdumpdev -P -p /dev/new_dump_device`
3. Remove the paging space by typing the following fast path:
`smit rmps`

hd6 paging space resize and relocation

You might want to reduce or move the default paging space in order to enhance storage system performance by forcing paging and swapping to other disks in the system that are less busy. Reducing or moving the default paging also conserves disk space on `hdisk0`.

Whether moving the paging space or reducing its size, the rationale is the same: move paging space activity to disks that are less busy. The installation default creates a paging logical volume (`hd6`) on drive `hdisk0`, that contains part or all of the busy `/` (root) and `/usr` file systems. If the minimum inter-disk allocation policy is chosen, meaning that all of `/` and a large amount of `/usr` are on `hdisk0`, moving the paging space to a disk that is less busy can significantly improve performance. Even if the maximum inter-disk allocation policy is implemented and both `/` and `/usr` are distributed across multiple physical volumes, your `hdisk2` (assuming three disks) likely contains fewer logical partitions belonging to the busiest file systems. (For more information on inter-disk allocation policies, see *Choosing an inter-disk allocation policy for your system*.)

The following procedures describe how to make the `hd6` paging space smaller and how to move the `hd6` paging space within the same volume group.

Making the `hd6` paging space smaller

The following procedure uses the **chps** command to shrink existing paging spaces, including the primary paging space and the primary and secondary dump device.

The **chps** command calls the **shrinkps** script, which safely shrinks the paging space without leaving the system in an unbootable state. Specifically, the script does the following:

1. Creates a temporary paging space in the same volume
2. Moves information to that temporary space
3. Creates a new, smaller paging space in the same volume
4. Removes the old paging space

For the **chps** command to complete successfully, enough free disk space (space not allocated to any logical volume) must exist to create a temporary paging space. The size of the temporary paging space is equal to amount of space needed to hold all the paged out pages in the old paging space. The minimum size for a primary paging space is 32 MB. The minimum size for any other paging space is 16 MB.

Note: If the following procedure encounters an I/O error, the system might require immediate shutdown and rebooting.

1. Check your logical volume and file system distribution across physical volumes by typing the following command:

```
lspv -l hdiskX
```

Where *hdiskX* is the name of your physical volume.

2. To shrink the paging space size, type the following on the command line:

```
smit chps
```

Note: The primary paging space is hardcoded in the boot record. Therefore, the primary paging space will always be activated when the system is restarted. The **chps** command cannot deactivate the primary paging space.

Priority is given to maintaining an operational configuration. System checks can lead to immediate refusal to shrink the paging space. Errors occurring while the temporary paging space is being created cause the procedure to exit, and the system will revert to the original settings. Other problems are likely to provoke situations that will require intervention by the system administrator or possibly an immediate reboot. Some errors may prevent removal of the temporary paging space. This would normally require non-urgent attention from the administrator.

Attention: If an I/O error is detected on system backing pages or user backing pages by the **swapoff** command within the **shrinkps** script, an immediate shutdown is advised to avoid a possible system crash. At reboot, the temporary paging space is active and an attempt can be made to stop and restart the applications which encountered the I/O errors. If the attempt is successful and the **swapoff** command is able to complete deactivation, the shrink procedure can be completed manually using the **mkps**, **swapoff** and **rmps** commands to create a paging space with the required size and to remove the temporary paging space.

Do not attempt to remove (using **rmps**) or reactivate (using **chps**) a deactivated paging space that was in the I/O ERROR state before the system restart. There is a risk that the disk space will be reused and may cause additional problems.

Moving the hd6 paging space within the same volume group

Moving the default paging space from *hdisk0* to a different disk within the same volume group does not require the system to shut down and reboot.

With root authority, type the following command to move the default (*hd6*) paging space from *hdisk0* to *hdisk2*:

```
migratepv -l hd6 hdisk0 hdisk2
```

Attention: Moving a paging space with the name `hd6` from `rootvg` to another volume group is not recommended because the name is hardcoded in several places, including the second phase of the boot process and the process that accesses the root volume group when booting from removable media. Only the paging spaces in `rootvg` are active during the second phase of the boot process, and having no paging space in `rootvg` could severely affect system boot performance. If you want the majority of paging space on other volume groups, it is better to make `hd6` as small as possible (the same size as physical memory) and then create larger paging spaces on other volume groups (see “Paging space addition and activation” on page 83).

Paging space troubleshooting

The most common problem regarding paging space is caused by running out of allocated space.

The total amount of paging space is often determined by trial and error. One commonly used guideline is to double the RAM size and use that figure as a paging space target. If paging space runs low, processes can be lost, and if paging space runs out, the system can panic. The following signal and error information can help you monitor and resolve or prevent paging space problems.

The operating 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.

Note: If the `low_ps_handling` parameter is set to 2 (under the `vmo` command) and if no process was found to kill (without the **SIGDANGER** handler), the system will send the **SIGKILL** signal to the youngest processes that have a signal handler for the **SIGDANGER** signal.

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.

If you get error messages similar to the following, increase the paging space:

```
INIT: Paging space is low!
```

OR

```
You are close to running out of paging space.  
You may want to save your documents because  
this program (and possibly the operating system)  
could terminate without future warning when the  
paging space fills up.
```

To increase the size of your paging space, see “Changing or removing a paging space” on page 84 or “`hd6` paging space resize and relocation” on page 84.

Virtual Memory Manager

The Virtual Memory Manager (VMM) services memory requests from the system and its applications.

Virtual-memory segments are partitioned in units called *pages*; each page is either located in real physical memory (RAM) or stored on disk until it is needed. AIX uses virtual memory to address more memory than is physically available in the system. The management of memory pages in RAM or on disk is handled by the VMM.

Real memory management in Virtual Memory Manager

In AIX, virtual-memory segments are partitioned into 4096-byte units called pages. Real memory is divided into 4096-byte page frames.

The VMM has two major functions:

- Manage the allocation of page frames
- Resolve references to virtual-memory pages that are not currently in RAM (stored in paging space) or do not yet exist.

To accomplish these functions, the VMM maintains a *free list* of available page frames. The VMM also uses a page-replacement algorithm to determine which virtual-memory pages currently in RAM will have their page frames reassigned to the free list. The page-replacement algorithm takes into account the existence of persistent versus working segments, repaging, and VMM thresholds.

Virtual Memory Manager Free list

The VMM maintains a list of free (unallocated) page frames that it uses to satisfy page faults.

AIX tries to use all of RAM all of the time, except for a small amount which it maintains on the free list. To maintain this small amount of unallocated pages the VMM uses *page outs* and *page steals* to free up space and reassign those page frames to the free list. The virtual-memory pages whose page frames are to be reassigned are selected using the VMM's page-replacement algorithm.

Persistent or working memory segments in Virtual Memory Manager

AIX distinguishes between different types of memory segments. To understand the VMM, it is important to understand the difference between working and persistent segments.

A *persistent segment* has a permanent storage location on disk. Files containing data or executable programs are mapped to persistent segments. When a JFS or JFS2 file is opened and accessed, the file data is copied into RAM. VMM parameters control when physical memory frames allocated to persistent pages should be overwritten and used to store other data.

Working segments are transitory and exist only during their use by a process. Working segments have no permanent disk storage location. Process stack and data regions are mapped to working segments and shared library text segments. Pages of working segments must also occupy disk storage locations when they cannot be kept in real memory. The disk paging space is used for this purpose. When a program exits, all of its working pages are placed back on the free list immediately.

Working segments and paging space in Virtual Memory Manager

Working pages in RAM that can be modified and paged out are assigned a corresponding slot in paging space.

The allocated paging space is used only if the page needs to be paged out. However, an allocated page in paging space cannot be used by another page. It remains reserved for a particular page for as long as that page exists in virtual memory. Because persistent pages are paged out to the same location on disk from which they came, paging space does not need to be allocated for persistent pages residing in RAM.

The VMM has two modes for allocating paging space: *early* and *late*. Early allocation policy reserves paging space whenever a memory request for a working page is made. Late allocation policy only assigns paging space when the working page is actually paged out of memory, which significantly reduces the paging space requirements of the system.

Virtual Memory Manager memory load control facility

When a process references a virtual-memory page that is on disk, because it either has been paged out or has never been read, the referenced page must be paged in, and this might cause one or more pages to be paged out if the number of available (free) page frames is low. The VMM attempts to steal page frames that have not been recently referenced and, therefore, are not likely to be referenced in the near future, using a page-replacement algorithm.

A successful page-replacement keeps the memory pages of all currently active processes in RAM, while the memory pages of inactive processes are paged out. However, when RAM is over-committed, it becomes difficult to choose pages for page out because, they will probably be referenced in the near future by currently running processes. The result is that pages that are likely to be referenced soon might still get paged out and then paged in again when actually referenced. When RAM is over-committed, continuous paging in and paging out, called *thrashing*, can occur. When a system is thrashing, the system spends most of its time paging in and paging out instead of executing useful instructions, and none of the active processes make any significant progress. The VMM has a memory load control algorithm that detects when the system is thrashing and then attempts to correct the condition.

Paging space

A *paging space* is a type of logical volume with allocated disk space that stores information which is resident in virtual memory but is not currently being accessed.

This logical volume has an attribute type equal to paging, and is usually simply referred to as paging space or *swap space*. When the amount of free RAM in the system is low, programs or data that have not been used recently are moved from memory to paging space to release memory for other activities.

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

The amount of paging space required depends on the type of activities performed on the system. If paging space runs low, processes can be lost, and if paging space runs out, the system can panic. When a paging-space low condition is detected, define additional paging space. For instructions, see “Paging space troubleshooting” on page 86.

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.

Paging space allocation policies

The *PSALLOC* environment variable determines which paging space allocation algorithm is used: late or early.

AIX uses two modes for paging space allocation. The default is late. You can switch to an early paging space allocation mode by changing the value of the *PSALLOC* environment variable, but there are several factors to consider before making such a change. When using the early allocation algorithm, in a worst-case scenario, it is possible to crash the system by using up all available paging space.

Comparisons of 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 late allocation algorithm assists in the efficient use of disk resources and supports applications that prefer a sparse allocation algorithm for resource management. This 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.

However, this paging space might never be used, especially on systems with large real memory where paging is rare. Therefore, the late algorithm further delays allocation of paging space 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.

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.

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

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

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

There are different ways to change the *PSALLOC* environment variable to early, depending on how broadly you want to apply the change. (See “Setting the *PSALLOC* environment variable for early allocation mode” on page 83)

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

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

Early allocation mode:

The early allocation algorithm guarantees as much paging space as requested by a memory allocation request. Thus, correct paging space allocation on the system disk is important for efficient operations.

When available paging space drops below a certain threshold, new processes cannot be started and currently running processes might not be able to get more memory. Any processes running under the default late allocation mode become highly vulnerable to the **SIGKILL** signal mechanism. In addition, because the operating system kernel sometimes requires memory allocation, it is possible to crash the system by using up all available paging space.

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

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

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

Using the paging space allocation mode programming interface

The programming interface that controls the paging space allocation mode uses the *PSALLOC* environment variable.

To ensure that an application always runs under the desired mode (with or without early paging space allocation), do the following:

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

Paging space default size

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 16 MB, except for hd6 which can use no less than 64 MB in AIX 4.3 and later.
- Paging space can use no more than 20% of total disk space.
- If real memory is less than 256 MB, paging space is two times real memory.
- If real memory is greater than or equal to 256 MB, paging space is 512 MB.

Paging space file, commands, and options

The `/etc/swapspace` 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. If the paging space size is too large, you can subtract logical partitions from the paging space without rebooting using the `chps -d` command.

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. 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 a different set of options. 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.
rmps	Removes an inactive paging space.
swapoff	Deactivates one or more paging space without rebooting the system. Information in the paging space is moved to other active paging space areas. The deactivated paging space can then be removed using the rmps command.
swapon	Activates a paging space. The swapon command is used during early system initialization to activate the initial paging-space device. During a later phase of initialization, when other devices become available, the swapon command is used to activate additional paging spaces so that paging activity occurs across several devices.

The **paging type** option is required for all logical volume paging spaces.

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.

Chapter 5. File systems

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. For information on the structure of the file system, see “Organization and contents of the file tree” on page 94.

A file system resides on a single logical volume. Every file and directory belongs to a file system within a logical volume. Because of its structure, 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. You can make a point-in-time image of a JFS file system or a JFS2 file system, called a *snapshot* (AIX 5.2 and later).

Note: The maximum number of logical partitions per logical volume is 32,512. For more information on file system logical volume characteristics, see the **chlv** command.

The **mkfs** (make file system) command or the System Management Interface Tool (**smit** command) creates a file system on a logical volume. For more information on managing file systems, see “File system management tasks” on page 102.

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 more information on mounting a file system, see “Mounting” on page 117.

The basic type of file system used by AIX is called the *journaled file system (JFS)*. 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.

Multiple file system types are supported for AIX 5.2, including the journaled file system (JFS) and the enhanced journaled file system (JFS2). For more information on file system types and the characteristics of each type, see “File system types” on page 123.

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 if the system fails
- Maintaining file systems in a consistent state

These tasks should be performed by your system administrator.

Organization and contents of the file tree

The file tree organizes files into directories containing similar information. This organization facilitates remote mounting of directories and files.

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

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

The file tree has the following characteristics:

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

File system structure

It is important to understand the difference between a file system and a directory. A file system is a section of hard disk that has been allocated to contain files. This section of hard disk is accessed by mounting the file system over a directory. After the file system is mounted, it looks just like any other directory to the end user.

However, because of the structural differences between the file systems and directories, the data within these entities can be managed separately.

When the operating system is installed for the first time, it is loaded into a directory structure, as shown in the following illustration.

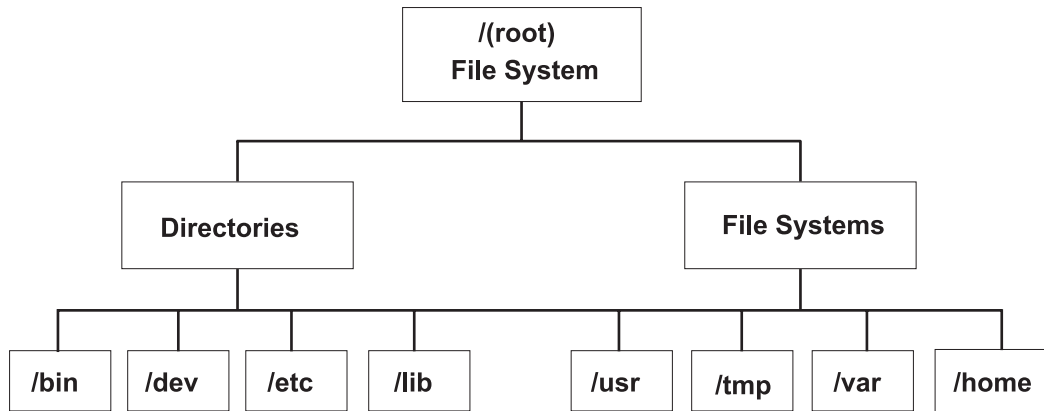


Figure 7. / (root) File System Tree. This tree chart shows a directory structure with the / (root) file system at the top, branching downward to directories and file systems. Directories branch to /bin, /dev, /etc, and /lib. File systems branch to /usr, /tmp, /var, and /home.

The directories on the right (/usr, /tmp, /var, and /home) are all file systems so they have separate sections of the hard disk allocated for their use. These file systems are mounted automatically when the system is started, so the end user does not see the difference between these file systems and the directories listed on the left (/bin, /dev, /etc, and /lib).

On standalone machines, the following file systems reside on the associated devices by default:

/File System	/Device
/dev/hd1	/home
/dev/hd2	/usr
/dev/hd3	/tmp
/dev/hd4	/(root)
/dev/hd9var	/var
/proc	/proc
/dev/hd10opt	/opt

The file tree has the following characteristics:

- Files that can be shared by machines of the same hardware architecture are located in the /usr file system.
- Variable per-client files, for example, spool and mail files, are located in the /var file system.
- The / (root) file system contains files and directories critical for system operation. For example, it contains
 - A device directory (/dev)
 - Mount points where file systems can be mounted onto the root file system, for example, /mnt
- The /home file system is the mount point for users' 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.
- The /proc file system contains information about the state of processes and threads in the system.
- The /opt file system contains optional software, such as applications.

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

/bin	Symbolic link to the /usr/bin 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.
/etc	Contains configuration files that vary for each machine. Examples include: <ul style="list-style-type: none"> • /etc/hosts • /etc/passwd
/export	Contains the directories and files on a server that are for remote clients.
/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, a separate local file system is mounted over the /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.
/lib	Symbolic link to the /usr/lib directory, which contains architecture-independent libraries with names in the form lib*.a .
/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.
/tmp	Serves as a mount point for a file system that contains system-generated temporary files.
/u	Symbolic link to the /home directory.
/usr	Serves as a mount point for a file system containing files that do not change and can be shared by machines (such as executable programs and ASCII documentation). Standalone machines mount 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.
/var	Serves as a mount point for files that vary on each machine. The /var file system is configured as a file system because the files that it contains tend to grow. For example, it is a symbolic link to the /usr/tmp directory, which contains temporary work files.

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

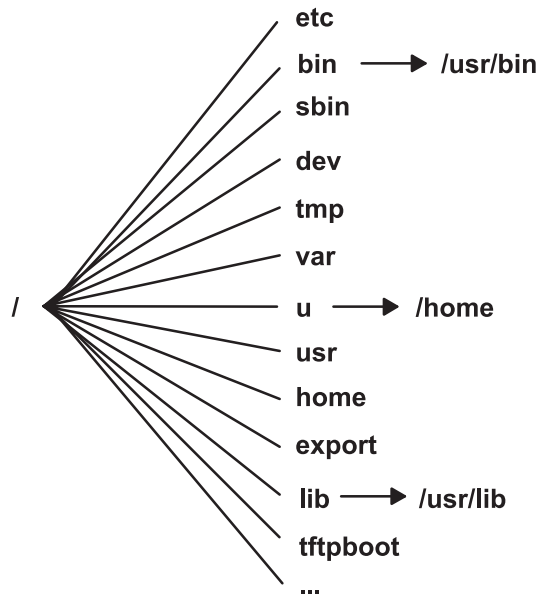


Figure 8. Root File System. This diagram shows the root file system and its subdirectories. The **/bin** subdirectory points to the **/usr/bin** directory. The **/lib** subdirectory points to the **/usr/lib** directory. The **/u** subdirectory points to the **/home** directory.

The 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 “**/var** file system” on page 100 for more information.

/u Symbolic link to the **/home** directory.

/usr Contains files that do not change and can be shared by machines such as executables and ASCII documentation.

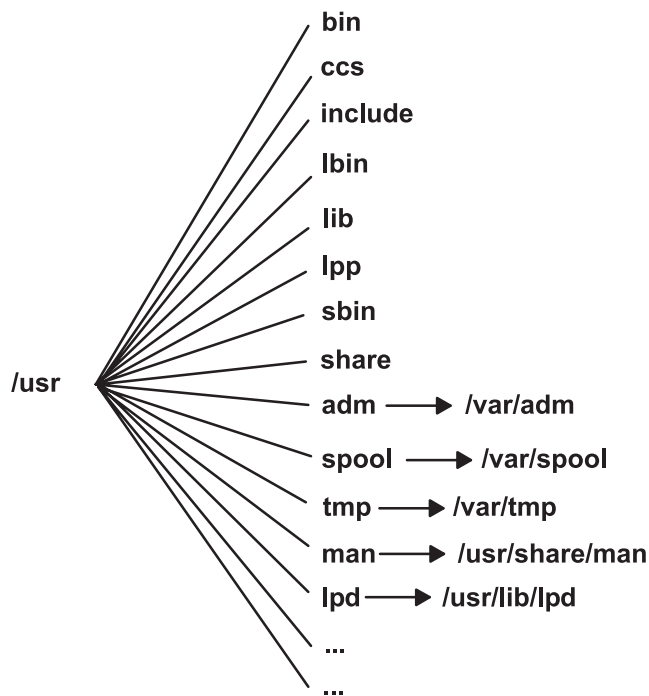
Standalone machines mount the root of a separate local file system over the **/usr** directory. Diskless machines and machines with limited disk resources mount a directory from a remote server over the **/usr** file system. See “**/usr** file system” on page 98 for more information about the file tree mounted over the **/usr** directory.

/home	Serves as a mount point for a file system containing user home directories. The /home file system contains per-user files and directories. In a standalone machine, the /home directory is contained in a separate file system whose root is mounted over the /home directory root file system. In a network, a server might contain user files that are accessible from several machines. In this case, the server copy of the /home directory is remotely mounted onto a local /home file system.
/export	Contains the directories and files on a server that are for remote clients. See “/export directory” on page 101 for more information about the file tree that resides under the /export directory.
/lib	Symbolic link to the /usr/lib directory. See “/usr file system” for more information.
/tftpboot	Contains boot images and boot information for diskless clients.

/usr file system

The **/usr** file system contains executable files that can be shared among machines.

The major subdirectories of the **/usr** directory are shown in the following diagram.



*Figure 9. /usr File System. This diagram shows the major subdirectories of the **/usr** directory, which includes: **/bin**, **/ccs**, **/lib**, **/lpp**, **/adm** and its **/var/adm** subdirectory, and **/man** and its **/usr/share/man** subdirectory.*

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

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

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

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

The following are symbolic links to the **/var** directory:

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

The following are symbolic links to the **/usr/share** and **/usr/lib** directories:

/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

/usr/share directory

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

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

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

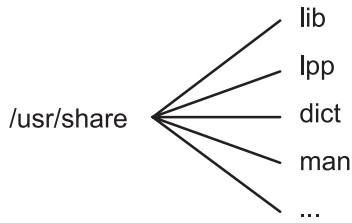


Figure 10. /usr/share Directory.

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

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

/usr/share/man	Contains the manual pages if they have been loaded
/usr/share/dict	Contains the spelling dictionary and its indexes
/usr/share/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

/var file system

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.

Attention: If applications on your system use the **/var** file system extensively, routinely run the **skulker** command or increase the file system size beyond the 4MB **/var** default.

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

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

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

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

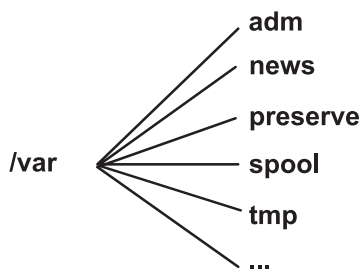


Figure 11. /var Directory. This diagram shows the major subdirectories of the **/var** directory, including **/adm**, **/news**, **/preserve**, **/spool**, and **/tmp**.

/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/tmp .

/export directory

The **/export** directory contains server files exported to clients, such as diskless, dataless, or disk-poor machines.

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

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

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

/export/root

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

/export/exec

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

/export/share

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

/export/home

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

/export/swap

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

/export/dump

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

microcode

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

File system management tasks

A file system is a complete directory structure, including a root directory and any subdirectories and files beneath it.

File systems are confined to a single logical volume. Some of the most important system management tasks are concerning file systems, specifically:

- Allocating space for file systems on logical volumes
- Creating file systems
- Making file system space available to system users
- Monitoring file system space usage
- Backing up file systems to guard against data loss in the event of system failures
- Making a snapshot to capture a consistent block-level image of a file system at a given point in time
- Maintaining file systems in a consistent state.

Following is a list of system management commands that help manage file systems:

backup	Performs a full or incremental backup of a file system
chfs -a	Creates an online backup of a mounted JFS file system
splitcopy	
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 system-wide naming structure so that files and directories in that file system can be accessed
restore	Restores files from a backup
snapshot	Creates a snapshot of a JFS2 file system
umount	Removes a file system from the system-wide naming structure, making the files and directories in the file system inaccessible.

Displaying available space on a file system (df command)

Use the **df** command to display information about total space and available space on a file system. The **FileSystem** parameter specifies the name of the device on which the file system resides, the directory on which the file system is mounted, or the relative path name of a file system.

If you do not specify the **FileSystem** parameter, the **df** command displays information for all currently mounted file systems. If a file or directory is specified, then the **df** command displays information for the file system on which it resides.

Normally, the **df** command uses free counts contained in the superblock. Under certain exceptional conditions, these counts might be in error. For example, if a file system is being actively modified when the **df** command is running, the free count might not be accurate.

See the **df** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Note: On some remote file systems, such as Network File Systems (NFS), the columns representing the available space on the display are left blank if the server does not provide the information.

The following are examples of how to use the **df** command:

- To display information about all mounted file systems, type the following:
df

If your system is configured so the `/`, `/usr`, `/site`, and `/usr/venus` directories reside in separate file systems, the output from the `df` command is similar to the following:

```
Filesystem 512-blocks  free   %used   Iused %Iused  Mounted on
/dev/hd4    20480   13780   32%     805   13%    /
/dev/hd2    385024  15772   95%    27715  28%    /usr
/dev/hd9var 40960   38988   4%      115   1%     /var
/dev/hd3    20480   18972   7%      81    1%     /tmp
/dev/hd1    4096    3724    9%      44    4%     /home
```

- To display available space on the file system in which your current directory resides, type the following:
`df .`

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 systems configuration tasks

When adding or configuring file systems, you can select options in the File Systems container of Web-based System Manager or use the SMIT fast paths.

The SMIT fast paths are provided in the following table:

Table 3. Managing Logical Volumes and File Systems Tasks

Task	SMIT Fast Path
Add a JFS or JFS2	smit crfs
Add a JFS2 to an existing logical volume	smit crjfs2lvstd
Add a JFS to a previously defined logical volume menu	Create logical volume, then smit crjfslv
Change the attributes of a JFS or a JFS2 ^{Note 1}	smit chfs
Check size of a file system	smit fs
Increase size of a file system	JFS: smit chjfs JFS2: smit chjfs2
Decrease size of a file system	JFS2: smit chjfs2

Note: The SMIT Fast Path for **Decrease size of a file system** is only for JFS2.

File systems maintenance tasks

The simplest tasks you might need when maintaining file systems are grouped within this table.

Table 4. Maintaining File Systems Tasks

Task	SMIT Fast Path	Command or File
Backup by name files or directories	smit backfile	backup ^{Note 1}
Create and back up a JFS2 snapshot image	smit backsnap	backsnap ^{Note 1}
List all file systems on a disk	smit lsmntdsk	
List file systems on a removable disk	smit lsmntdsk	
List mounted file systems	smit fs	
Mount a group of file systems ^{Note 5}	smit mountg	mount -t GroupName
Mount a JFS or JFS2 ^{Note 3}	smit mountfs	mount
Mount a JFS2 snapshot	smit mntsnap	mount -v jfs2 -o snapshot Device MountPoint
Remove a JFS or JFS2	smit rmfs	
Remove a JFS2 snapshot	smit rmsnap	snapshot -d SnapshotDevice
Revert a JFS2 file system to a point-in-time snapshot	smit rollbacksnap	rollback [-s] [-v] [-c] snappedFS snapshotObject
Unmount a file system ^{Note 4}	smit umountfs	
Unmount a file system on a removable disk ^{Note 4}	smit umntdsk	
Unmount a group of file systems ^{Note 5}	smit umountg	umount -t GroupName
Manage Enhanced Journaled File Systems quotas	smit j2fsquotas	
Enable or disable quota management	smit j2enablequotas	
Stop/restart quota limits enforcement	smit j2enforcequotas	quotaonloff -v
List quota usage	smit j2repquota	repquota -v
Recalculate current disk block and file usage statistics	smit j2quotacheck	quotacheck -v
Add a limits class	smit j2addlimit	j2edlimit -e
Change/show characteristics of a limits class	smit j2changelimit	
Make a limits class the default limits for a file system	smit j2defaultlimit	
Assign a user or group to a limits class	smit j2assignlimit	
List limits classes for a file system	smit j2listlimits	j2edlimit -l '-u'
Remove a limits class	smit j2removelimit	

Note:

1. For options, refer to the individual commands.
2. Do not change the names of system-critical file systems, which are / (root) on logical volume 4 (hd4), /usr on hd2, /var on hd9var, /tmp on hd3, and /blv on hd5. If you use the hdn convention, start at hd10.

3. Check the file systems before mounting by using the procedure “File system verification” on page 107 or running the **fsck** command.
4. If an unmount fails, it might be because a user or process has an opened file in the file system being unmounted. The **fuser** command lets you find out which user or process might be causing the failure.
5. A file system group is a collection of file systems which have the same value for the **type=** identifier in the **/etc/filesystems** file.

Recovering one or more files from an online JFS2 snapshot

When a file becomes corrupted, you can replace it if you have an accurate copy in an online JFS2 snapshot.

Use the following procedure to recover one or more files from a JFS2 snapshot image:

1. Mount the snapshot. For example:

```
mount -v jfs2 -o snapshot /dev/mysnap1v /home/aaa/mysnap
```

2. Change to the directory that contains the snapshot. For example:

```
cd /home/aaa/mysnap
```

3. Copy the accurate file to overwrite the corrupted one. For example:

```
cp myfile /home/aaa/myfs
```

copies only the file named **myfile**. The following example copies all files at once:

```
cp -R home/aaa/mysnap /home/aaa/myfs
```

For more examples, see the **cp** or **cpio** command descriptions in the *AIX 5L Version 5.3 Commands Reference, Volume 1*.

File systems on CD-ROM and DVD disks

CDs and DVDs are not automatically mounted, but this feature can be enabled.

To enable this feature, use the **cdmount** command to mount the CDRFS or UDFS file system, for example:

```
cdmount cd0
```

You can manually mount a read/write UDFS with the following command:

```
mount -V udfs DevName MtPt
```

Where *DevName* is the name of the DVD drive and *MtPt* is the mount point for the file system.

Using file systems on read/write optical media

CDRFS and JFS file systems can be used on read/write optical media.

A CD-ROM file system (CDRFS) can be stored on read/write optical media, provided that the optical media is write-protected, as well as on a CD-ROM. The following table tells you how to add, mount, or unmount a CDRFS on read/write optical media. You must specify the following information when mounting the file system:

Device name	Defines the name of device containing the media.
Mount point	Specifies the directory where the file system will be mounted.
Automatic mount	Specifies whether the file system will be mounted automatically at system restart.

CDRFS on Optical Media Tasks		
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Adding a CDRFS ¹	smit crcdrfs	<ol style="list-style-type: none"> 1. Add the file system: crfs -v cdrfs -p ro -d DeviceName -m MountPoint -A AutomaticMount 2. Mount the file system: mount MountPoint
Removing a CDRFS ²	<ol style="list-style-type: none"> 1. Unmount the file system: smit umountfs 2. Remove the file system: smit rmcdrfs 	<ol style="list-style-type: none"> 1. Unmount the file system: umount FileSystem 2. Remove the file system: rmfs MountPoint

Note:

- Make sure the read/write optical media is write-protected.
- A CDRFS file system must be unmounted before the read/write optical media can be removed.

A JFS provides a read/write file system on optical media similar to those on a hard disk. You must have system authority to create or import a read/write file system on read/write optical media (that is, your login must belong to the system group) and you must have the following information:

Volume group name

Specifies the name of the volume group

Device name

Specifies the logical name of the read/write optical drive

Mount point

Specifies the directories where the file systems will be mounted

Automatic mount

Specifies whether the file system will be mounted automatically at system restart

Note:

- Any volume group created on read/write optical media must be self contained on that media. Volume groups cannot go beyond one read/write optical disk.
- When accessing a previously created journaled file system, the volume group name does not need to match the one used when the volume group was created.

JFS on Optical Media Tasks		
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Add a JFS	<ol style="list-style-type: none"> 1. Insert optical disk into drive. 2. Create a volume group (if necessary): smit mkvg 3. Create a journaled file system: smit crfs 	<ol style="list-style-type: none"> 1. Insert optical disk into drive. 2. Create a volume group (if necessary): mkvg -f -y VGName -d 1 DeviceName 3. Create a journaled file system: crfs -v jfs -g VGName -a size=SizeFileName -m MountPoint -A AutomaticMount -p rw 4. Mount the file system: mount MountPoint

JFS on Optical Media Tasks		
Accessing previously created JFS ^{Note 1}	<ol style="list-style-type: none"> 1. Insert optical disk into drive. 2. Import the volume group: smit importvg 	<ol style="list-style-type: none"> 1. Insert optical disk into drive. 2. Import the volume group: importvg -y VGName DeviceName 3. Mount the file system: mount MountPoint
Removing a JFS ^{Note 2}	<ol style="list-style-type: none"> 1. Unmount the file system: smit umountfs 2. Remove the file system: smit rmjfs 	<ol style="list-style-type: none"> 1. Unmount the file system: umount FileSystem 2. Remove the file system: rmfs MountPoint

Note:

- This procedure is required whenever inserting media containing journaled file systems.
- Removing a journaled file system destroys all data contained in that file system and on the read/write optical media.

File system verification

Inconsistencies can occur in file systems when the system is stopped while file systems remained mounted or when a disk is damaged. In such circumstances, it is important to verify file systems before mounting them.

Also verify your file systems in the following circumstances:

- After a malfunction; for example, if a user cannot change directories to a directory that has that user's permissions (uid)
- Before backing up file systems to prevent errors and possible restoration problems
- At installation or system boot to make sure that there are no operating system file errors

Checking a user-defined file system

To check a user-defined file system, perform the following steps.

1. Unmount the user-defined file system being checked.
2. Ensure you have write permission on files in the file system. Otherwise, the **fsck** cannot repair damaged files even if you answer Yes to repair prompts.
3. Use the **smit fsck** fast path to access the **Verify a File System** menu.
4. Do one of the following:
 - Specify the name of an individual file system to check in the **NAME of file system** field, or
 - Select a general file system type to check, such as a journaled file system (JFS) in the **TYPE of file system** field.
5. If you want to limit your check to the most likely candidates, specify Yes in the **FAST check?** field. The fast-check option checks only those file systems that are likely to have inconsistencies such as the file systems that were mounted when the system stopped at some point in the past.
6. Specify the name of a temporary file on a file system not being checked in the **SCRATCH file** field.
7. Start the file system check.

Checking root and /usr file systems

To run the **fsck** command on / or /usr file system, you must shut down the system and reboot it from removable media because the / (root) and /usr file systems cannot be unmounted from a running system.

The following procedure describes how to run **fsck** on the / and /usr file systems from the maintenance shell.

1. With root authority, shut down your system.
2. Boot from your installation media.
3. From the **Welcome** menu, choose the **Maintenance** option.
4. From the **Maintenance** menu, choose the option to access a volume group.
5. Choose the rootvg volume group. A list of logical volumes that belong to the volume group you selected is displayed.
6. Choose **2** to access the volume group and to start a shell before mounting file systems. In the following steps, you will run the **fsck** command using the appropriate options and file system device names. The **fsck** command checks the file system consistency and interactively repairs the file system. The / (root) file system device is **/dev/hd4** and the **/usr** file system device is **/dev/hd2**.
7. To check / file system, type the following:

```
$ fsck -y /dev/hd4
```

The **-y** flag is recommended for less experienced users (see the **fsck** command).
8. To check the **/usr** file system, type the following:

```
$ fsck -y /dev/hd2
```
9. To check other file systems in the rootvg, type the **fsck** command with the appropriate device names. The device for **/tmp** is **/dev/hd3**, and the device for **/var** is **/dev/hd9var**.
10. When you have completed checking the file systems, reboot the system.

Disk overflows

A disk overflow occurs when too many files fill up the allotted space. This can be caused by a runaway process that creates many unnecessary files.

You can use the following procedures to correct the problem:

Note: You must have root user authority to remove processes other than your own.

Identifying problem processes

Use this procedure to isolate problem processes.

1. To check the process status and identify processes that might be causing the problem, type:

```
ps -ef | pg
```

The **ps** command shows the process status. The **-e** flag writes information about all processes (except kernel processes), and the **-f** flag generates a full listing of processes including what the command name and parameters were when the process was created. The **pg** command limits output to a single page at a time, so information does not scroll too quickly off the screen.

Check for system or user processes that are using excessive amounts of a system resource, such as CPU time. System processes such as **sendmail**, **routed**, and **lpd** seem to be the system processes most prone to becoming runaways.

2. To check for user processes that use more CPU than expected, type:

```
ps -u
```

3. Note the process ID (PID) of each problem process.

Terminating a process

You can terminate problem processes.

Use the following procedure to terminate a problem process:

1. Terminate the process that is causing the problem by typing:

```
kill -9 PID
```

Where *PID* is the ID of the problem process.

2. Remove the files the process has been making by typing:

```
rm file1 file2 file3
```

Where *file1 file2 file3* represents names of process-related files.

Reclamation of file space without terminating a process

To reclaim the blocks allocated to an active file without terminating the process, redirect the output of another command to the file. The data redirection truncates the file and reclaims the blocks of memory.

When the active file is removed from the file system, the blocks allocated to the file remain allocated until the last open reference is removed, either as a result of the process closing the file or because of the termination of the processes that have the file open. If a runaway process is writing to a file and the file is removed, the blocks allocated to the file are not freed until the process terminates.

For example:

```
$ ls -l
total 1248
-rwxrwxr-x    1 web  staff  1274770 Jul 20 11:19 datafile
$ date > datafile
$ ls -l
total 4
-rwxrwxr-x    1 web  staff      29 Jul 20 11:20 datafile
```

The output of the **date** command replaced the previous contents of the **datafile** file. The blocks reported for the truncated file reflect the size difference from 1248> to 4. If the runaway process continues to append information to this newly truncated file, the next **ls** command produces the following results:

```
$ ls -l
total 8
-rxwxr-x    1 web  staff  1278866 Jul 20 11:21 datafile
```

The size of the **datafile** file reflects the append done by the runaway process, but the number of blocks allocated is small. The **datafile** file now has a hole in it. File holes are regions of the file that do not have disk blocks allocated to them.

/ (root) overflow

Check the following when the root file system (*/*) has become full.

- Use the following command to read the contents of the **/etc/security/failedlogin** file:

```
who /etc/security/failedlogin
```

The condition of TTYs respawning too rapidly can create failed login entries. To clear the file after reading or saving the output, execute the following command:

```
cp /dev/null /etc/security/failedlogin
```

- Check the **/dev** directory for a device name that is typed incorrectly. If a device name is typed incorrectly, such as **rmt0** instead of **rmt0**, a file will be created in **/dev** called **rmt0**. The command will normally proceed until the entire root file system is filled before failing. **/dev** is part of the root (*/*) file system. Look for entries that are not devices (that do not have a major or minor number). To check for this situation, use the following command:

```
cd /dev
ls -l | pg
```

In the same location that would indicate a file size for an ordinary file, a device file has two numbers separated by a comma. For example:

```
crw-rw-rw-  1 root  system  12,0 Oct 25 10:19 rmt0
```

If the file name or size location indicates an invalid device, as shown in the following example, remove the associated file:

```
crw-rw-rw-  1 root    system  9375473 Oct 25 10:19 rmt0
```

Note:

- Do not remove valid device names in the **/dev** directory. One indicator of an invalid device is an associated file size that is larger than 500 bytes.
- If system auditing is running, the default **/audit** directory can rapidly fill up and require attention.
- Check for very large files that might be removed using the **find** command. For example, to find all files in the root (**/**) directory larger than 1 MB, use the following command:

```
find / -xdev -size +2048 -ls |sort -r +6
```

This command finds all files greater than 1 MB and sorts them in reverse order with the largest files first. Other flags for the **find** command, such as **-newer**, might be useful in this search. For detailed information, see the command description for the **find** command.

Note: When checking the root directory, major and minor numbers for devices in the **/dev** directory will be interspersed with real files and file sizes. Major and minor numbers, which are separated by a comma, can be ignored.

Before removing any files, use the following command to ensure a file is not currently in use by a user process:

```
fuser filename
```

Where *filename* is the name of the suspect large file. If a file is open at the time of removal, it is only removed from the directory listing. The blocks allocated to that file are not freed until the process holding the file open is killed.

Resolving overflows in the /var file system

Check the following when the **/var** file system has become full.

- You can use the **find** command to look for large files in the **/var** directory. For example:

```
find /var -xdev -size +2048 -ls | sort -r +6
```

For detailed information, see the command description for the **find** command.

- Check for obsolete or leftover files in **/var/tmp**.
- Check the size of the **/var/adm/wtmp** file, which logs all logins, rlogins and telnet sessions. The log will grow indefinitely unless system accounting is running. System accounting clears it out nightly. The **/var/adm/wtmp** file can be cleared out or edited to remove old and unwanted information. To clear it, use the following command:

```
cp /dev/null /var/adm/wtmp
```

To edit the **/var/adm/wtmp** file, first copy the file temporarily with the following command:

```
/usr/sbin/acct/fwtmp < /var/adm/wtmp >/tmp/out
```

Edit the **/tmp/out** file to remove unwanted entries then replace the original file with the following command:

```
/usr/sbin/acct/fwtmp -ic < /tmp/out > /var/adm/wtmp
```

- Clear the error log in the **/var/adm/ras** directory using the following procedure. The error log is never cleared unless it is manually cleared.

Note: Never use the **cp /dev/null** command to clear the error log. A zero-length **errlog** file disables the error logging functions of the operating system and must be replaced from a backup.

1. Stop the error daemon using the following command:

```
/usr/lib/errstop
```

2. Remove or move to a different filesystem the error log file by using one of the following commands:

```
rm /var/adm/ras/errlog
```

or

```
mv /var/adm/ras/errlog filename
```

Where *filename* is the name of the moved errlog file.

Note: The historical error data is deleted if you remove the error log file.

- Restart the error daemon using the following command:

```
/usr/lib/errdemon
```

Note: Consider limiting the errlog by running the following entries in **cron**:

```
0 11 * * * /usr/bin/errclear -d S,0 30
0 12 * * * /usr/bin/errclear -d H 90
```

- Check whether the **trcfile** file in this directory is large. If it is large and a trace is not currently being run, you can remove the file using the following command:

```
rm /var/adm/ras/trcfile
```

- If your dump device is set to hd6 (which is the default), there might be a number of **vmcore*** files in the **/var/adm/ras** directory. If their file dates are old or you do not want to retain them, you can remove them with the **rm** command.

- Check the **/var/spool** directory, which contains the queueing subsystem files. Clear the queueing subsystem using the following commands:

```
stopsrc -s qdaemon
rm /var/spool/lpd/qdir/*
rm /var/spool/lpd/stat/*
rm /var/spool/qdaemon/*
startsrc -s qdaemon
```

- Check the **/var/adm/acct** directory, which contains accounting records. If accounting is running, this directory may contain several large files. Information on how to manage these files is in Chapter 11, "System accounting," on page 245.
- Check the **/var/preserve** directory for terminated **vi** sessions. Generally, it is safe to remove these files. If a user wants to recover a session, you can use the **vi -r** command to list all recoverable sessions. To recover a specific session, use **vi -r filename**.
- Modify the **/var/adm/sulog** file, which records the number of attempted uses of the **su** command and whether each was successful. This is a flat file and can be viewed and modified with a favorite editor. If it is removed, it will be recreated by the next attempted **su** command. Modify the **/var/tmp/snmpd.log**, which records events from the **snmpd** daemon. If the file is removed it will be recreated by the **snmpd** daemon.

Note: The size of the **/var/tmp/snmpd.log** file can be limited so that it does not grow indefinitely. Edit the **/etc/snmpd.conf** file to change the number (in bytes) in the appropriate section for size.

Fixing a user-defined file system overflow

Use this procedure to fix an overflowing user-defined file system.

- Remove old backup files and core files. The following example removes all ***.bak**, **.*.bak**, **a.out**, **core**, *****, or **ed.hup** files.

```
find / \( -name "*.bak" -o -name core -o -name a.out -o \
    -name "...*" -o -name ".*.bak" -o -name ed.hup \) \
    -atime +1 -mtime +1 -type f -print | xargs -e rm -f
```

- To prevent files from regularly overflowing the disk, run the **skulker** command as part of the **cron** process and remove files that are unnecessary or temporary.

The **skulker** command purges files in **/tmp** directory, files older than a specified age, **a.out** files, core files, and **ed.hup** files. It is run daily as part of an accounting procedure run by the **cron** command during off-peak periods (assuming you have turned on accounting).

The **cron** daemon runs shell commands at specified dates and times. Regularly scheduled commands such as **skulker** can be specified according to instructions contained in the **crontab** files. Submit **crontab** files with the **crontab** command. To edit a **crontab** file, you must have root user authority.

For more information about how to create a **cron** process or edit the **crontab** file, refer to “Setting up an accounting system” on page 245.

Fix other file systems and general search techniques

Use the **find** command with the **-size** flag to locate large files or, if the file system recently overflowed, use the **-newer** flag to find recently modified files.

To produce a file for the **-newer** flag to find against, use the following touch command:

```
touch mmdhmm filename
```

Where *mm* is the month, *dd* is the date, *hh* is the hour in 24-hour format, *mm* is the minute, and *filename* is the name of the file you are creating with the **touch** command.

After you have created the touched file, you can use the following command to find newer large files:

```
find /filesystem_name -xdev -newer touch_filename -ls
```

You can also use the **find** command to locate files that have been changed in the last 24 hours, as shown in the following example:

```
find /filesystem_name -xdev -mtime 0 -ls
```

Fixing a damaged file system

File systems can get corrupted when the i-node or superblock information for the directory structure of the file system gets corrupted.

This corruption can be caused by a hardware-related ailment or by a program that gets corrupted that accesses the i-node or superblock information directly. (Programs written in assembler and C can bypass the operating system and write directly to the hardware.) One symptom of a corrupt file system is that the system cannot locate, read, or write data located in the particular file system.

To fix a damaged file system, you must diagnose the problem and then repair it. The **fsck** command performs low-level diagnosis and repairs.

The following is the procedure for fixing a damaged file system:

1. With root authority, unmount the damaged file system using one of the following SMIT fast paths: **smit unmountfs** (for a file system on a fixed disk drive) or **smit unmntdsk** (for a file system on a removeable disk).
2. Assess file system damage by running the **fsck** command. In the following example, the **fsck** command checks the unmounted file system located on the **/dev/myfilelv** device:

```
fsck /dev/myfilelv
```

The **fsck** command checks and interactively repairs inconsistent file systems. Normally, the file system is consistent, and the **fsck** command merely reports on the number of files, used blocks, and free blocks in the file system. If the file system is inconsistent, the **fsck** command displays information about the inconsistencies found and prompts you for permission to repair them. The **fsck** command is conservative in its repair efforts and tries to avoid actions that might result in the loss of valid data. In certain cases, however, the **fsck** command recommends the destruction of a damaged file. Refer to the **fsck** command description in *AIX 5L Version 5.3 Commands Reference, Volume 2* for a list of inconsistencies that this command checks for.

3. If the file system cannot be repaired, restore it from backup.

Attention: Restoring a file system from a backup destroys and replaces any file system previously stored on the disk.

To restore the file system from backup, use the SMIT fastpath **smit restfilesys** or the series of commands shown in the following example:

```
mkfs /dev/myfilelv
mount /dev/myfilelv /myfilesys
cd /myfilesys
restore -r
```

In this example, the **mkfs** command makes a new file system on the device named **/dev/myfilelv** and initializes the volume label, file system label, and startup block. The **mount** command establishes **/dev/myfilelv** as the mountpoint for **myfilesys** and the **restore** command extracts the file system from the backup.

If your backup was made using incremental file system backups, you must restore the backups in increasing backup-level order (for example, 0, 1, 2). For more information about restoring a file system from backup, refer to "Restoring from Backup Image Individual User Files".

When using **smit restfilesys** to restore an entire file system, enter the target directory, restore device (other than **/dev/rfd0**), and number of blocks to read in a single input operation.

Comparing file systems on different machines

When file systems that exist on different machines should be identical but you suspect one is damaged, you can compare the file systems.

The following procedure describes how to compare the attributes of a file system that resides on your current host (in this scenario, called *orig_host*) to the same file system on a remote host.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Log in to the remote host as the root user. For example:

```
tn juniper.mycompany.com
```

```
AIX Version 5
(C) Copyrights by IBM and by others 1982, 2002.
login: root
root's Password:
```

2. Using your favorite editor, edit the remote host's **.rhosts** file to add a stanza that allows the root user to execute secure remote commands. Use the following format for the new stanza:

```
orig_host root
```

The resulting **.rhosts** file might look similar to the following:

```
NIM.mycompany.com root
nim.mycompany.com root
host.othernetwork.com root
orig_host.mycompany.com root
```

3. Save your changes and exit the remote connection.
4. With root authority on *orig_host*, create another file using your favorite editor. For this scenario, the new file is named **compareFS**. For example:

```
vi compareFS
```

5. Insert the following text in this file, where *FSname* is the name of the file system that you want to compare, and *remote_host* is the name of the host on which the comparison file system resides:

```
FSname -> remote_host
install -v ;
```

Note: In the **install** command line of this file, there must be a space between the **-v** parameter and the semicolon (;).

For example:

```
/home/jane/* -> juniper.mycompany.com
install -v ;
```

6. Save the file and exit the editor. The **compareFS** file is used as the *distfile* for the **rdist** command in the following step.
7. Type the following at the command prompt:

```
/usr/bin/rdist -f compareFS
```

Or, if you expect a significant amount of output from the comparison, send the output to a file name. For example:

```
/usr/bin/rdist -f compareFS > compareFS_output
```

The output lists any differences between the file systems.

For more information

- The **rdist** command description in *AIX 5L Version 5.3 Commands Reference, Volume 4*.
- The ".rhosts File Format for TCP/IP" in the *AIX 5L Version 5.3 Files Reference*.
- "Understanding the Secure Remote Commands" in the *Security*.

Reducing the size of a file system in your root volume group

The simplest way to reduce *all* file systems to their minimum size is to set the **SHRINK** option to **yes** when restoring the base operating system from backup.

The simplest way to reduce *all* file systems to their minimum size is to set the **SHRINK** option to **yes** when restoring the base operating system from backup. The **SHRINK** option and the following scenario cannot be used in tandem. If you set the **SHRINK** option to **yes** after doing the following procedure, the installation overrides your changes to the */image.data* file.

This scenario leads you through a manual process to reduce the size of a selected **rootvg** file system. You will identify a file system that is not using all of its allocated disk space and then reallocate based on the amount of space the file system actually uses, thus freeing more space for the root volume group's use. As part of this procedure, you will back up your volume groups and reinstall the operating system, using the revised allocations.

Attention: This procedure requires shutting down and reinstalling the base operating system. Whenever you reinstall any operating system, schedule your downtime when it least impacts your workload to protect yourself from a possible loss of data or functionality. Before reinstalling the operating system, ensure you have reliable backups of your data and any customized applications or volume groups.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Create a separate backup of all file systems that are *not* contained in the **rootvg**. The separate backup helps ensure the integrity of all your file systems.
2. With root authority, check which file systems in your root volume group are not using their allocated disk space by typing the following command:

```
df -k
```

The **-k** flag displays the file-system sizes in kilobytes. Your result will look similar to the following:

Filesystem	1024-blocks	Free	%Used	Iused	%Iused	Mounted on
/dev/hd4	196608	4976	98%	1944	2%	/
/dev/hd2	1769472	623988	65%	36984	9%	/usr

/dev/hd9var	163840	65116	61%	676	2%	/var
/dev/hd3	65536	63024	4%	115	1%	/tmp
/dev/hd1	49152	8536	83%	832	7%	/home
/proc	-	-	-	-	-	/proc
/dev/hd10opt	32768	26340	20%	293	4%	/opt

Looking at these results, you notice a large number of free blocks and a fairly low percentage of use associated with the file system that is mounted on **/usr**. You decide you can release a significant number of blocks by reducing the number of partitions allocated to the **/usr** file system.

3. Check the contents of the **/etc/filesystems** file to ensure that all file systems in the rootvg are mounted. If not, they will not be included in the reinstalled system.
4. Create an **/image.data** file, which lists all the active file systems in the rootvg that are included in the installation procedure, by typing the following command:

```
mkszfile
```

5. Open the **/image.data** file in your favorite editor.
6. Search for the **usr** text string to locate the **lv_data** stanza that pertains to the **/usr** file system. Use numbers from this stanza as a base to determine how much you can reduce the **/usr** file system's number of logical partitions. The default size of each additional logical partition is defined in the **PP_SIZE** entry of the **/image.data** file. Your **/image.data** file would look similar to the following:

```
lv_data:
VOLUME_GROUP= rootvg
LV_SOURCE_DISK_LIST= hdisk0
LV_IDENTIFIER= 00042345d300bf15.5
LOGICAL_VOLUME= hd2
VG_STAT= active/complete
TYPE= jfs
MAX_LPS= 32512
COPIES= 1
LPs= 108
STALE_PPs= 0
INTER_POLICY= minimum
INTRA_POLICY= center
MOUNT_POINT= /usr
MIRROR_WRITE_CONSISTENCY= on/ACTIVE
LV_SEPARATE_PV= yes
PERMISSION= read/write
LV_STATE= opened/syncd
WRITE_VERIFY= off
PP_SIZE= 16
SCHED_POLICY= parallel
PP= 108
BB_POLICY= relocatable
RELOCATABLE= yes
UPPER_BOUND= 32
LABEL= /usr
MAPFILE=
LV_MIN_LPS= 70
STRIPE_WIDTH=
STRIP_SIZE=
```

The number of logical partitions devoted to this logical volume is 108 (LPs=108).

7. Determine the number of logical partitions needed by the existing data in the **/usr** file system by using your result from step 2. You can display the existing file sizes specifically for the **/usr** file system by using the following command:

```
df -k /usr
```

The result repeats the numbers (in kilobytes) you received for the **/usr** file system in step 2. For example:

Filesystem	1024-blocks	Free	%Used	Iused	%Iused	Mounted on
/dev/hd2	1769472	623988	65%	36984	9%	/usr

- a. Subtract the amount of free space from the total number of 1024-blocks allocated:

1769472 - 623988 = 1145484

- b. Add an estimate of the space you might need to accommodate any future growth expected in this file system. For this example, add 200000 to the result.

1145484 + 200000 = 1345484

- c. Divide the result by the logical-partition size in bytes (16*1024) to determine the minimum number of logical partitions you need.

1345484 / 16384 = 82.121826171875

Use this result, rounded upward, to redefine the number of logical partitions needed (LPs=83).

8. In your **image.data** file, change the LPs field from 108 to 83.
9. Find the **fs_data** stanza that pertains to the **/usr** file system. Your **fs_data** stanza will look similar to the following:

```
fs_data:
FS_NAME= /usr
FS_SIZE= 3538944
FS_MIN_SIZE= 2290968
FS_LV= /dev/hd2
FS_FS= 4096
FS_NBPI= 4096
FS_COMPRESS= no
FS_BF= false
FS_AGSIZE= 8
```

10. Calculate the file-system size (FS_SIZE) by multiplying the physical partition size (PP_SIZE) by 2 (the number of 512-byte blocks used by the physical partitions) by the number of logical partitions (LPs). Given the values used in this example, the calculation is:

```
PP_SIZE * 512 blocks * LPs = FS_SIZE
16384 * 2 * 83 = 2719744
```

11. In your **image.data** file, change the FS_SIZE field from 3538944 to 2719744.
12. Calculate the minimum file-system size (FS_MIN_SIZE) based on the actual size of the current data used by the **/usr** file system, as follows:
 - a. Calculate the minimum number of partitions needed. Given the values used in this example, the calculation is:

```
size_in_use (see step 7a) / PP_SIZE = partitions
1145484 / 16384 = 69.914794921875
```

- b. Calculate the minimum size required by that number of partitions. Rounding the previous calculation results upward to 70, the calculation is:

```
PP_SIZE * 512 blocks * partitions = FS_MIN_SIZE
16384 * 2 * 70 = 2293760
```

13. In your **image.data** file, change the FS_MIN_SIZE field from 2290968 to 2293760.
14. Save your edits and exit the editor.
15. Unmount all file systems that are not in the rootvg volume group.
16. If you have any user-defined volume groups, type the following commands to vary off and export them:

```
varyoffvg VGName
exportvg VGName
```

17. With a tape in the tape drive, type the following command to initiate a complete system backup:

```
mksysb /dev/rmt0
```

This type of backup includes the file-system size information you specified in the **/image.data** file, which will be used later to reinstall your system with the new file-system sizes.

Note: To initiate this backup, you must run the **mksysb** command from the command line. If you use a system management tool, such as SMIT, the backup creates a new **image.data** file, overwriting the changes you have made.

18. Use this backup to reinstall the operating system using the **Install With Current System Settings** option. During the installation, check that the following options are set appropriately:

- **Use Maps** must be set to **no**
- **Shrink the File Systems** must be set to **no**

If you need more information about the installation procedure, see the Installing system backups.

19. After the operating system is installed, reboot the system in Normal mode. At this point, the **/usr** file system is resized, but your user-defined file systems are not available.

20. Mount all file systems by typing the following command:

```
mount all
```

If you receive Device Busy messages about file systems that are already mounted, you can ignore these messages.

At this point, your **/usr** file system is resized, your root volume group has more free space, and your file systems are usable.

For more information

- Logical volume storage
- Creating a Root Volume Group Backup to Tape or File and Installing a System Backup on the Source Machine
- The **/image.data** file description
- The **mkszfile** and **mksysb** command descriptions

Mounting

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.

Mount points

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

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

When a file system is mounted over a directory, the permissions of the root directory of the mounted file system take precedence over the permissions of the mount point. The one exception involves the **..** (**dot**

dot) parent directory entry in the mounted-over directory. In order for the operating system to access the new file system, the mount point parent directory information must be available.

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

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

```
pwd: Permission denied
```

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

Mounting file systems, directories, and files

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

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

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

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

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

Automatic mount control

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. For more information about the **/etc/filesystems** file, see **/etc/filesystems**.

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=false
```

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

Mount security for diskless workstations

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

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

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

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

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

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

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

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

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

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

Exporting the **/export/root** Directory

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

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

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

Exporting the **/export/exec** Directory

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

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

Exporting the **/export/share** Directory

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

Exporting the **/export/home** Directory

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

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

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

- You can mount the **/home** directory on the server over the **/home** directory of the client. In this case, the **/home** directory is exported with read/write permissions and without root access. To ensure system security, mount the **/home** directory on both the server and client with the **nosuid** and **nodev** options of the **mount** command.

- Alternatively, you can mount on the client each **/home/UserName** directory on the server over the **/home/Username** directory on the client so users can log in to different machines and still have access to their home directories. In this case, the **/home/Username** directories on the server and clients are both mounted with the **nousid** and **nodev** options of the **mount** command.

Exporting the **/export/dump** Directory

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

Exporting the **/export/swap** Directory

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

Diskless mounts

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

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

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

For more information about the **/export** directory, see “/export directory” on page 101.

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

Exporting the **/export/root** Directory

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

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

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

Exporting the **/export/exec** Directory

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

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

Exporting the **/export/share** Directory

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

Exporting the **/export/home** Directory

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

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

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

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

Exporting the **/export/dump** Directory

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

Exporting the **/export/swap** Directory

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

Fixing a corrupted magic number in the file system superblock

If the superblock of a file system is damaged, the file system cannot be accessed. You can fix a corrupted magic number in the file system superblock.

Most damage to the superblock cannot be repaired. The following procedure describes how to repair a superblock in a JFS file system when the problem is caused by a corrupted magic number. If the primary superblock is corrupted in a JFS2 file system, use the **fsck** command to automatically copy the secondary superblock and repair the primary superblock.

In the following scenario, assume **/home/myfs** is a JFS file system on the physical volume **/dev/lv02**.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Unmount the **/home/myfs** file system, which you suspect might be damaged, using the following command:

```
umount /home/myfs
```
2. To confirm damage to the file system, run the **fsck** command against the file system. For example:

```
fsck -p /dev/lv02
```

If the problem is damage to the superblock, the **fsck** command returns one of the following messages:
fsck: Not an AIXV5 file system

OR

Not a recognized filesystem type

3. With root authority, use the **od** command to display the superblock for the file system, as shown in the following example:

```
od -x -N 64 /dev/lv02 +0x1000
```

Where the **-x** flag displays output in hexadecimal format and the **-N** flag instructs the system to format no more than 64 input bytes from the offset parameter (+), which specifies the point in the file where the file output begins. The following is an example output:

```
0001000 1234 0234 0000 0000 0000 4000 0000 000a
0001010 0001 8000 1000 0000 2f6c 7633 0000 6c76
0001020 3300 0000 000a 0003 0100 0000 2f28 0383
0001030 0000 0001 0000 0200 0000 2000 0000 0000
0001040
```

In the preceding output, note the corrupted magic value at 0x1000 (1234 0234). If all defaults were taken when the file system was created, the magic number should be 0x43218765. If any defaults were overridden, the magic number should be 0x65872143.

4. Use the **od** command to check the secondary superblock for a correct magic number. An example command and its output follows:

```
$ od -x -N 64 /dev/lv02 +0x1f000
001f000 6587 2143 0000 0000 0000 4000 0000 000a
001f010 0001 8000 1000 0000 2f6c 7633 0000 6c76
001f020 3300 0000 000a 0003 0100 0000 2f28 0383
001f030 0000 0001 0000 0200 0000 2000 0000 0000
001f040
```

Note the correct magic value at 0x1f000.

5. Copy the secondary superblock to the primary superblock. An example command and output follows:

```
$ dd count=1 bs=4k skip=31 seek=1 if=/dev/lv02 of=/dev/lv02
dd: 1+0 records in.
dd: 1+0 records out.
```

6. Use the **fsck** command to clean up inconsistent files caused by using the secondary superblock. For example:

```
fsck /dev/lv02 2>&1 | tee /tmp/fsck.errs
```

For more information

- The **fsck** and **od** command descriptions in *AIX 5L Version 5.3 Commands Reference, Volume 4*
- AIX Logical Volume Manager from A to Z: Introduction and Concepts, an IBM Redbook
- AIX Logical Volume Manager from A to Z: Troubleshooting and Commands, an IBM Redbook
- "Boot Problems" in *Problem Solving and Troubleshooting in AIX 5L*, an IBM Redbook

File system types

AIX supports multiple file system types.

These include the following:

Journalled File System (JFS) or Enhanced Journalled File System (JFS2)

Supports the entire set of file system semantics. These file systems use database journaling techniques to maintain structural consistency. This prevents damage to the file system when the system is halted abnormally.

Each JFS or JFS2 resides on a separate logical volume. The operating system mounts the file system 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.

JFS is the basic file system type that supports the entire set of file system commands.

JFS2 is the basic file system type that supports the entire set of file system commands.

A difference between JFS and JFS2 is that JFS2 is designed to support large files and large file systems. These file system types are described more thoroughly in “JFS and JFS2.”

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. NFS is described more thoroughly in Network file system.

CD-ROM File System (CDRFS)

Allows access to the contents of a CD-ROM through the normal file system interfaces (open, read, and close). CDRFS is described more thoroughly in “CD-ROM file system and UDFS” on page 136.

DVD-ROM File System (UDFS)

Allows access to the contents of a DVD through the normal file system interfaces. UDFS is described more thoroughly in “CD-ROM file system and UDFS” on page 136.

JFS and JFS2

The journaled file system (JFS) and the enhanced journaled file system (JFS2) are built into the base operating system. Both file system types link their file and directory data to the structure used by the AIX Logical Volume Manager for storage and retrieval.

A difference is that JFS2 is designed to accommodate a 64-bit kernel and larger files.

The following sections describe these file systems. Unless otherwise noted, the following sections apply equally to JFS and JFS2.

JFS and JFS2 functions

Enhanced Journaled File System (JFS2) is a file system, introduced in AIX 5.1, that provides the capability to store much larger files than the existing Journaled File System (JFS).

Customers can choose to implement either JFS, which is the recommended file system for 32-bit environments, or JFS2, which offers 64-bit functionality.

Note: Unlike the JFS file system, the JFS2 file system will not allow the **link()** API to be used on files of type directory. This limitation may cause some applications that operate correctly on a JFS file system to fail on a JFS2 file system.

The following table provides a summary of JFS and JFS2 functions:

Functions	JFS2	JFS
Fragments and block size	Block sizes (bytes): 512, 1024, 2048, 4096 Maximum file system size in terabytes (TBs): 4, 8, 16, 32	Fragment sizes (bytes): 512, 1024, 2048, 4096 Maximum file system size in gigabytes (GBs): 128, 256, 512, 1024
Maximum file system size	32 TBs	1 TB

Functions	JFS2	JFS
Minimum file system size	16 MBs	Not Applicable
Maximum file size	16 TBs	Approximately 63.876 GBs
Number of i-nodes	Dynamic, limited by disk space	Fixed, set at file system creation
Directory organization	B-tree	Linear
Compression	No	Yes
Quotas	Yes	Yes
Error logging	Yes	Yes

Note:

1. The maximum file size and maximum file system size is limited to (1 TB - (physical partition size)) when used with the 32-bit kernel. For example, if the physical partition size for the volume group is 64 MB, then the maximum file system size is (1 TB - 64 MB) = (1048576 MB - 64 MB) = 1048512 MB. This is due to an underlying limitation on the maximum size of a logical volume when using the 32-bit kernel.
2. JFS2 supports the standard AIX error logging scheme beginning in AIX 5.2. For more information on AIX error logging, please see Error-Logging Overview in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

JFS and JFS2 disk space segmentation

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

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

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

The journaled file system (JFS) divides disk space into allocation units called *fragments*. The enhanced journaled file system (JFS2) segments disk space into *blocks*. The objective is the same: to efficiently store data.

JFS fragments are smaller than the default disk allocation size of 4096 bytes. Fragments minimize wasted disk space by more efficiently storing the data in a file or directory partial logical blocks. The functional behavior of JFS fragment support is based on that provided by Berkeley Software Distribution (BSD) fragment support.

JFS2 supports multiple file system block sizes of 512, 1024, 2048, and 4096. Smaller block sizes minimize wasted disk space by more efficiently storing the data in a file or directory's partial logical blocks. Smaller block sizes also result in additional operational overhead. The block size for a JFS2 is specified during its creation. Different file systems can have different block sizes, but only one block size can be used within a single file system.

JFS fragments:

In JFS, the disk space allocation unit is called a *fragment*, and it can be smaller than the logical block size of 4096 bytes.

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

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

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

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

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

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

JFS2 blocks:

The enhanced journaled file system segments disk space into *blocks*. JFS2 supports multiple file system block sizes of 512, 1024, 2048, and 4096.

Different file systems can have different block sizes, but only one block size can be used within a single file system.

Smaller block sizes minimize wasted disk space by more efficiently storing the data in a file or directory's partial logical blocks. Smaller block sizes can also result in additional operational overhead. Also, device drivers must provide disk block addressability that is the same or smaller than the file system block size.

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

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

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

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

Variable number of i-nodes

Segmenting disk space at sizes smaller than 4096 bytes optimizes disk space utilization, but 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.

JFS and i-nodes:

JFS allows you to specify the number of disk i-nodes created within a file system in case more or fewer than the default number of disk i-nodes is desired.

The number of disk i-nodes at file system creation is specified in a value called as the *number of bytes per i-node* or *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.

For JFS file systems, one i-node is created for every NBPI bytes of allocation group space allocated to the file system. The total number of i-nodes in a file system limits the total number of files and the total size of the file system. An allocation group can be partially allocated, though the full number of i-nodes per allocation group is still allocated. NBPI is inversely proportional to the total number of i-nodes in a file system.

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

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

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

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

You can identify fragment size and NBPI value using the Web-based System Manager, the System Management Interface Tool (SMIT), or the **lsfs** command. For application programs, use the **statfs** subroutine to identify the file system fragment size.

JFS2 and i-nodes:

JFS2 allocates i-nodes as needed.

If there is room in the file system for additional i-nodes, they are automatically allocated. Therefore, the number of i-nodes available is limited by the size of the file system itself.

JFS and JFS2 size limitations

You define the maximum size for a JFS when you create the file system. The decision of what size to define for a JFS is based on several significant issues.

The following section describes the key considerations.

The recommended maximum size for a JFS2 is 16 TBs. The minimum file system size for a JFS2 is 16 MBs. Key considerations for very large JFS2 file systems are described in “JFS2 size limits” on page 129.

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

The allocation state of each fragment (JFS) or block (JFS2) within a file system is recorded in the file system allocation map. More virtual memory and file system disk space might be required to hold allocation maps for file systems with a fragment or block size smaller than 4096 bytes.

Because disk space is allocated in smaller units for a file system with a fragment (JFS) or block (JFS2) size other than 4096 bytes, allocation activity can occur more often when files or directories are repeatedly extended in size. For example, a write operation that extends the size of a zero-length file by 512 bytes results in the allocation of one 512-byte fragment or block to the file, depending on the file system type. If the file size is extended further by another write of 512 bytes, an additional fragment or block must be allocated to the file. Applying this example to a file system with 4096-byte fragments or blocks, disk space allocation occurs only once, as part of the first write operation. No additional allocation activity must be performed as part of the second write operation since the initial 4096-byte allocation is large enough to hold the data added by the second write operation. Allocation activity can be minimized if the files are extended by 4096 bytes at a time.

One size-related issue is the size of the file system log.

For JFS, in most instances, multiple file systems use a common log configured to be 4 MB in size. For example, after initial installation, all file systems within the root volume group use logical volume hd8 as a common JFS log. The default logical volume partition size is 4 MB, and the default log size is one partition, therefore, the root volume group normally contains a 4 MB JFS log. When file systems exceed 2 GB or when the total amount of file system space using a single log exceeds 2 GB, the default log size might not be sufficient. In either case, the log sizes are scaled upward as the file system size increases.

When the size of the log logical volume is changed, the **logform** command must be run to reinitialize the log before the new space can be used. The JFS log is limited to a maximum size of 256 MB.

There is a practical limit to the size of the combined file systems that a single JFS log can support. As a guideline, one trillion bytes of total file system capacity is the recommended limitation for a single JFS log. When this guideline is exceeded or is close to being exceeded, or when out-of-memory errors occur from the **logredo** command (which is called by the **fsck** command), add an additional JFS log and then share the load between the two JFS log files.

For JFS2, in most instances, multiple file systems also use a common log. When file systems exceed 2 GB or when the total amount of file system space using a single log exceeds 2 GB, the default log size might not be sufficient. In either case, you can scale log sizes upward as the file system size increases or you can add an additional JFS2 log and then share the load between the two JFS2 log files.

JFS size limits:

The maximum JFS size is defined when the file system is created. The NBPI, fragment size, and allocation group size are contributing factors to the decision.

The file system size limitation is the minimum of the following:

$$NBPI * 2^{24}$$

or

$$FragmentSize * 2^{28}$$

For example, if you select an NBPI ratio of 512, the file system size is limit to 8 GB ($512 * 2^{24} = 8$ GB). JFS supports NBPI values of 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, and 131072.

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

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

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

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

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}).

JFS2 size limits:

Testing has shown that extremely large JFS2 file systems that contain very large files are more practical to maintain than those that contain a large number of small files. When a large file system contains many small files, the **fsck** command and other file system maintenance tasks take a long time to run.

The following size limitations are recommended:

Maximum JFS2 file system size:	32TB
Maximum JFS2 file size:	16TB
Minimum JFS2 file system size:	16MB

JFS free space fragmentation:

For JFS file systems, using fragments smaller than 4096 bytes can cause greater fragmentation of the free space on the disk.

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

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

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

Sparse files

A file is a sequence of indexed blocks. Blocks are mapped from the i-node to the logical offset of the file they represent.

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

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

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

A read to a section of a file that has unallocated data blocks results in a buffer of zeroes being returned. A write to a section of a file that has unallocated data blocks causes the necessary data blocks to be allocated and the data written.

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

- **cp**
- **mv**
- **tar**
- **cpio**

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

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

- **backup**
- **restore**
- **pax**

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

JFS and large files

You can create large files with the JFS file system type.

All JFS2 file systems support large files.

File systems enabled for large files can be created with the **crfs** and **mkfs** commands. Both commands have an option (**bf=true**) to specify file systems enabled for large files. You can also use SMIT or Web-based System Manager to create these file systems.

In file systems enabled for large files, file data stored before the 4 MB file offset is allocated in 4096-byte blocks. File data stored beyond the 4 MB file offset is allocated with large disk blocks of 128 KB in size. The large disk blocks are actually 32 contiguous 4096-byte blocks.

For example, in a regular file system, the 132-MB file requires 33K 4-KB disk blocks (33 single indirect blocks each filled with 1024 4 KB disk addresses). A 132-MB file in a file system enabled for large files has 1024 4-KB disk blocks and 1024 128-KB disk blocks. The large file geometry requires only two single indirect blocks for the 132 MB file. Both large and regular file types require one double indirect block.

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

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.

JFS data compression

JFS supports fragmented and compressed file systems, which 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.

Data compression is not supported for JFS2.

In a fragmented file system, only the last logical block of files no larger than 32KB are stored in this manner, so that fragment support is only beneficial for file systems containing numerous small files. Data compression, however, allows all logical blocks of any-sized file to be stored as one or more contiguous fragments. On average, data compression saves disk space by about a factor of two.

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

Data compression in the current JFS is compatible with previous versions of this operating system. The 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 "JFS and JFS2 disk space segmentation" on page 125.

JFS data compression implementation:

Data compression is an attribute of a file system which is specified when the file system is created with the **crfs** or **mkfs** command. You can also use Web-based System Manager or SMIT to specify data compression.

Attention: The root file system (*/*) must not be compressed. Compressing the */usr* file system is not recommended because **installp** must be able to accurately calculate its size for updates and new installs. See the "Implicit behavior of JFS data compression" section for more information on size and calculations.

Compression only applies to regular files and long symbolic links in such file systems. Fragment support continues to apply to directories and metadata that are not compressed. Each logical block of a file is compressed by itself before being written to the disk. Compression in this manner facilitates random seeks and updates, while losing only a small amount of freed disk space in comparison to compressing data in larger units.

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

The **lsfs -q** command displays the current value for compression. You can also use Web-based System Manager or SMIT to identify data compression.

Implicit behavior of JFS data compression:

Because a program that writes a file does not expect an out-of-space (ENOSPC) condition to occur after a successful write (or successful store for mapped files), it is necessary to guarantee that space be available when logical blocks are written to the disk.

This is accomplished by allocating 4096 bytes to a logical block when it is first modified so that there is disk space available even if the block does not compress. If a 4096-byte allocation is not available, the system returns an ENOSPC or EDQUOT error condition even though there might be enough disk space to accommodate the compressed logical block. Premature reporting of an out-of-space condition is most likely when operating near disk quota limits or with a nearly full file system.

Compressed file systems might also exhibit the following behavior:

- Because 4096 bytes are initially allocated to a logical block, certain system calls might receive an ENOSPC or EDQUOT error. For example, an old file might be mapped using the **mmap** system call, and a store operation into a previously written location can result in an ENOSPC error.

- With data compression, a full disk block remains allocated to a modified block until it is written to disk. If the block had a previously committed allocation of less than a full block, the amount of disk space tied up by the block is the sum of the two, the previous allocation not being freed until the file (i-node) is committed. This is the case for normal fragments. The number of logical blocks in a file that can have previously committed allocations is, at most, one for normal fragments but can be as many as the number of blocks in a file with compression.
- None of the previously committed resources for a logical block are freed until the **fsync** or **sync** system call is run by the application program.
- The **stat** system call indicates the number of fragments allocated to a file. The number reported is based on 4096 bytes being allocated to modified but unwritten blocks and the compressed size of unmodified blocks. Previously committed resources are not counted by the **stat** system call. The **stat** system call reports the correct number of allocated fragments after an i-node commit operation if none of the modified blocks compressed. Similarly, disk quotas are charged for the current allocation. As the logical blocks of a file are written to the disk, the number of fragments allocated to them decrease if they compress, and thereby change disk quotas and the result from **stat**.

JFS data 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 first occurrence of the string and its length.

At the beginning of the compression process, no strings have been identified, so at least the first byte of data must be represented as a "raw" character requiring 9-bits (0,byte). After a given amount of data is compressed, say N bytes, the compressor searches for the longest string in the N bytes that matches the string starting at the next unprocessed byte. If the longest match has length 0 or 1, the next byte is encoded as a "raw" character. Otherwise, the string is represented as a (pointer,length) pair and the number of bytes processed is incremented by length. Architecturally, IBM LZ supports values of N of 512, 1024, or 2048. IBM LZ specifies the encoding of (pointer,length) pairs and of raw characters. The pointer is a fixed-length field of size $\log_2 N$, while the length is encoded as a variable-length field.

Performance costs of JFS data compression:

Because data compression is an extension of fragment support, the performance associated with fragments also applies to data compression.

Compressed file systems also affect performance in the following ways:

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

JFS online backups and JFS2 snapshots

You can make an point-in-time image of a JFS file system or of a JFS2 file system (AIX 5.2 and later), which you can then use for backup purposes. There are differences, however, in the requirements and behavior of this image for each file system type.

For a JFS file system, you can split off a read-only static copy of a mirrored copy of the file system. Typically, a mirrored copy is updated whenever the original file system is updated, but this point-in-time copy does not change. It remains a stable image of the point in time at which the copy was made. When

this image is used for backing up, any modifications that begin after you begin the procedure to create the image might not be present in the backup copy. Therefore, it is recommended that file system activity be minimal while the split is taking place. Any changes that occur after the split is made will not be present in the backup copy.

For a JFS2 file system, the point-in-time image is called a *snapshot*. The snapshot remains static and it retains the same security permissions as the original file system (called the *snappedFS*) had when the snapshot was made. Also, you can create a JFS2 snapshot without unmounting or quiescing the file system. You can use a JFS2 snapshot to use as an online backup of the file system, to access the files or directories as they existed when the snapshot was taken, or to back up to removable media. Note the following about JFS2 snapshots:

- A snapshot image of the root (*/*) or */usr* file system is overwritten when the system is rebooted. Snapshots of other file systems can be preserved by unmounting the file system before rebooting. Snapshots created in AIX 5.2 with 5200-01 are recoverable. When **fsck** or **logredo** runs on a JFS2 filesystem with a snapshot created on AIX 5.2 with 5200-01, the snapshot will be preserved. A cleanly unmounted filesystem with an AIX 5.2-created snapshot will also be recoverable once it is mounted on an AIX 5.2 with 5200-01 system.
- Running the **defrags** command against a file system with snapshots is not recommended. Every block that is moved during defragmentation must also be copied to the snapshot, which is both time consuming and a waste of space in the snapshot logical volume.
- If a snapshot runs out of space, all snapshots for that snappedFS are deleted. This failure writes an entry to the error log.
- If a write to a snapshot fails, all snapshots for that snappedFS are deleted. This failure writes an entry to the error log.
- A snapshot that is created or accessed on a AIX 5.2 with 5200-01 system cannot be accessed on an AIX 5.2 system. These snapshots must be deleted before the filesystem can be mounted.
- A JFS2 file system that has a snapshot on AIX 5.3 cannot be accessed on any releases prior to AIX 5.2 with 5200-01. If the system is to be moved back, the snapshots must be deleted first to allow file system access.

Compatibility and migration

JFS file systems are fully compatible within AIX 5.1 and AIX 5.2. Previous supported versions of the operating system are compatible with the current JFS, although file systems with a nondefault fragment size, NBPI value, or allocation group size might require special attention if migrated to a previous version.

JFS2 file systems, with the exception of snapshots, are compatible within AIX 5.1 and AIX 5.2, but not with earlier versions of the operating system. JFS2 file systems with snapshots are not supported in AIX 5.1. Always ensure a clean unmount of all JFS2 file systems before reverting to an earlier version of AIX because the **logredo** command does not necessarily run against a file system created for a later release.

Note: JFS2 file systems created or converted to v2 format cannot be accessed on prior releases of AIX.

The following list describes aspects that might cause problems with file systems created under earlier versions of the operating system:

JFS file system images

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 JFS file system images created under AIX 4.3 and later versions of this operating system without requiring any special migration activities.

Note: JFS2 Snapshots: JFS2 snapshots created or accessed in AIX 5L Version 5.2 with the 5200-01 Recommended Maintenance package are not accessible on earlier releases. These snapshots must be deleted before the filesystem can be mounted.

Backup and restore between JFS file systems

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

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

JFS and JFS2 device driver limitations

A device driver must provide disk block addressability that is the same or smaller than the JFS file system fragment size or the JFS2 block 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.

Copying a JFS to another physical volume

You can copy a JFS file system to a different physical volume while retaining file system integrity.

The following scenario describes copying JFS file system to a different physical volume while retaining file system integrity.

Table 5. Things to Consider

For the following scenario to be successful in a concurrent volume group environment, AIX 4.3.2 or later must be installed on all concurrent nodes.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.
--

To copy a JFS to another physical volume while maintaining file system integrity, do the following:

1. Stop activity for the file system that you want to copy. Unless the application that is using the file system is quiescent or the file system is in a state that is known to you, you cannot know what is in the data of the backup.
2. Mirror the logical volume, by typing the following SMIT fast path on the command line:
`smit mklvcopy`
3. Copy the file system, using the following command:
`chfs -a splitcopy=/backup -a copy=2 /testfs`

The **splitcopy** parameter for the **-a** flag causes the command to split off a mirrored copy of the file system and mount it read-only at the new mount point. This action provides a copy of the file system with consistent journaled meta data that can be used for backup purposes.

4. If you want to move the mirrored copy to a different mount point, use the following SMIT fast path:
`smit cplv`

At this point, the file system copy is usable.

For more information

- Logical volume storage
- The **mkiv** command description in *AIX 5L Version 5.3 Commands Reference, Volume 3*
- "Chapter 2. Mirroring" in *AIX Logical Volume Manager from A to Z: Introduction and Concepts*, an IBM Redbook
- *AIX Storage Management*, an IBM Redbook

CD-ROM file system and UDFS

The CD-ROM file system (CDRFS) is a read-only local file system implementation under the logical file system (LFS) layer. The CDRFS is stored on CD-ROM media. The DVD-ROM file system (UDFS) is a read-only file system stored on DVD-ROM media.

Beginning with AIX 5.2, CDs are automatically mounted by default but this feature can be disabled. If the feature has been disabled, use the **cdmount** command to mount the CDRFS file system.

For AIX 5.1 and earlier, mount the CD-ROM and its file system using the mount command and the CD must be write-protected. For example:

```
mount -r -v cdrfs /dev/cd0 /mnt
```

You must specify the following when mounting a CD-ROM (AIX 5.1 and earlier):

Device name

Defines the name of the device containing the media.

Mount point

Specifies the directory where the file system will be mounted.

Automatic mount

Specifies whether the file system will be mounted automatically at system restart.

AIX supports the following CDRFS volume and file structure formats:

Type	Description
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).

Note: The CDRFS must be unmounted from the system before you can remove the CD-ROM media.

Beginning with AIX 5.2, another supported file system type is UDFS, which is a read-only file system stored on DVD-ROM media. The UDFS must be unmounted from the system before you can remove the media. The operating system supports UDFS format versions 1.50, 2.00, and 2.01.

To use the **cdmount** command to automatically mount a read/write UDFS, edit the **cdromd.conf** file. You can also manually mount a read/write UDFS with the **mount** command.

Directories

A *directory* is a unique type of file that contains only the information needed to access files or other directories. As a result, a directory occupies less space than other types of files.

File systems consist of groups of directories and the files within the directories. File systems are commonly represented as an inverted tree. The root directory, denoted by the slash (/) symbol, defines a file system and appears at the top of a file system tree diagram.

Directories branch downward from the root directory in the tree diagram and can contain both files and subdirectories. Branching creates unique paths through the directory structure to every object in the file system.

Collections of files are stored in directories. These collections of files are often related to each other; storing them in a structure of directories keeps them organized.

A *file* is a collection of data that can be read from or written to. A file can be a program you create, text you write, data you acquire, or a device you use. Commands, printers, terminals, correspondence, and application programs are all stored in files. This allows users to access diverse elements of the system in a uniform way and gives great flexibility to the file system.

Directories let you group files and other directories to organize the file system into a modular hierarchy, which gives the file system structure flexibility and depth.

Directories contain directory entries. Each entry contains a file or subdirectory name and an index node reference number (*i-node* number). To increase speed and enhance use of disk space, the data in a file is stored at various locations in the memory of the computer. The i-node number contains the addresses used to locate all the scattered blocks of data associated with a file. The i-node number also records other information about the file, including times of modification and access, access modes, number of links, file owner, and file type.

A special set of commands controls directories. For example, you can link several names for a file to the same i-node number by creating directory entries with the **ln** command.

Because directories often contain information that should not be available to all users of the system, directory access can be protected. By setting a directory's permissions, you can control who has access to the directory, also determining which users (if any) can alter information within the directory. See "File and directory access modes" on page 413 for more information.

Types of directories

Directories can be defined by the operating system, by the system administrator, or by users.

The system-defined directories contain specific kinds of system files, such as commands. At the top of the file system hierarchy is the system-defined **/(root)** directory. The **/(root)** directory usually contains the following standard system-related directories:

/dev	Contains special files for I/O devices.
/etc	Contains files for system initialization and system management.

/home	Contains login directories for the system users.
/tmp	Contains files that are temporary and are automatically deleted after a specified number of days.
/usr	Contains the lpp , include , and other system directories.
/usr/bin	Contains user-executable programs.

Some directories, such as your login or home directory (**\$HOME**), are defined and customized by the system administrator. When you log in to the operating system, the login directory is the current directory.

Directories that you create are called *user-defined* directories. These directories allow you to organize and maintain your files.

Directory organization

Directories contain files, subdirectories, or a combination of both. A *subdirectory* is a directory within a directory. The directory containing the subdirectory is called the *parent directory*.

Each directory has an entry for the parent directory in which it was created, .. (dot dot), and an entry for the directory itself, . (dot). In most directory listings, these files are hidden.

Directory Tree

The file system structure of directories can easily become complex. Attempt to keep the file and directory structure as simple as possible. Create files and directories with easily recognizable names. This makes working with files easier.

Parent Directory

Each directory, except for **/(root)**, has one parent directory and may have child directories.

Home Directory

When you log in, the system puts you in a directory called your *home directory* or login directory. Such a directory is set up by the system administrator for each user. Your home directory is the repository for your personal files. Normally, directories that you create for your own use will be subdirectories of your home directory. To return to your home directory at any time, type the **cd** command and press Enter at the prompt.

Working Directory

You are always working within a directory. Whichever directory you are currently working in is called your *current* or *working* directory. The **pwd** (present working directory) command reports the name of your working directory. Use the **cd** command to change working directories.

Directory naming conventions

The name of each directory must be unique within the directory where it is stored. This ensures that the directory has a unique path name in the file system.

Directories follow the same naming conventions as files, as explained in “File naming conventions” on page 380.

Directory abbreviations

Abbreviations provide a convenient way to specify certain directories.

The following is a list of abbreviations:

Abbreviation	Meaning
.	The current working directory.
..	The directory above the current working directory (the parent of the current directory).
~	Your home directory. (This is not true for the Bourne shell. For more information, see “Bourne shell” on page 483.)

Abbreviation	Meaning
\$HOME	Your home directory. (This is true for all shells.)

Directory path names

Each file and directory can be reached by a unique path, known as the *path name*, through the file system tree structure. The path name specifies the location of a directory or file within the file system.

Note: Path names cannot exceed 1023 characters in length.

The file system uses the following kinds of path names:

absolute path name	Traces the path from the /(root) directory. Absolute path names always begin with the slash (/) symbol.
relative path name	Traces the path from the current directory through its parent or its subdirectories and files.

An absolute path name represents the complete name of a directory or file from the **/(root)** directory downward. Regardless of where you are working in the file system, you can always find a directory or file by specifying its absolute path name. Absolute path names start with a slash (/), the symbol representing the root directory. The path name **/A/D/9** is the absolute path name for **9**. The first slash (/) represents the **/(root)** directory, which is the starting place for the search. The remainder of the path name directs the search to **A**, then to **D**, and finally to **9**.

Two files named **9** can exist because the absolute path names to the files give each file a unique name within the file system. The path names **/A/D/9** and **/C/E/G/9** specify two unique files named **9**.

Unlike full path names, relative path names specify a directory or file based on the current working directory. For relative path names, you can use the notation dot dot (..) to move upward in the file system hierarchy. The dot dot (..) represents the parent directory. Because relative path names specify a path starting in the current directory, they do not begin with a slash (/). Relative path names are used to specify the name of a file in the current directory or the path name of a file or directory above or below the level of the current directory in the file system. If **D** is the current directory, the relative path name for accessing **10** is **F/10**. However, the absolute path name is always **/A/D/F/10**. Also, the relative path name for accessing **3** is **../B/3**.

You can also represent the name of the current directory by using the notation dot (.). The dot (.) notation is commonly used when running programs that read the current directory name.

Creating directories (mkdir command)

Use the **mkdir** command to create one or more directories specified by the *Directory* parameter.

Each new directory contains the standard entries dot (.) and dot dot (..). You can specify the permissions for the new directories with the **-m Mode** flag.

When you create a directory, it is created within the current, or working, directory unless you specify an absolute path name to another location in the file system.

The following are examples of how to use the **mkdir** command:

- To create a new directory called **Test** in the current working directory with default permissions, type the following:

```
mkdir Test
```
- To create a directory called **Test** with **rwxr-xr-x** permissions in a previously created **/home/demo/sub1** directory, type the following:

```
mkdir -m 755 /home/demo/sub1/Test
```

- To create a directory called **Test** with default permissions in the **/home/demo/sub2** directory, type the following:

```
mkdir -p /home/demo/sub2/Test
```

The **-p** flag creates the **/home**, **/home/demo**, and **/home/demo/sub2** directories if they do not already exist.

See the **mkdir** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Moving or renaming directories (mvdircmd)

Use the **mvdircmd** command to move or rename a directory.

The following are examples of how to use the **mvdircmd** command:

- To move a directory, type the following:

```
mvdircmd book manual
```

This moves the **book** directory under the directory named **manual**, if the **manual** directory exists. Otherwise, the **book** directory is renamed to **manual**.

- To move and rename a directory, type the following:

```
mvdircmd book3 proj4/manual
```

This moves the **book3** directory to the directory named **proj4** and renames **proj4** to **manual** (if the **manual** directory did not previously exist).

See the **mvdircmd** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Displaying the current directory (pwdcmd)

Use the **pwdcmd** command to write to standard output the full path name of your current directory (from the **/**(root) directory).

All directories are separated by a slash (/). The **/**(root) directory is represented by the first slash (/), and the last directory named is your current directory.

For example, to display your current directory, type the following:

```
pwd
```

The full path name of your current directory displays similar to the following:

```
/home/thomas
```

Changing to another directory (cdcmd)

Use the **cdcmd** command to move from your present directory to another directory. You must have execute (search) permission in the specified directory.

If you do not specify a *Directory* parameter, the **cdcmd** command moves you to your login directory (**\$HOME** in the **ksh** and **bsh** environments, or **\$home** in the **cs**h environment). If the specified directory name is a full path name, it becomes the current directory. A full path name begins with a slash (/) indicating the **/**(root) directory, a dot (.) indicating current directory, or a dot dot (..) indicating parent directory. If the directory name is not a full path name, the **cdcmd** command searches for it relative to one of the paths specified by the **\$CDPATH** shell variable (or **\$cdpath** **cs**h variable). This variable has the same syntax as, and similar semantics to, the **\$PATH** shell variable (or **\$path** **cs**h variable).

The following are examples of how to use the **cd** command:

- To change to your home directory, type the following:

```
cd
```

- To change to the **/usr/include** directory, type the following:

```
cd /usr/include
```

- To go down one level of the directory tree to the **sys** directory, type the following:

```
cd sys
```

If the current directory is **/usr/include** and it contains a subdirectory named **sys**, then **/usr/include/sys** becomes the current directory.

- To go up one level of the directory tree, type the following:

```
cd ..
```

The special file name, dot dot (**..**), refers to the directory immediately above the current directory, its parent directory.

See the **cd** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Copying directories (cp command)

Use the **cp** command to create a copy of the contents of the file or directory specified by the *SourceFile* or *SourceDirectory* parameters into the file or directory specified by the *TargetFile* or *TargetDirectory* parameters.

If the file specified as the *TargetFile* exists, the copy writes over the original contents of the file. If you are copying more than one *SourceFile*, the target must be a directory.

To place a copy of the *SourceFile* into a directory, specify a path to an existing directory for the *TargetDirectory* parameter. Files maintain their respective names when copied to a directory unless you specify a new file name at the end of the path. The **cp** command also copies entire directories into other directories if you specify the **-r** or **-R** flags.

The following are examples of how to use the **cp** command:

- To copy all the files in the **/home/accounts/customers/orders** directory to the **/home/accounts/customers/shipments** directory, type the following:

```
cp /home/accounts/customers/orders/* /home/accounts/customers/shipments
```

This copies the files, but not the directories, from the **orders** directory into the **shipments** directory.

- To copy a directory, including all its files and subdirectories, to another directory, type the following:

```
cp -R /home/accounts/customers /home/accounts/vendors
```

This copies the **customers** directory, including all its files, subdirectories, and the files in those subdirectories, into the **vendors** directory.

See the **cp** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Displaying contents of a directory (ls command)

Use the **ls** command to display the contents of a directory.

The **ls** command writes to standard output the contents of each specified *Directory* or the name of each specified *File*, along with any other information you ask for with the flags. If you do not specify a *File* or *Directory*, the **ls** command displays the contents of the current directory.

By default, the **ls** command displays all information in alphabetic order by file name. If the command is executed by a user with root authority, it uses the **-A** flag by default, listing all entries except dot (.) and dot dot (..). To show all entries for files, including those that begin with a dot (.), use the **ls -a** command.

You can format the output in the following ways:

- List one entry per line, using the **-l** flag.
- List entries in multiple columns, by specifying either the **-C** or **-x** flag. The **-C** flag is the default format when output is to a tty.
- List entries in a comma-separated series by specifying the **-m** flag.

To determine the number of character positions in the output line, the **ls** command uses the **\$COLUMNS** environment variable. If this variable is not set, the command reads the **terminfo** file. If the **ls** command cannot determine the number of character positions by either of these methods, it uses a default value of 80.

The information displayed with the **-e** and **-l** flags is interpreted as follows:

The first character of each entry may be one of the following:

- d** Entry is a directory.
- b** Entry is a block special file.
- c** Entry is a character special file.
- l** Entry is a symbolic link.
- p** Entry is a first-in, first-out (FIFO) pipe special file.
- s** Entry is a local socket.
- Entry is an ordinary file.

The next nine characters are divided into three sets of three characters each. The first three characters show the file or directory owner's permission. The next three characters show the permission of the other users in the group. The last three characters show the permission of anyone else with access to the file. The three characters in each set show read, write, and execute permission of the file. Execute permission of a directory lets you search a directory for a specified file.

Permissions are indicated as follows:

- r** Read permission granted
- t** Only the directory owner or the file owner can delete or rename a file within that directory, even if others have write permission to the directory.
- w** Write (edit) permission granted
- x** Execute (search) permission granted
- Corresponding permission not granted.

The information displayed with the **-e** flag is the same as with the **-l** flag, except for the addition of an 11th character, interpreted as follows:

- +** Indicates a file has extended security information. For example, the file might have extended **ACL**, **TCB**, or **TP** attributes in the mode.
- Indicates a file does not have extended security information.

When the size of the files in a directory are listed, the **ls** command displays a total count of blocks, including indirect blocks.

See the following examples:

- To list all files in the current directory, type the following:

```
ls -a
```

This lists all files, including

- dot (.)
- dot dot (..)
- Other files whose names might or might not begin with a dot (.)

- To display detailed information, type the following:

```
ls -l chap1 .profile
```

This displays a long listing with detailed information about **chap1** and **.profile**.

- To display detailed information about a directory, type the following:

```
ls -d -l . manual manual/chap1
```

This displays a long listing for the directories **.** and **manual**, and for the file **manual/chap1**. Without the **-d** flag, this would list the files in the **.** and **manual** directories instead of the detailed information about the directories themselves.

See the **ls** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Deleting or removing directories (rmdir command)

Use the **rmdir** command to remove the directory, specified by the *Directory* parameter, from the system.

The directory must be empty (it can contain only **.** and **..**) before you can remove it, and you must have write permission in its parent directory. Use the **ls -a***Directory* command to check whether the directory is empty.

The following are examples of how to use the **rmdir** command:

- To empty and remove a directory, type the following:

```
rm mydir/* mydir/.  
rmdir mydir
```

This removes the contents of **mydir**, then removes the empty directory. The **rm** command displays an error message about trying to remove the directories dot (.) and dot dot (..), and then the **rmdir** command removes them and the directory itself.

Note: **rm mydir/* mydir/.*** first removes files with names that do not begin with a dot, and then removes those with names that do begin with a dot. The **ls** command does not list file names that begin with a dot unless you use the **-a** flag.

- To remove the **/tmp/jones/demo/mydir** directory and all the directories beneath it, type the following:

```
cd /tmp  
rmdir -p jones/demo/mydir
```

This removes the **jones/demo/mydir** directory from the **/tmp** directory. If a directory is not empty or you do not have write permission to it when it is to be removed, the command terminates with appropriate error messages.

See the **rmdir** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Comparing the contents of directories (dircmp command)

Use the **dircmp** command to compare two directories specified by the *Directory1* and *Directory2* parameters and write information about their contents to standard output.

First, the **dircmp** command compares the file names in each directory. If the same file name is contained in both, the **dircmp** command compares the contents of both files.

In the output, the **dircmp** command lists the files unique to each directory. It then lists the files with identical names in both directories, but with different contents. If no flag is specified, it also lists files that have identical contents as well as identical names in both directories.

The following are examples of how to use the **dircmp** command:

- To summarize the differences between the files in the **proj.ver1** and **proj.ver2** directories, type the following:

```
dircmp proj.ver1 proj.ver2
```

This displays a summary of the differences between the **proj.ver1** and **proj.ver2** directories. The summary lists separately the files found only in one directory or the other, and those found in both. If a file is found in both directories, the **dircmp** command notes whether the two copies are identical.

- To show the details of the differences between the files in the **proj.ver1** and **proj.ver2** directories, type the following:

```
dircmp -d -s proj.ver1 proj.ver2
```

The **-s** flag suppresses information about identical files. The **-d** flag displays a **diff** listing for each of the differing files found in both directories.

See the **dircmp** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Command summary for file systems and directories

The following are commands for file systems and directories, commands for directory-handling procedures, and a list of directory abbreviations.

Table 6. Command summary for file systems

df	Reports information about space on file systems.
-----------	--

Table 7. Directory abbreviations

.	The current working directory.
..	The directory above the current working directory (the parent directory).
~	Your home directory. (This is not true for the Bourne shell. For more information, see “Bourne shell” on page 483.)
\$HOME	Your home directory. (This is true for all shells.)

Table 8. Command summary for directory-handling procedures

cd	Changes the current directory.
cp	Copies files or directories.
dircmp	Compares two directories and the contents of their common files.
ls	Displays the contents of a directory.
mkdir	Creates one or more new directories.
mvdir	Moves (renames) a directory.
pwd	Displays the path name of the working directory.
rmdir	Removes a directory.

Chapter 6. Backup methods

Once your system is in use, your next consideration should be to back up the file systems, directories, and files. If you back up your file systems, you can restore files or file systems in the event of a hard disk crash. There are different methods for backing up information.

Backing up file systems, directories, and files represents 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.

Attention: 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.

If you use 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.

Several methods exist for backing up information. One of the most frequently used methods is called *backup by name, file name archive, or regular backup*. This is a copy of a file system, directory, or file that is kept for file transfer or in case the original data is unintentionally changed or destroyed. 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 i-node, file system archive, or archive backup*. This method of backup is done when the **i** flag is *not* specified. This is used for future reference, historical purposes, or for recovery if the original data is damaged or lost. It is used to make a backup copy of an entire file system and is the method commonly used by system administrators to back up large groups of files, such as all of the user accounts in **/home**. A file system backup allows incremental backups to be performed easily. An incremental backup backs up all files that have been modified since a specified previous backup.

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

Several commands create backups and archives. Because of this, data that has been backed up needs to be labeled as to which command was used to initiate the backup, and how the backup was made (by name or by file system).

backup	Backs up files by name or by file system. For more information, see backup .
mksysb	Creates an installable image of the rootvg. For more information, see mksysb .
cpio	Copies files into and out of archive storage. For more information, see cpio .
dd	Converts and copies a file. Commonly used to convert and copy data to and from systems running other operating systems, for example, mainframes. dd does not group multiple files into one archive; it is used to manipulate and move data. For more information, see dd .
tar	Creates or manipulates tar format archives. For more information, see tar .
rdump	Backs up files by file system onto a remote machine's device. For more information, see rdump .
pax	(POSIX-conformant archive utility) Reads and writes tar and cpio archives. For more information, see pax .

Backup policies

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 one hundred users. Likewise, a policy developed for a system on which many files are changed daily would be inefficient for a system on which data changes infrequently.

Whatever the appropriate backup strategy for your site, it is very important that one exist and that backups be done frequently and regularly. It is difficult to recover from data loss if a good backup strategy has not been implemented.

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

- **Make sure you can recover from major losses.**

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

- **Check your backups periodically.**

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

- **Keep old backups.**

Develop a regular cycle for reusing your backup media; however, do not reuse all of your backup media. Sometimes it is months before you or some other user of your system notices that an important file is damaged or missing. Save old backups for such possibilities. For example, you could have the following three cycles of backup tapes or diskettes:

- Once per week, recycle all daily diskettes except the Friday diskette.
- Once per month, recycle all Friday diskettes except for the one from the last Friday of the month. This makes the last four Friday backups always available.
- Once per quarter, recycle all monthly diskettes except for the last one. Keep the last monthly diskette from each quarter indefinitely, preferably in a different building.

- **Check file systems before backing up.**

A backup made from a damaged file system might be useless. Before making your backups, it is good policy to check the integrity of the file system with the **fsck** command. For more information, see **fsck**.

- **Ensure files are not in use during a backup.**

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

- **Back up your system before major changes are made to the system.**

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

- **Other factors.**

When planning and implementing a backup strategy, consider the following factors:

- How often does the data change? The operating system data does not change very often, so you do not need to back it up frequently. User data, on the other hand, usually changes frequently, so you should back it up frequently.
- How many users are on the system? The number of users affects the amount of storage media and frequency required for backups.
- How difficult would it be to re-create the data? It is important to consider that some data cannot be re-created if there is no backup available.

Having a backup strategy in place to preserve your data is very important. Evaluating the needs of your site will help you to determine the backup policy that is best for you. Perform user information backups frequently and regularly. Recovering from data loss is very difficult if a good backup strategy has not been implemented.

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

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

Backup media

Several different types of backup media are available. The different types of backup media available to your specific system configuration depend upon both your software and hardware.

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 tapes (8-mm tape and 9-track tape), diskettes (5.25-inch diskette and 3.5-inch diskette), remote archives, and alternate local hard disks. 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.

Attention: Running the **backup** command results in the loss of all material previously stored on the selected backup medium.

Diskettes

Diskettes are the standard backup medium. Unless you specify a different device using the **backup -f** command, the **backup** command automatically writes its output to the **/dev/rfd0** device, which is the diskette drive. To back up data to the default tape device, type **/dev/rmt0** and press Enter.

Be careful when you handle diskettes. Because each piece of information occupies such a small area on the diskette, small scratches, dust, food, or tobacco particles can make the information unusable. Be sure to remember the following:

- Do not touch the recording surfaces.
- Keep diskettes away from magnets and magnetic field sources, such as telephones, dictation equipment, and electronic calculators.
- Keep diskettes away from extreme heat and cold. The recommended temperature range is 10 degrees Celsius to 60 degrees Celsius (50 degrees Fahrenheit to 140 degrees Fahrenheit).
- Proper care helps prevent loss of information.
- Make backup copies of your diskettes regularly.

Attention: Diskette drives and diskettes must be the correct type to store data successfully. If you use the wrong diskette in your 3.5-inch diskette drive, the data on the diskette could be destroyed.

The diskette drive uses the following 3.5-inch diskettes:

- 1 MB capacity (stores approximately 720 KB of data)
- 2 MB capacity (stores approximately 1.44 MB of data)

Tapes

Because of their high capacity and durability, tapes are often chosen for storing large files or many files, such as archive copies of file systems. They are also used for transferring many files from one system to another. Tapes are not widely used for storing frequently accessed files because other media provide much faster access times.

Tape files are created using commands such as **backup**, **cpio**, and **tar**, which open a tape drive, write to it, and close it.

Backup strategy

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

- Complete system backup
- Incremental backup

To understand these two types of backups and which one is right for a site or system, it is important to have an understanding of file system structure and data placement. After you have decided on a strategy for data placement, you can develop a backup strategy for that data. See “Implementing scheduled backups” on page 153 for an example of a backup strategy that includes weekly complete system backups and daily incremental backups.

System data versus user data

Data is defined as programs or text and for this discussion is broken down into two classes:

- System data, which makes up the operating system and its extensions. This data is always to be kept in the system file systems, namely **/** (root), **/usr**, **/tmp**, **/var**, and so on.
- User data is typically local data that individuals need to complete their specific tasks. This data is to be kept in the **/home** file system or in file systems that are created specifically for user data.

User programs and text are not to be placed in file systems designed to contain system data. For example, a system manager might create a new file system and mount it over **/local**. An exception is **/tmp**, which is used for temporary storage of system and user data.

Backups

In general, backups of user and system data are kept in case data is accidentally removed or in case of a disk failure. It is easier to manage backups when user data is kept separate from system data.

The following are reasons for keeping system data separate from user data:

- User data tends to change much more often than operating system data. Backup images are much smaller if the system data is not backed up into the same image as the user data. The number of users affects the storage media and frequency required for backup.
- It is quicker and easier to restore user data when it is kept separate. Restoring the operating system along with the user data requires extra time and effort. The reason is that the method used to recover the operating system data involves booting the system from removable media (tape or CD-ROM) and installing the system backup.

To back up the system data, unmount all user file systems, including **/home** with the **umount** command. If these file systems are in use, you cannot unmount them. Schedule the backups at low usage times so they can be unmounted; if the user data file systems remain mounted, they are backed up along with the operating system data. Use the **mount** command to ensure that only the operating system file systems are mounted.

The only mounted file systems are **/**, **/usr**, **/var**, and **/tmp**, and the output of the **mount** command should be similar to the following:

node	mounted	mounted over	vfs	date	options
	/dev/hd4	/	jfs	Jun 11 10:36	rw,log=/dev/hd8
	/dev/hd2	/usr	jfs	Jun 11 10:36	rw,log=/dev/hd8
	/dev/hd9var	/var	jfs	Jun 11 10:36	rw,log=/dev/hd8
	/dev/hd	/tmp	jfs	Jun 11 10:36	rw,log=/dev/hd8

After you are certain that all user file systems are unmounted, see “Backup of the system image and user-defined volume groups” on page 150 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.

System replication (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

Two procedures can be used to back up files and file systems: the SMIT fast paths **smit backfile** or **smit backfilesys**, and the **backup** command.

- If you are backing up by i-node file systems that may be in use, unmount them first to prevent inconsistencies.

Attention: If you attempt to back up a mounted file system, a warning message is displayed. The **backup** command continues, but inconsistencies in the file system may occur. This warning does not apply to the root (**/**) file system.

- To prevent errors, make sure the backup device has been cleaned recently.

To back up user files and file systems, you can use the Web-based System Manager, the SMIT fast paths **smit backfile** or **smit backfilesys**, or the commands listed in Chapter 6, “Backup methods,” on page 145, such as 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. SMIT might hang if multiple tapes or disks are needed during the backup. For more information, see the **backup** command description in *AIX 5L Version 5.3 Commands Reference, Volume 1*.

Use the **backup** command when you want to back up large and multiple file systems. You can specify 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.
- Entire file systems backed up by i-node using the **-Level** and **FileSystem** parameters. The file system is defragmented when it is restored from backup.

Attention: Backing up by i-node does not work correctly for files that have a user ID (UID) or a group ID (GID) greater than 65535. These files are backed up with UID or GID truncated and will, therefore, have the wrong UID or GID attributes when restored. For these cases, you must back up by name.

For additional information about backing up user files or file systems, see Chapter 6, “Backup methods,” on page 145.

Backing Up User Files or File Systems Tasks

<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Back Up User Files	smit backfile	<ol style="list-style-type: none"> 1. Log in to your user account. 2. Backup: <code>find . -print backup -ivf /dev/rmt0</code>
Back Up User File Systems	smit backfilesys	<ol style="list-style-type: none"> 1. Unmount files systems that you plan to back up. For example: <code>umount all</code> or <code>umount /home /filesys1</code> 2. Verify the file systems. For example: <code>fck /home /filesys1</code> 3. Back up by i-node. For example: <code>backup -5 -uf/dev/rmt0 /home/libr</code> 4. Restore the files using the following command: restore -t

Note: If this command generates an error message, you must repeat the entire backup.

Backup of the system image and user-defined volume groups

The rootvg is stored on a hard disk, or group of disks, and contains start up files, the 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. You can backup an image of the system and volume groups using Web-based System Manager, SMIT, or command procedures.

A backup image serves two purposes. One is to restore a corrupted system using the system backup image. The other is to transfer installed and configured software from one system to others.

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.

Note:

1. Startup tapes cannot be made on or used to start a PowerPC-based 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 products and service updates for information on how to install software packages and options.

Backing up the system image and user-defined volume groups

You can make backups of the system image and the user-defined volume groups.

Before backing up the rootvg volume group:

- All hardware must already be installed, including external devices, such as tape and CD-ROM drives.
- This backup procedure requires the **sysbr** fileset, which is in the BOS System Management Tools and Applications software package. Type the following command to determine whether the **sysbr** fileset is installed on your system:

```
ls1pp -l bos.sysmgt.sysbr
```

If your system has the **sysbr** fileset installed, continue the backup procedures.

If the **ls1pp** command does not list the **sysbr** fileset, install it before continuing with the backup procedure. See Installing optional software products and service updates in the *Installation and migration* for instructions.

```
installp -agqXd device bos.sysmgt.sysbr
```

where *device* is the location of the software; for example, */dev/rmt0* for a tape drive.

Before backing up a user-defined volume group:

- Before being saved, a volume group must be varied on and the file systems must be mounted.
 - Attention:** Executing the **savevg** command results in the loss of all material previously stored on the selected output medium.
- Make sure the backup device has been cleaned recently to prevent errors.

The following procedures describe how to make an installable image of your system.

Backing Up Your System Tasks

Task	SMIT Fast Path	Command or File
Backing up the rootvg volume group	<ol style="list-style-type: none"> 1. Log in as root. 2. Mount file systems for backup. smit mountfs 3. Unmount any local directories that are mounted over another local directory. smit umountfs 4. Make at least 8.8MB of free disk space available in the /tmp directory.² 5. Back up: smit mksysb 6. Write-protect the backup media. 7. Record any backed-up root and user passwords. 	<ol style="list-style-type: none"> 1. Log in as root. 2. Mount file systems for backup.¹ See mount command. 3. Unmount any local directories that are mounted over another local directory. See umount command. 4. Make at least 8.8MB of free disk space available in the /tmp directory.² 5. Back up. See mksysb command. 6. Write-protect the backup media. 7. Record any backed-up root and user passwords.
Verify a Backup Tape ³	smit lsmksysb	
Backing up a user-defined volume group ⁴	smit savevg	<ol style="list-style-type: none"> 1. Modify the file system size before backing up, if necessary.⁵ mkvgdata VGName then edit /tmp/vgdata/VGName/VGName.data 2. Save the volume group. See the savevg command.

Note:

1. The **mksysb** command does not back up file systems mounted across an NFS network.

2. The **mksysb** command requires this working space for the duration of the backup. Use the **df** command, which reports in units of 512-byte blocks, to determine the free space in the **/tmp** directory. Use the **chfs** command to change the size of the file system, if necessary.
3. This procedure lists the contents of a **mksysb** backup tape. The contents list verifies most of the information on the tape but does not verify that the tape can be booted for installations. The only way to verify that the boot image on a **mksysb** tape functions correctly is by booting from the tape.
4. If you want to exclude files in a user-defined volume group from the backup image, create a file named **/etc/exclude.volume_group_name**, where *volume_group_name* is the name of the volume group that you want to back up. Then edit **/etc/exclude.volume_group_name** and enter the patterns of file names that you do not want included in your backup image. The patterns in this file are input to the pattern matching conventions of the **grep** command to determine which files are excluded from the backup.
5. If you choose to modify the **VGName.data** file to alter the size of a file system, you must not specify the **-i** flag or the **-m** flag with the **savevg** command, because the **VGName.data** file is overwritten.

For more information about installing (or *restoring*) a backup image, see *Installing system backups in the Installation and migration*.

Pre-backup configuration

Configure the source system before creating a backup image of it. If, however, you plan to use a backup image for installing other, differently configured target systems, create the image *before* configuring the source system.

The *source* system is the system from which you created the backup copy. The *target* system is the system on which you are installing the backup copy.

The installation program automatically installs only the device support required for the hardware configuration of the installed machine. Therefore, if you are using a system backup to install other machines, you might need to install additional devices on the source system before making the backup image and using it to install one or more target systems.

Use Web-based System Manager or the SMIT fast path, `smit devinst`, to install additional device support on the source system.

- If there is sufficient disk space on the source and target systems, install all device support.
- If there is limited disk space on the source and target systems, selectively install device support.

For information on installing optional software, see *Installing optional software and service updates*.

A backup transfers the following configurations from the source system to the target system:

- Paging space information
- Logical volume information
- rootvg information
- Placement of logical partitions (if you have selected the map option).

See *Customizing your installation* 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.

File system mounts and unmounts

Before performing a backup, you must mount all file systems you want to back up and unmount all file systems you do not want to back up.

The Backup Methods procedure backs up only mounted file systems in the rootvg. You must, therefore, mount all file systems you want to back up before starting. Similarly, you must unmount file systems you do *not* want backed up.

This backup procedure backs up files twice if a local directory is mounted over another local directory in the same file system. For example, if you mount **/tmp** over **/usr/tmp**, the files in the **/tmp** directory are backed up twice. This duplication might exceed the number of files a file system can hold, which can cause a future installation of the backup image to fail.

Security considerations for backups

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.

Backup image restoration

When installing the backup image, the system checks whether the target system has enough disk space to create all the logical volumes stored on the backup. If there is enough space, the entire backup is recovered. Otherwise, the installation halts and the system prompts you to choose more destination hard disks.

File systems created on the target system are the same size as they were on the source system, unless the **SHRINK** variable was set to yes in the **image.data** file before the backup image was made. An exception is the **/tmp** directory, which can be increased to allocate enough space for the **bosboot** command. For information about setting variables, see the **image.data** file.

When the system finishes installing the backup image, the installation program reconfigures the ODM on the target system. If the target system does not have exactly the same hardware configuration as the source system, the program might modify device attributes in the following target system files:

- All files in **/etc/objrepos** beginning with Cu
- All files in the **/dev** directory.

For more information about installing (or restoring) a backup image, see Installing system backups.

Implementing scheduled backups

This procedure describes how to develop and use a script to perform a weekly full backup and daily incremental backups of user files.

- The amount of data scheduled for backup cannot exceed one tape when using this script.
- Make sure the tape is loaded in the backup device before the **cron** command runs the script.
- Make sure the device is connected and available, especially when using scripts that run at night. Use the **lsdev -C l pg** command to check availability.
- Make sure the backup device has been cleaned recently to prevent errors.
- If you are backing up file systems that might be in use, unmount them first to prevent file system corruption.
- Check the file system before making the backup. Use the procedure “File system verification” on page 107 or run the **fsck** command.

The script included in this procedure is intended only as a model and needs to be carefully tailored to the needs of the specific site.

Backing up file systems using the cron command

This procedure describes how to write a **crontab** script that you can pass to the **crontab** command for execution.

The script backs up two user file systems, **/home/plan** and **/home/run**, on Monday through Saturday nights. Both file systems are backed up on one tape, and each morning a new tape is inserted for the next night. The Monday night backups are full archives (level 0). The backups on Tuesday through Saturday are incremental backups.

1. The first step in making the **crontab** script is to issue the **crontab -e** command. This opens an empty file where you can make the entries that are submitted to the **crontab** script for execution each night (the default editor is **vi**). Type:

```
crontab -e
```

2. The following example shows the six **crontab** fields. Field 1 is for the minute, field 2 is for the hour on a 24-hour clock, field 3 is for the day of the month, and field 4 is for the month of the year. Fields 3 and 4 contain an * (asterisk) to show that the script runs every month on the day specified in the **day/wk** field. Field 5 is for the day of the week, and can also be specified with a range of days, for example, 1-6. Field 6 is for the shell command being run.

```
min hr day/mo mo/yr day/wk      shell command
0  2  *    *    1      backup -0 -uf /dev/rmt0.1 /home/plan
```

The command line shown assumes that personnel at the site are available to respond to prompts when appropriate. The **-0** (zero) flag for the **backup** command stands for level zero, or full backup. The **-u** flag updates the backup record in the **/etc/dumpdates** file and the **f** flag specifies the device name, a raw magnetic tape device 0.1 as in the example above. See *rmt Special File* in the *AIX 5L Version 5.3 Files Reference* for information on the meaning of extension .1 and other extensions (1-7).

3. Type a line similar to that in step 2 for each file system backed up on a specific day. The following example shows a full script that performs six days of backups on two file systems:

```
0 2 * * 1 backup -0 -uf/dev/rmt0.1 /home/plan
0 3 * * 1 backup -0 -uf/dev/rmt0.1 /home/run
0 2 * * 2 backup -1 -uf/dev/rmt0.1 /home/plan
0 3 * * 2 backup -1 -uf/dev/rmt0.1 /home/run
0 2 * * 3 backup -2 -uf/dev/rmt0.1 /home/plan
0 3 * * 3 backup -2 -uf/dev/rmt0.1 /home/run
0 2 * * 4 backup -3 -uf/dev/rmt0.1 /home/plan
0 3 * * 4 backup -3 -uf/dev/rmt0.1 /home/run
0 2 * * 5 backup -4 -uf/dev/rmt0.1 /home/plan
0 3 * * 5 backup -4 -uf/dev/rmt0.1 /home/run
0 2 * * 6 backup -5 -uf/dev/rmt0.1 /home/plan
0 3 * * 6 backup -5 -uf/dev/rmt0.1 /home/run
```

4. Save the file you created and exit the editor. The operating system passes the **crontab** file to the **crontab** script.

Backup of files on a DMAPI-managed JFS2 file system

Beginning with AIX 5L Version 5.3 with the 5300-03 Recommended Maintenance package, there are options in the **tar** and **backbyinode** commands that allow you to back up the extended attributes (EAs).

With the **backbyinode** command on a DMAPI file system, only the data resident in the file system at the time the command is issued is backed up. The **backbyinode** command examines the current state of metadata to do its work. This can be advantageous with DMAPI, because it backs up the state of the managed file system. However, any offline data will not be backed up.

To back up all of the data in a DMAPI file system, use a command that reads entire files, such as the **tar** command. This can cause a DMAPI-enabled application to restore data for every file accessed by the **tar** command, moving data back and forth between secondary and tertiary storage, so there can be performance implications.

Formatting diskettes (format or fdformat command)

You can format diskettes in the diskette drive specified by the *Device* parameter (the **/dev/rfd0** device by default) with the **format** and **fdformat** commands.

Attention: Formatting a diskette destroys any existing data on that diskette.

The **format** command determines the device type, which is one of the following:

- 5.25-inch low-density diskette (360 KB) containing 40x2 tracks, each with 9 sectors
- 5.25-inch high-capacity diskette (1.2 MB) containing 80x2 tracks, each with 15 sectors
- 3.5-inch low-density diskette (720 KB) containing 80x2 tracks, each with 9 sectors
- 3.5-inch high-capacity diskette (2.88 MB) containing 80x2 tracks, each with 36 sectors

The sector size is 512 bytes for all diskette types.

Use the **format** command to format a diskette for high density unless the *Device* parameter specifies a different density.

Use the **fdformat** command to format a diskette for low density unless the **-h** flag is specified. The *Device* parameter specifies the device containing the diskette to be formatted (such as the **/dev/rfd0** device for drive 0).

Before formatting a diskette, the **format** and **fdformat** commands prompt for verification. This allows you to end the operation cleanly if necessary.

See the following examples:

- To format a diskette in the **/dev/rfd0** device, type the following:

```
format -d /dev/rfd0
```
- To format a diskette without checking for bad tracks, type the following:

```
format -f
```
- To format a 360 KB diskette in a 5.25-inch, 1.2 MB diskette drive in the **/dev/rfd1** device, type the following:

```
format -l -d /dev/rfd1
```
- To force high-density formatting of a diskette when using the **fdformat** command, type the following:

```
fdformat -h
```

See the **format** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Checking the integrity of a file system (fsck command)

Use the **fsck** command to check and interactively repair inconsistent file systems.

It is important to run this command on every file system as part of system initialization. You must be able to read the device file on which the file system resides (for example, the **/dev/hd0** device). Normally, the file system is consistent, and the **fsck** command merely reports on the number of files, used blocks, and free blocks in the file system. If the file system is inconsistent, the **fsck** command displays information about the inconsistencies found and prompts you for permission to repair them. The **fsck** command is

conservative in its repair efforts and tries to avoid actions that might result in the loss of valid data. In certain cases, however, the **fsck** command recommends the destruction of a damaged file.

Attention: Always run the **fsck** command on file systems after a system malfunction. Corrective actions can result in some loss of data. The default action for each consistency correction is to wait for the operator to type yes or no. If you do not have *write* permission for an affected file, the **fsck** command will default to a no response.

See the following examples:

- To check all the default file systems, type the following:

```
fsck
```

This form of the **fsck** command asks you for permission before making any changes to a file system.

- To fix minor problems automatically with the default file systems, type the following:

```
fsck -p
```

- To check the **/dev/hd1** file system, type the following:

```
fsck /dev/hd1
```

This checks the unmounted file system located on the **/dev/hd1** device.

Note: The **fsck** command does not make corrections to a mounted file system.

See the **fsck** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Copying to or from diskettes (fscopy command)

Use the **fscopy** command to copy a diskette (opened as **/dev/rfd0**) to a file named **floppy** created in the current directory.

The message Change floppy, hit return when done displays as needed. The **fscopy** command then copies the **floppy** file to the diskette.

See the following examples:

- To copy **/dev/rfd1** to the **floppy** file in the current directory, type the following:

```
fscopy -f /dev/rfd1 -r
```

- To copy the first 100 tracks of the diskette, type the following:

```
fscopy -f /dev/rfd1 -t 100
```

See the **fscopy** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Copying files to tape or disk (cpio -o command)

Use the **cpio -o** command to read file path names from standard input and copy these files to standard output, along with path names and status information.

Path names cannot exceed 128 characters. Avoid giving the **cpio** command path names made up of many uniquely linked files because it might not have enough memory to keep track of the path names and would lose linking information.

See the following examples:

- To copy files in the current directory whose names end with **.c** onto diskette, type the following:

```
ls *.c | cpio -ov >/dev/rfd0
```

The **-v** flag displays the names of each file.

- To copy the current directory and all subdirectories onto diskette, type the following:

```
find . -print | cpio -ov >/dev/rfd0
```

This saves the directory tree that starts with the current directory (.) and includes all of its subdirectories and files.

- To use a shorter command string, type the following:

```
find . -cpio /dev/rfd0 -print
```

The **-print** entry displays the name of each file as it is copied.

See the **cpio** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Copying files from tape or disk (cpio -i command)

Use the **cpio -i** command to read from standard input an archive file created by the **cpio -o** command and copy from it the files with names that match the *Pattern* parameter.

These files are copied into the current directory tree. You can list more than one *Pattern* parameter by using the file name notation described in the **ksh** command. The default for the *Pattern* parameter is an asterisk (*), which selects all files in the current directory. In an expression such as [a-z], the hyphen (-) means *through*, according to the current collating sequence.

Note: The patterns "*.c" and "*.o" must be enclosed in quotation marks to prevent the shell from treating the asterisk (*) as a pattern-matching character. This is a special case in which the **cpio** command itself decodes the pattern-matching characters.

See the following examples:

- To list the files that have been saved onto a diskette with the **cpio** command, type the following:

```
cpio -itv </dev/rfd0
```

This displays the table of contents of the data previously saved onto the **/dev/rfd0** file in the **cpio** command format. The listing is similar to the long directory listing produced by the **ls -l** command.

- To list only the file path names, use only the **-it** flags.
- To copy the files previously saved with the **cpio** command from a diskette, type the following:

```
cpio -idmv </dev/rfd0
```

This copies the files previously saved onto the **/dev/rfd0** file by the **cpio** command back into the file system (specify the **-i** flag). The **-d** flag allows the **cpio** command to create the appropriate directories if a directory tree is saved. The **-m** flag maintains the last modification time in effect when the files are saved. The **-v** flag causes the **cpio** command to display the name of each file as it is copied.

- To copy selected files from diskette, type the following:

```
cpio -i "*.c" "*.o" </dev/rfd0
```

This copies the files that end with **.c** or **.o** from diskette.

See the **cpio** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Copying to or from tapes (tcopy command)

Use the **tcopy** command to copy magnetic tapes.

For example, to copy from one streaming tape to a 9-track tape, type the following:

```
tcopy /dev/rmt0 /dev/rmt8
```

See the **tcopy** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Checking the integrity of a tape (tapechk command)

Use the **tapechk** command to perform rudimentary consistency checking on an attached streaming tape device.

Some hardware malfunctions of a streaming tape drive can be detected by simply reading a tape. The **tapechk** command provides a way to perform tape reads at the file level.

For example, to check the first three files on a streaming tape device, type the following:

```
tapechk 3
```

See the **tapechk** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

File backup

Use either the **backup** command or the **smit** command to create copies of your files on backup media, such as a magnetic tape or diskette.

Attention: If you attempt to back up a mounted file system, a message displays. The **backup** command continues, but inconsistencies in the file system can occur. This situation does not apply to the root (/) file system.

The copies you created with the **backup** command or the **smit** command are in one of the following backup formats:

- Specific files backed up by name, using the **-i** flag.
- Entire file system backed up by i-node number, using the **-Level** and **FileSystem** parameters.

Note:

- The possibility of data corruption always exists when a file is modified during system backup. Therefore, make sure that system activity is at a minimum during the system backup procedure.
- If a backup is made to 8-mm tape with the device block size set to 0 (zero), it is not possible to directly restore data from the tape. If you have done backups with the 0 setting, you can restore data from them by using special procedures described under the **restore** command.

Attention: Be sure the flags you specify match the backup media.

File backup using the backup command

Use the **backup** command to create copies of your files on backup media.

For example, to back up selected files in your **\$HOME** directory by name, type the following:

```
find $HOME -print | backup -i -v
```

The **-i** flag prompts the system to read from standard input the names of files to be backed up. The **find** command generates a list of files in the user's directory. This list is piped to the **backup** command as standard input. The **-v** flag displays a progress report as each file is copied. The files are backed up on the default backup device for the local system.

See the following examples:

- To back up the root file system, type the following:

```
backup -0 -u /
```

The 0 level and the / tell the system to back up the / (root) file system. The file system is backed up to the **/dev/rfd0** file. The **-u** flag tells the system to update the current backup level record in the **/etc/dumpdates** file.

- To back up all files in the / (root) file system that were modified since the last 0 level backup, type the following:

```
backup -1 -u /
```

See the **backup** command in the *AIX 5L Version 5.3 Commands Reference* for the complete syntax.

Backing up files using the smit command

Use the **smit** command to run the **backup** command, which creates copies of your files on backup media.

1. At the prompt, type the following:

```
smit backup
```

2. Type the path name of the directory on which the file system is normally mounted in the **DIRECTORY full pathname** field:

```
/home/bill
```

3. In the **BACKUP** device or **FILE** fields, enter the output device name, as in the following example for a raw magnetic tape device:

```
/dev/rmt0
```

4. Use the Tab key to toggle the optional **REPORT each phase of the backup** field if you want error messages printed to the screen.
5. In a system management environment, use the default for the **MAX number of blocks to write on backup medium** field because this field does not apply to tape backups.
6. Press Enter to back up the named directory or file system.
7. Run the **restore -t** command. If this command generates an error message, you must repeat the entire backup.

Compressing files (compress and pack commands)

Use the **compress** command and the **pack** command to compress files for storage.

Use the **uncompress** command and the **unpack** command to expand the restored files.

The process of compressing and expanding files takes time; however, after the files are packed, the data uses less space on the backup medium.

To compress a file system, use one of the following methods:

- Use the **-p** flag with the **backup** command.
- Use the **compress** or **pack** commands.

Advantages for compressing files include:

- Saving money and time by compressing files before sending them over a network.

- Saving storage and archiving system resources:
 - Compress file systems before making backups to preserve tape space.
 - Compress log files created by shell scripts that run at night; it is easy to have the script compress the file before it exits.
 - Compress files that are not currently being accessed. For example, the files belonging to a user who is away for extended leave can be compressed and placed into a **tar** archive on disk or to a tape and later be restored.

Note:

- The **compress** command might run out of working space in the file system while compressing. The command creates the compressed files before it deletes any of the uncompressed files, so it needs a space about 50% larger than the total size of the files.
- A file might fail to compress because it is already compressed. If the **compress** command cannot reduce file sizes, the command fails.

For the procedure to compress a file and write the percentage compression to standard error, see “Compressing files using the compress command.”

See the **compress** command for details about the return values but, in general, the problems encountered when compressing files can be summarized as follows:

- The command might run out of working space in the file system while compressing. Because the **compress** command creates the compressed files before it deletes any of the uncompressed files, it needs extra space—from 50% to 100% of the size of any given file.
- A file might fail to compress because it is already compressed. If the **compress** command cannot reduce the file size, it fails.

For information about file compression using the pack command, see “Compressing files using the pack command” on page 161.

Compressing files using the compress command

Use the **compress** command to reduce the size of files using adaptive Lempel-Zev coding.

Each original file specified by the *File* parameter is replaced by a compressed file with a **.Z** appended to its name. The compressed file retains the same ownership, modes, and access and modification times of the original file. If no files are specified, the standard input is compressed to the standard output. If compression does not reduce the size of a file, a message is written to standard error and the original file is not replaced.

Use the **uncompress** command to restore compressed files to their original form.

The amount of compression depends on the size of the input, the number of bits per code specified by the *Bits* variable, and the distribution of common substrings. Typically, source code or English text is reduced by 50 to 60 percent. The compression of the **compress** command is generally more compact and takes less time to compute than the compression achieved by the **pack** command, which uses adaptive Huffman coding.

For example, to compress the **foo** file and write the percentage compression to standard error, type the following:

```
compress -v foo
```

See the **compress** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Compressing files using the pack command

Use the **pack** command to store the file or files specified by the *File* parameter in a compressed form using Huffman coding.

The input file is replaced by a packed file with a name derived from the original file name (*File.z*), with the same access modes, access and modification dates, and owner as the original file. The input file name can contain no more than 253 bytes to allow space for the added **.z** suffix. If the **pack** command is successful, the original file is removed.

Use the **unpack** command to restore packed files to their original form.

If the **pack** command cannot create a smaller file, it stops processing and reports that it is unable to save space. (A failure to save space generally happens with small files or files with uniform character distribution.) The amount of space saved depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each **.z** file, you do not save space with files smaller than three blocks. Typically, text files are reduced 25 to 40 percent.

The exit value of the **pack** command is the number of files that it could not pack. Packing is not done under any of the following conditions:

- The file is already packed.
- The input file name has more than 253 bytes.
- The file has links.
- The file is a directory.
- The file cannot be opened.
- No storage blocks are saved by packing.
- A file named *File.z* already exists.
- The **.z** file cannot be created.
- An I/O error occurred during processing.

For example, to compress the files **chap1** and **chap2**, type the following:

```
pack chap1 chap2
```

This compresses **chap1** and **chap2** and replaces them with files named **chap1.z** and **chap2.z**. The **pack** command displays the percent decrease in size for each file.

See the **pack** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Expanding compressed files (uncompress and unpack commands)

Use the **uncompress** and **unpack** commands to expand compressed files.

Expanding files using the uncompress command

Use the **uncompress** command to restore original files that were compressed by the **compress** command. Each compressed file specified by the *File* variable is removed and replaced by an expanded copy. The expanded file has the same name as the compressed version but without the **.Z** extension. The expanded file retains the same ownership, modes, and access and modification times as the original file. If no files are specified, standard input is expanded to standard output.

Although similar to the **uncompress** command, the **zcat** command always writes the expanded output to standard output.

For example, to uncompress the **foo** file, type the following:

```
uncompress foo
```

See the **uncompress** command in *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Expanding files using the **unpack** command

Use the **unpack** command to expand files created by the **pack** command. For each file specified, the **unpack** command searches for a file called **File.z**. If this file is a packed file, the **unpack** command replaces it with its expanded version. The **unpack** command renames the new file by removing the **.z** suffix from *File*. The new file has the same access modes, access and modification dates, and owner as the original packed file.

The **unpack** command operates only on files ending in **.z**. As a result, when you specify a file name that does not end in **.z**, the **unpack** command adds the suffix and searches the directory for a file name with that suffix.

The exit value is the number of files that the **unpack** command was unable to unpack. A file cannot be unpacked if any of the following situations exists:

- The file name (exclusive of **.z**) has more than 253 bytes.
- The file cannot be opened.
- The file is not a packed file.
- A file with the unpacked file name already exists.
- The unpacked file cannot be created.

Note: The **unpack** command writes a warning to standard error if the file it is unpacking has links. The new unpacked file has a different i-node (index node) number than the packed file from which it was created. However, any other files linked to the original i-node number of the packed file still exist and are still packed.

For example, to unpack the packed files **chap1.z** and **chap2.z**, type the following:

```
unpack chap1.z chap2
```

This expands the packed files **chap1.z** and **chap2.z**, and replaces them with files named **chap1** and **chap2**.

Note: You can provide the **unpack** command with file names with or without the **.z** suffix.

See the **unpack** command in *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Restoring backed-up files

After the data has been correctly backed up, there are several different methods of restoring the data based upon the type of backup command you used.

You need to know how your backup or archive was created to restore it correctly. Each backup procedure gives information about restoring data. For example, if you use the **backup** command, you can specify a backup either by file system or by name. That backup must be restored the way it was done, by file system or by name. For information about the **backup** command, see **backup**.

Several commands restore backed up data, such as:

restore	Copies files created by the backup command. For more information about using this command, see the section below.
rrestore	Copies file systems backed up on a remote machine to the local machine. For more information, see rrestore .
cpio	Copies files into and out of archive storage. For more information, see cpio .
tar	Creates or manipulates tar archives. For more information, see tar .

pax (POSIX-conformant archive utility) Reads and writes **tar** and **cpio** archives. For more information, see **pax**.

The following sections discuss the **restore** and **smit** commands.

Note:

- Files must be restored using the same method by which they were backed up. For example, if a file system was backed up by name, it must be restored by name.
- When more than one diskette is required, the **restore** command reads the diskette that is mounted, prompts you for a new one, and waits for your response. After inserting the new diskette, press the Enter key to continue restoring files.

Restoring files using the **restore** command

Use the **restore** command to read files written by the **backup** command and restore them on your local system.

See the following examples:

- To list the names of files previously backed up, type the following:

```
restore -T
```

Information is read from the **/dev/rfd0** default backup device. If individual files are backed up, only the file names are displayed. If an entire file system is backed up, the i-node number is also shown.

- To restore files to the main file system, type the following:

```
restore -x -v
```

The **-x** flag extracts all the files from the backup media and restores them to their proper places in the file system. The **-v** flag displays a progress report as each file is restored. If a file system backup is being restored, the files are named with their i-node numbers. Otherwise, only the names are displayed.

- To copy the **/home/mike/manual/chap1** file, type the following:

```
restore -xv /home/mike/manual/chap1
```

This command extracts the **/home/mike/manual/chap1** file from the backup medium and restores it. The **/home/mike/manual/chap1** file must be a name that the **restore -T** command can display.

- To copy all the files in a directory named **manual**, type the following:

```
restore -xdv manual
```

This command restores the **manual** directory and the files in it. If the directory does not exist, a directory named **manual** is created in the current directory to hold the files being restored.

See the **restore** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Restoring files using the **smit** command

Use the **smit** command to run the **restore** command, which reads files written by the **backup** command and restores them on your local system.

1. At the prompt, type the following:

```
smit restore
```

2. Make your entry in the **Target DIRECTORY** field. This is the directory where you want the restored files to reside.

3. Proceed to the **BACKUP device** or **FILE** field and enter the output device name, as in the following example for a raw magnetic tape device:

```
/dev/rmt0
```

If the device is not available, a message similar to the following is displayed:

```
Cannot open /dev/rmtX, no such file or directory.
```

This message indicates that the system cannot reach the device driver because there is no file for **rmtX** in the **/dev** directory. Only items in the **available** state are in the **/dev** directory.

4. For the **NUMBER of blocks to read in a single input** field, the default is recommended.
5. Press Enter to restore the specified file system or directory.

Creating a remote archive

Use this procedure to archive files to a remote tape device.

Running AIX systems cannot mount a remote tape device as if it were local to the system; however, data can be sent to a remote machine tape device using the **rsh** command. The following procedure writes to a single tape only. Multiple-tape archives require specialized application software.

In the following procedure, assume the following:

blocksize

Represents the target tape device blocksize.

remotehost

Is the name of the target system (the system that has the tape drive).

sourcehost

Is the name of the source system (the system being archived).

/dev/rmt0

Is the name of the remote tape device

pathname

Represents the full pathname of a required directory or file.

When using the following instructions, assume that both the local and remote user is root.

1. Ensure you have access to the remote machine. The source machine must have access to the system with the tape drive. (The target system can be accessed using any of the defined users on that system, but the user name must have root authority to do many of the following steps.)
2. Using your favorite editor, create a file in the **/** (root) directory of the target system called **.rhosts** that allows the source system access to the target system. You need to add the authorized host name and user ID to this file. To determine the name of the source machine for the **.rhosts** file, you can use the following command:

```
host SourceIPAddress
```

For the purposes of this example, assume you add the following line to the **.rhosts** file:

```
sourcehost.mynet.com root
```

3. Save the file and then change its permissions using the following command:

```
chmod 600 .rhosts
```

4. Use the **rsh** command to test your access from the source machine. For example:

```
rsh remotehost
```

If everything is set up correctly, you should be granted shell access to the remote machine. You should not see a login prompt asking for a user name. Type **exit** to log out of this test shell.

5. Decide on the appropriate tape device blocksize. The following are the recommended values:

9-track or 0.25-in. media blocksize:	512
8-mm or 4-mm media blocksize:	1024

If you are unsure and want to check the current block size of the tape device, use the **tctl** command. For example:

```
tctl -f /dev/rmt0 status
```

If you want to change the tape blocksize, use the **chdev** command. For example:

```
chdev -l rmt0 -a block_size=1024
```

6. Create your archive using one of the following methods:

Backup by Name

To remotely create a backup archive by name, use the following command:

```
find pathname -print | backup -ivqf- | rsh remotehost \  
"dd of=/dev/rmt0 bs=blocksize conv=sync"
```

Backup by inode

To remotely create a backup archive by inode, first unmount your file system then use the **backup** command. For example:

```
umount /myfs  
backup -0 -uf- /myfs | rsh remotehost \  
"dd of=/dev/rmt0 bs=blocksize conv=sync"
```

Create and Copy an Archive to Remote Tape

To create and copy an archive to the remote tape device, use the following command:

```
find pathname -print | cpio -ovcB | rsh remotehost \  
"dd ibs=5120 obs=blocksize of=/dev/rmt0"
```

Create a tar Archive

To remotely create a **tar** archive, use the following command:

```
tar -cvdf- pathname | rsh remotehost \  
"dd of=/dev/rmt0 bs=blocksize conv=sync"
```

Create a Remote Dump

To remotely create a remote dump of the **/myfs** file system, use the following command:

```
rdump -u -0 -f remotehost:/dev/rmt0 /myfs
```

The **-u** flag tells the system to update the current backup level records in the **/etc/dumpdates** file. The **-0** is the setting of the *Level* flag. Backup level 0 specifies that all the files in the **/myfs** directory are to be backed up. For more information, see the **rdump** command description in *AIX 5L Version 5.3 Commands Reference, Volume 4*.

7. Restore your remote archive using one of the following methods:

Restore a Backup by Name

To restore a remote backup archive by name, use the following command:

```
rsh remotehost "dd if=/dev/rmt0 bs=blocksize" | restore \  
-xvqdf- pathname
```

Restore a Backup by inode

To restore a remote backup archive by inode, use the following command:

```
rsh remotehost "dd if=/dev/rmt0 bs=blocksize" | restore \  
-xvqf- pathname
```

Restore a Remote cpio Archive

To restore a remote archive created with the **cpio** command, use the following command:

```
rsh remotehost "dd if=/dev/rmt0 ibs=blocksize obs=5120" | \  
cpio -icvdumB
```

Restore a tar Archive

To restore a remote **tar** archive, use the following command:

```
rsh remotehost "dd if=/dev/rmt0 bs=blocksize" | tar -xvpf- pathname
```

Restore a Remote Dump

To restore a remote dump of the **/myfs** file system, use the following command:

```
cd /myfs  
rrestore -rvf remotehost:/dev/rmt0
```

Restoring user files from a backup image

If you need to restore a backup image destroyed by accident, your most difficult problem is determining which of the backup tapes contains this file. The **restore -T** command can be used to list the contents of an archive. It is a good idea to restore the file in the **/tmp** directory so that you do not accidentally overwrite the user's other files.

Make sure the device is connected and available. To check availability, type:

```
lsdev -C | pg
```

If the backup strategy included incremental backups, then it is helpful to find out from the user when the file was most recently modified. This helps to determine which incremental backup contains the file. If this information cannot be obtained or is found to be incorrect, then start searching the incremental backups in reverse order (7, 6, 5, ...). For incremental file system backups, the **-i** flag (interactive mode) of the **restore** command is very useful in both locating and restoring the lost file. (Interactive mode is also useful for restoring an individual user's account from a backup of the **/home** file system.)

The procedures in the following table describe how to implement a level 0 (full) restoration of a directory or file system.

Restoring from Backup Image Tasks		
Task	SMIT Fast Path	Command or File
Restore Individual User Files	smit restfile	See restore command.
Restoring a User File System	smit restfilesys	1. mkfs /dev/hd1 2. mount /dev/hd1 /filesys 3. cd /filesys 4. restore -r
Restoring a User Volume Group	smit restvg	See restvg -q command.

Restoring access to an unlinked or deleted system library

When the existing **libc.a** library is not available, most operating system commands are not recognized.

The most likely causes for this type of problem are the following:

- The link in **/usr/lib** no longer exists.
- The file in **/usr/ccs/lib** has been deleted.

The following procedure describes how to restore access to the **libc.a** library. This procedure requires system downtime. If possible, schedule your downtime when it least impacts your workload to protect yourself from a possible loss of data or functionality.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.
--

For more information

- The **mount**, **umount** and **reboot** commands description in the *AIX 5L Version 5.3 Commands Reference*.
- "Boot Problems" in *Problem Solving and Troubleshooting in AIX 5L*, an IBM Redbook

Restoring a deleted symbolic link

Use the following procedure to restore a symbolic link from the **/usr/lib/libc.a** library to the **/usr/ccs/lib/libc.a** path.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. With root authority, set the **LIBPATH** environment variable to point to the **/usr/ccs/lib** directory by typing the following commands:

```
# LIBPATH=/usr/ccs/lib:/usr/lib
# export LIBPATH
```

At this point, you should be able to execute system commands.

2. To restore the links from the **/usr/lib/libc.a** library and the **/lib** directory to the **/usr/lib** directory, type the following commands:

```
ln -s /usr/ccs/lib/libc.a /usr/lib/libc.a
ln -s /usr/lib /lib
```

At this point, commands should run as before. If you still do not have access to a shell, skip the rest of this procedure and continue with the next section, "Restoring a deleted system library file."

3. Type the following command to unset the **LIBPATH** environment variable.

```
unset LIBPATH
```

Restoring a deleted system library file

This procedure to restore a deleted system library file requires system downtime. The system is booted and then the library is restored from a recent **mksysb** tape.

1. Before your reboot, ensure the **PROMPT** field in the **bosinst.data** file is set to yes.
2. Insert a recent **mksysb** tape into the tape drive. The **mksysb** must contain the same OS and maintenance package or technology level as the installed system. If you restore a **libc.a** library from a **mksysb** that conflicts with the level on the installed system, you will not be able to issue commands.
3. Reboot the machine.
4. When the screen of icons appears, or when you hear a double beep, press the F1 key repeatedly until the System Management Services menu is displayed.
5. Select **Multiboot**.
6. Select **Install From**.
7. Select the tape device that holds the **mksysb** and then select **Install**. It can take several minutes before the next prompt appears.
8. Define your current system as the system console by pressing the F1 key and press Enter.
9. Select the number of your preferred language and press Enter.
10. Select **Start Maintenance Mode for System Recovery** by typing 3 and press Enter.
11. Select **Access a Root Volume Group**. A message displays explaining that you will not be able to return to the Installation menus without rebooting if you change the root volume group at this point.
12. Type 0 and press Enter.
13. Type the number of the appropriate volume group from the list and press Enter.

14. Select **Access this Volume Group** by typing 2 and press Enter.
15. Mount the / (root) and /usr file systems by typing the following commands:

```
mount /dev/hd4 /mnt
mount /dev/hd2 /mnt/usr
cd /mnt
```

16. To restore the symbolic link for the **libc.a** library, if needed, type the following command:

```
ln -s /usr/ccs/lib/libc.a /mnt/usr/lib/libc.a
```

After the command runs, do one of the following:

- If the command is successful, skip to step 20.
- If a message displays that the link already exists, continue with step 17.

17. Set the block size of the tape drive by issuing the following commands, where *X* is the number of the appropriate tape drive.

```
tctl -f /dev/rmtX rewind
tctl -f /dev/rmtX.1 fsf 1
restbyname -xvqf /dev/rmtX.1 ./tapeblksz
cat tapeblksz
```

If the value from the **cat tapeblksz** command is *not equal* to 512, type the following commands, replacing *Y* with the value from the **cat tapeblksz** command:

```
ln -sf /mnt/usr/lib/methods /etc/methods
/etc/methods/chgdevn -l rmtX -a block_size=Y
```

You should receive a message that *rmtX* has been changed.

18. Ensure the tape is at the correct location for restoring the library by typing the following commands (where *X* is the number of the appropriate tape drive):

```
tctl -f /dev/rmtX rewind
tctl -f /dev/rmtX.1 fsf 3
```

19. Restore the missing library using one of the following commands (where *X* is the number of the appropriate tape drive):

- To restore the **libc.a** library only, type the following command:

```
restbyname -xvqf /dev/rmtX.1 ./usr/ccs/lib/libc.a
```
- To restore the **/usr/ccs/lib** directory, type the following command:

```
restbyname -xvqf /dev/rmtX.1 ./usr/ccs/lib
```
- To restore the **/usr/ccs/bin** directory, type the following command:

```
restbyname -xvqf /dev/rmtX.1 ./usr/ccs/bin
```

20. Flush the data to disk by typing the following commands:

```
cd /mnt/usr/sbin
./sync;./sync;./sync
```

21. Unmount the **/usr** and / (root) file systems by typing the following commands:

```
cd /
umount /dev/hd2
umount /dev/hd4
```

If either **umount** command fails, cycle power on this machine and begin this procedure again.

22. Reboot the system by typing the following command:

```
reboot
```

After the system is rebooted, operating system commands should be available.

Archiving files (tar command)

The archive backup method is used for a copy of one or more files, or an entire database that is saved for future reference, historical purposes, or for recovery if the original data is damaged or lost.

Usually, an archive is used when that specific data is removed from the system.

Use the **tar** command to write files to or retrieve files from an archive storage. The **tar** command looks for archives on the default device (usually tape), unless you specify another device.

When writing to an archive, the **tar** command uses a temporary file (the **/tmp/tar*** file) and maintains in memory a table of files with several links. You receive an error message if the **tar** command cannot create the temporary file or if there is not enough memory available to hold the link tables.

See the following examples:

- To write the **file1** and **file2** files to a new archive on the default tape drive, type the following:

```
tar -c file1 file2
```
- To extract all files in the **/tmp** directory from the archive file on the **/dev/rmt2** tape device and use the time of extraction as the modification time, type the following:

```
tar -xm -f/dev/rmt2 /tmp
```
- To display the names of the files in the **out.tar** disk archive file from the current directory, type the following:

```
tar -vtf out.tar
```

See the **tar** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for more information and the complete syntax.

Command summary for backup files and storage media

Commands are available for backing up files and storing data.

backup	Backs up files and file systems
compress	Compresses and expands data
cpio	Copies files into and out of archive storage and directories
fdformat	Formats diskettes
flcopy	Copies to and from diskettes
format	Formats diskettes
fsck	Checks file system consistency and interactively repairs the file system
pack	Compresses files
restore	Copies previously backed-up file systems or files, which were created by the backup command from a local device
tapechk	Checks consistency of the streaming tape device
tar	Manipulates archives
tcopy	Copies a magnetic tape
uncompress	Compresses and expands data
unpack	Expands files

Recreating a corrupted boot image

The following procedure describes how to identify a corrupted boot image and re-create it.

If your machine is currently running and you know the boot image has been corrupted or deleted, recreate the boot image by running the **bosboot** command with root authority.

Attention: Never reboot the system when you suspect the boot image is corrupted.

The following procedure assumes your system is not rebooting correctly because of a corrupted boot image. If possible, protect your system from a possible loss of data or functionality by scheduling your downtime when it least impacts your workload.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Insert the product media into the appropriate drive.
2. Power on the machine following the instructions provided with your system.
3. From the **System Management Services** menu, select **Multiboot**.
4. From the next screen, select **Install From**.
5. Select the device that holds the product media and then select **Install**.
6. Select the AIX version icon.
7. Follow the online instructions until you can select which mode you use for installation. At that point, select **Start Maintenance Mode for System Recovery**.
8. Select **Access a Root Volume Group**.
9. Follow the online instructions until you can select **Access this Volume Group and start a shell**.
10. Use the **bosboot** command to re-create the boot image. For example:

```
bosboot -a -d /dev/hdisk0
```

If the command fails and you receive the following message:

```
0301-165 bosboot: WARNING! bosboot failed - do not attempt to boot device.
```

Try to resolve the problem using one of the following options, and then run the **bosboot** command again until you have successfully created a boot image:

- Delete the default boot logical volume (hd5) and then create a new hd5.

Or

- Run diagnostics on the hard disk. Repair or replace, as necessary.

If the **bosboot** command continues to fail, contact your customer support representative.

Attention: If the **bosboot** command fails while creating a boot image, do not reboot your machine.

11. When the **bosboot** command is successful, use the **reboot** command to reboot your system.

For more information

- System startup
- The **bosboot** command description in *AIX 5L Version 5.3 Commands Reference, Volume 1*
- "Boot Problems" in *Problem Solving and Troubleshooting in AIX 5L*, an IBM Redbooks publication

Making an online backup of a JFS

Making an online backup of a mounted journaled file system (JFS) or enhanced journaled file system (JFS2) creates a static image of the logical volume that contains the file system.

To make an online backup of a mounted JFS, the logical volume that the file system resides on and the logical volume that its log resides on must be mirrored.

Note: Because the file writes are asynchronous, the split-off copy might not contain all data that was written immediately before the split. Any modifications that begin after the split begins might not be present in the backup copy. Therefore, it is recommended that file system activity be minimal while the split is taking place.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

To split off a mirrored copy of the **/home/xyz** file system to a new mount point named **/jfsstaticcopy**, type the following:

```
chfs -a splitcopy=/jfsstaticcopy /home/xyz
```

You can control which mirrored copy is used as the backup by using the **copy** attribute. The second mirrored copy is the default if a copy is not specified by the user. For example:

```
chfs -a splitcopy=/jfsstaticcopy -a copy=1 /home/xyz
```

At this point, a read-only copy of the file system is available in **/jfsstaticcopy**. Any changes made to the original file system after the copy is split off are not reflected in the backup copy.

To reintegrate the JFS split image as a mirrored copy at the **/testcopy** mount point, use the following command:

```
rmfs /testcopy
```

The **rmfs** command removes the file system copy from its split-off state and allows it to be reintegrated as a mirrored copy.

Making and backing up a snapshot of a JFS2

Beginning with AIX 5.2, you can make a snapshot of a mounted JFS2 that establishes a consistent block-level image of the file system at a point in time.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

The snapshot image remains stable even as the file system that was used to create the snapshot, called the *snappedFS*, continues to change. The snapshot retains the same security permissions as the *snappedFS* had when the snapshot was made.

In the following scenario, you create a snapshot and back up the snapshot to removable media without unmounting or quiescing the file system, all with one command: **backsnap**. You can also use the snapshot for other purposes, such as accessing the files or directories as they existed when the snapshot was taken. You can do the various snapshot procedures using Web-based System Manager, SMIT, or the **backsnap** and **snapshot** commands.

To create a snapshot of the **/home/abc/test** file system and back it up (by name) to the tape device **/dev/rmt0**, use the following command:

```
backsnap -m /tmp/snapshot -s size=16M -i f/dev/rmt0 /home/abc/test
```

This command creates a logical volume of 16 megabytes for the snapshot of the JFS2 file system (**/home/abc/test**). The snapshot is mounted on **/tmp/snapshot** and then a backup by name of the snapshot is made to the tape device. After the backup completes, the snapshot remains mounted. Use the **-R** flag with the **backsnap** command if you want the snapshot removed when the backup completes.

For more information

- "File Systems"
- The **backsnap**, **chfs**, **rmfs**, and **snapshot** commands
- AIX Logical Volume Manager from A to Z: Introduction and Concepts, an IBM Redbook
- AIX Logical Volume Manager from A to Z: Troubleshooting and Commands, an IBM Redbook

Chapter 7. 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.

Profiles

The shell uses two types of profile files when you log in to the operating system.

The shell evaluates the commands contained in the files and then runs the commands to set up your system environment. The files have similar functions except that the **/etc/profile** file controls profile variables for all users on a system whereas the **.profile** file allows you to customize your own environment.

The following profile and system environment information is provided:

- **/etc/profile** file
- **.profile** file
- System environment variable setup
- Changing the Message of the Day
- “Time data manipulation services” on page 174.

/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. Because 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.

Time data manipulation services

The time functions access and reformat the current system date and time.

You do not need to specify any special flag to the compiler to use the time functions. Include the header file for these functions in the program. To include a header file, use the following statement:

```
#include <time.h>
```

The time services are the following:

adjtime	Corrects the time to allow synchronization of the system clock.
ctime, localtime, gmtime, mktime, difftime, asctime, tzset	Converts date and time to string representation.
getinterval, incinterval, absinterval, resinc, resabs, alarm, ualarm, getitimer, setitimer	Manipulates the expiration time of interval timers.
gettimer, settimer, restimer, stime, time	Gets or sets the current value for the specified systemwide timer.
gettimerid	Allocates a per-process interval timer.
gettimeofday, settimeofday, ftime	Gets and sets date and time.
nsleep, usleep, sleep	Suspends a current process from running.
reltimerid	Releases a previously allocated interval timer.

Dynamic Processor Deallocation

AIX can detect and automatically stop using a faulty processor.

Starting with machine type 7044 model 270, the hardware of all systems with two or more processors is able to detect correctable errors, which are gathered by the firmware. These errors are not fatal and, as long as they remain rare occurrences, can be safely ignored. However, when a pattern of failures seems to be developing on a specific processor, this pattern might indicate that this component is likely to exhibit a fatal failure in the near future. This prediction is made by the firmware based on the failure rates and threshold analysis.

On these systems, AIX implements continuous hardware surveillance and regularly polls the firmware for hardware errors. When the number of processor errors hits a threshold and the firmware recognizes that there is a distinct probability that this system component will fail, the firmware returns an error report. In all cases, the error is logged in the system error log. In addition, on multiprocessor systems, depending on the type of failure, 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.

Processor deallocation impacts to applications

Processor deallocation is transparent for the vast majority of applications, including drivers and kernel extensions. However, you can use the published interfaces to determine whether an application or kernel extension is running on a multiprocessor machine, find out how many processors there are, and bind threads to specific processors.

The `bindprocessor` interface for binding processes or threads to processors uses `bind CPU` numbers. The `bind 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) `bind CPU` to be deallocated. For instance, on an 8-way SMP, the `bind 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.

Note: In the rest of this description, the term *CPU* is used for the logical entity and the term *processor* for the physical entity.

Potentially, applications or kernel extensions that are binding processes or threads could 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. Dynamic Processor Deallocation provides programming interfaces so that such applications and kernel extensions can be notified that a processor deallocation is about to happen. When these applications and kernel extensions receive notification, they are responsible for moving their bound threads and associated resources (such as timer request blocks) away from the last `bind CPU` ID and for adapting themselves to the new CPU configuration.

After notification, if some threads remain bound to the last `bind CPU` ID, the deallocation is aborted, the aborted deallocation is logged in the error log, and AIX continues using the ailing processor. When the processor ultimately fails, it causes a total system failure. Therefore, it is important that applications or kernel extensions receive notification of an impending processor deallocation and act on this notice.

Even in the rare cases that the deallocation cannot go through, Dynamic Processor Deallocation still gives advanced warning to system administrators. By recording the error in the error log, it gives them a chance to schedule a maintenance operation on the system to replace the ailing component before a global system failure occurs.

Processor deallocation process

AIX can stop a failing processor by deallocating it.

The typical flow of events for processor deallocation is as follows:

1. The firmware detects that a recoverable error threshold has been reached by one of the processors.
2. The firmware error report is logged in the system error log, and, when AIX is executing on a machine that supports processor deallocation, AIX starts the deallocation process.
3. AIX notifies non-kernel processes and threads bound to the last `bind CPU`.
4. AIX waits up to ten minutes for all the bound threads to move away from the last `bind CPU`. If threads remain bound, AIX aborts the deallocation.
5. If all processes or threads are unbound from the ailing processor, the previously registered High Availability Event Handlers (HAEHs) are invoked. An HAEH might return an error that aborts the deallocation.
6. Unless aborted, the deallocation process ultimately stops the failing processor.

If there is a failure at any point of the deallocation, the failure and its cause are logged. 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 an application did not unbind its bound threads, the system administrator can stop the application, restart the deallocation, and then restart the application.

Enabling Dynamic Processor Deallocation

If your machine supports Dynamic Processor Deallocation, you can use SMIT or system commands to turn the feature **on** or **off**.

Beginning with AIX 5.2, Dynamic Processor Deallocation is enabled by default during installation, provided the machine has the correct hardware and firmware to support it. In previous versions of AIX, the feature is disabled by default, and if you try to enable it, a message alerts you when your machine cannot support this feature.

For additional information, see “Dynamic Processor Deallocation” on page 174.

SMIT fastpath procedure

1. With root authority, type `smit system` at the system prompt, then press Enter.
2. In the **Systems Environment** window, select **Change / Show Characteristics of Operating System**.
3. Use the SMIT dialogs to complete the task.

To obtain additional information for completing the task, you can select the F1 Help key in the SMIT dialogs.

Commands procedure

With root authority, you can use the following commands to work with the Dynamic Processor Deallocation:

- Use the **chdev** command to change the characteristics of the device specified. For information about using this command, see **chdev** in the *AIX 5L Version 5.3 Commands Reference, Volume 1*.
- If the processor deallocation fails for any reason, you can use the **ha_star** command to restart it after it has been fixed. For information about using this command, see **ha_star** in the *AIX 5L Version 5.3 Commands Reference, Volume 2*.
- Use the **errpt** command to generate a report of logged errors. For information about using this command, see **errpt** in the *AIX 5L Version 5.3 Commands Reference, Volume 2*.

Methods of 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`.

Beginning with AIX 5.2, the default is `enable` (the attribute **cpuguard** has a value of `enable`). System administrators who want to disable this feature must use either the Web-based System Manager system menus, the SMIT **System Environments** menu, or the **chdev** command. (In previous AIX versions, the default was `disable`.)

Note: If processor deallocation is turned off (disabled), the errors are still logged. The error log will contain an error such as `CPU_FAILURE_PREDICTED`, indicating that AIX was notified of a problem with a CPU.

Restarting an aborted processor deallocation

Sometimes the processor deallocation fails because an application did not move its bound threads away from the last logical CPU.

Once this problem has been fixed, either by unbinding (when it is safe to do so) or by 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

There are several things you should consider about processor states.

Physical processors are represented in the ODM database by objects named **proc n** where n is a decimal number that represents the physical processor number. Like any other device represented in the ODM database, processor objects have a state, such as Defined/Available, and attributes.

The state of a **proc** object is always Available as long as the corresponding processor is present, regardless of whether it is usable. The **state** attribute of a **proc** object indicates if the processor is used and, if not, the reason. This attribute can have three values:

enable	The processor is used.
disable	The processor has been dynamically deallocated.
faulty	The processor was declared defective by the firmware at startup time.

If an ailing processor is successfully deallocated, its state goes from **enable** to **disable**. Independently of AIX, this processor is also flagged in the firmware as defective. Upon reboot, the deallocated processor will not be available and will have its state set to **faulty**. The ODM **proc** object, however, is still marked Available. You must physically remove the defective CPU from the system board or remove the CPU board (if possible) for the **proc** object to change to Defined.

In the following example, processor **proc4** is working correctly and is being used by the operating system, as shown in the following output:

```
# lsattr -EH -l proc4
attribute value  description  user_settable

state enable Processor state False
type PowerPC_RS64-III Processor type False
#
```

When processor **proc4** gets a predictive failure, it gets deallocated by the operating system, as shown in the following:

```
# lsattr -EH -l proc4
attribute value  description  user_settable

state disable Processor state False
type PowerPC_RS64-III Processor type False
#
```

At the next system restart, processor **proc4** is reported by firmware as defective, as shown in the following:

```
# lsattr -EH -l proc4
attribute value  description  user_settable

state faulty Processor state False
type PowerPC_RS64-III Processor type False
#
```

But the status of processor **proc4** remains Available, as shown in the following:

```
# lsdev -CH -l proc4
name status location description

proc4 Available 00-04 Processor
#
```

Deallocation error log entries

Three different error log messages are associated with CPU deallocation.

The following are examples.

errpt short format - summary

The following is an example of entries displayed by the **errpt** command (without options):

```
# errpt
IDENTIFIER      TIMESTAMP      T    C    RESOURCE_NAME  DESCRIPTION
804E987A        1008161399    I    O    proc4          CPU DEALLOCATED
8470267F        1008161299    T    S    proc4          CPU DEALLOCATION ABORTED
1B963892        1008160299    P    H    proc4          CPU FAILURE PREDICTED
#
```

- If processor deallocation is enabled, a CPU FAILURE PREDICTED message is always followed by either a CPU DEALLOCATED message or a CPU DEALLOCATION ABORTED message.
- If processor deallocation is not enabled, only the CPU FAILURE PREDICTED message is logged. Enabling processor deallocation any time after one or more CPU FAILURE PREDICTED messages have been logged initiates the deallocation process and results in a success or failure error log entry, as described above, for each processor reported failing.

errpt long format - detailed description

The following is the form of output obtained with **errpt -a**:

- CPU_FAIL_PREDICTED

Error description: Predictive Processor Failure

This error indicates that the hardware detected that a processor has a high probability to fail in a near future. It is always logged whether or not processor deallocation is enabled.

DETAIL DATA: *Physical processor number, location*

Example error log entry - long form

```
LABEL: CPU_FAIL_PREDICTED
IDENTIFIER: 1655419A
```

```
Date/Time: Thu Sep 30 13:42:11
Sequence Number: 53
Machine Id: 00002F0E4C00
Node Id: auntbea
Class: H
Type: PEND
Resource Name: proc25
Resource Class: processor
Resource Type: proc_rspc
Location: 00-25
```

```
Description
CPU FAILURE PREDICTED
```

```
Probable Causes
CPU FAILURE
```

```
Failure Causes
CPU FAILURE
```

```
Recommended Actions
ENSURE CPU GARD MODE IS ENABLED
```


RUN SYSTEM DIAGNOSTICS.

```
Detail Data
PROBLEM DATA
0144 1000 0000 003A 8E00 9100 1842 1100 1999 0930 4019
0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 4942 4D00 5531
2E31 2D50 312D 4332 0000
0002 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000
... ..
```

- CPU_DEALLOC_SUCCESS

Error Description: A processor has been successfully deallocated after detection of a predictive processor failure. This message is logged when processor deallocation is enabled, and when the CPU has been successfully deallocated.

DETAIL DATA: *Logical CPU number of deallocated processor.*

Example: error log entry - long form:

```
LABEL: CPU_DEALLOC_SUCCESS
IDENTIFIER: 804E987A
```

```
Date/Time: Thu Sep 30 13:44:13
Sequence Number: 63
Machine Id: 00002F0E4C00
Node Id: auntbea
Class: 0
Type: INFO
Resource Name: proc24
```

```
Description
CPU DEALLOCATED
```

```
Recommended Actions
MAINTENANCE IS REQUIRED BECAUSE OF CPU FAILURE
```

```
Detail Data
LOGICAL DEALLOCATED CPU NUMBER
```

0

In this example, **proc24** was successfully deallocated and was logical CPU **0** when the failure occurred.

- CPU_DEALLOC_FAIL

Error Description: A processor deallocation, due to a predictive processor failure, was not successful. This message is logged when CPU deallocation is enabled, and when the CPU has not been successfully deallocated.

DETAIL DATA: *Reason code, logical CPU number, additional information depending of the type of failure.*

The reason code is a numeric hexadecimal value. The possible reason codes are:

- 2 One or more processes/threads remain bound to the last logical CPU. In this case, the detailed data give the PIDs of the offending processes.
- 3 A registered driver or kernel extension returned an error when notified. In this case, the detailed data field contains the name of the offending driver or kernel extension (ASCII encoded).

4 Deallocation a processor causes the machine to have less than two available CPUs. This operating system does not deallocate more than $N-2$ processors on an N -way machine to avoid confusing applications or kernel extensions using the total number of available processors to determine whether they are running on a Uni Processor (UP) system where it is safe to skip the use of multiprocessor locks, or a Symmetric Multi Processor (SMP).

200 (0xC8) Processor deallocation is disabled (the ODM attribute **cpuguard** has a value of disable). You normally do not see this error unless you start **ha_star** manually.

Examples: error log entries - long format

Example 1:

```

LABEL: CPU_DEALLOC_ABORTED
IDENTIFIER: 8470267F
Date/Time: Thu Sep 30 13:41:10
Sequence Number: 50
Machine Id: 00002F0E4C00
Node Id: auntbea
Class: S
Type: TEMP
Resource Name: proc26

```

Description
CPU DEALLOCATION ABORTED

Probable Causes
SOFTWARE PROGRAM

Failure Causes
SOFTWARE PROGRAM

Recommended Actions
MAINTENANCE IS REQUIRED BECAUSE OF CPU FAILURE
SEE USER DOCUMENTATION FOR CPU GARD

Detail Data
DEALLOCATION ABORTED CAUSE
0000 0003
DEALLOCATION ABORTED DATA
6676 6861 6568 3200

In this example, 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
```

In this example, the deallocation for **proc19** failed. The reason code 2 indicates thread(s) were bound to the last logical processor and did not unbind after 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 you to list all threads or processes along 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
```

In this example, the deallocation of **proc2** failed because there were two or fewer active processors at the time of failure (reason code 4).

Filesets needed for 64-bit mode

64-bit mode allows for fast access to large amounts of data and efficient handling of 64-bit data types.

The base operating system 64-bit runtime fileset is **bos.64bit**. Installing **bos.64bit** also installs the **/etc/methods/cfg64** file. The **/etc/methods/cfg64** file provides the option of enabling or disabling the 64-bit environment via SMIT, which updates the **/etc/inittab** file with the load64bit line. (Simply adding the load64bit line does not enable the 64-bit environment).

The command **lspp -l bos.64bit** reveals if this fileset is installed. The **bos.64bit** fileset is on the AIX media; however, installing the **bos.64bit** fileset does not ensure that you will be able to run 64-bit software. If the **bos.64bit** fileset is installed on 32-bit hardware, you should be able to compile 64-bit software, but you cannot run 64-bit programs on 32-bit hardware.

The **syscalls64** extension must be loaded in order to run a 64-bit executable. This is done from the `load64bit` entry in the `inittab` file. You must load the **syscalls64** extension even when running a 64-bit kernel on 64-bit hardware.

Hardware required for 64-bit mode

You must have 64-bit hardware to run 64-bit applications.

To determine whether your system has 32-bit or 64-bit hardware architecture:

1. Log in as a root user.
2. At the command line, enter `bootinfo -y`.

This produces the output of either **32** or **64**, depending on whether the hardware architecture is 32-bit or 64-bit. In addition, if you enter `lsattr -El proc0` at any version of AIX, the type of processor for your server displays.

Kernel extensions vs. 64-bit kernel

To determine if the 64-bit kernel extension is loaded, at the command line, enter `genkex |grep 64`.

Information similar to the following displays:

```
149bf58 a3ec /usr/lib/drivers/syscalls64.ext
```

Note: Having the driver extensions does not mean that the kernel is a 64-bit kernel. A 64-bit kernel became available at the AIX 5.1 level. The driver extensions simply allow the 64-bit application to be compiled by a 32-bit kernel. If the 32-bit kernel has a 64-bit processor, **syscalls64.ext** allows the 64-bit application to execute. Yet at the AIX 5L level, a 64-bit kernel and a 64-bit processor has better performance with 64-bit applications.

Commands to change from 32-bit to 64-bit

To truly change the kernel to 64-bit from 32-bit, the system must be at the AIX 5.1 or AIX 5.2 levels.

To change to a 64-bit kernel, enter the following commands:

```
ln -sf /usr/lib/boot/unix_64 /unix
ln -sf /usr/lib/boot/unix_64 /usr/lib/boot/unix
lslv -m hd5
bosboot -ad /dev/ipldevice
shutdown -Fr
```

Commands to change from 64-bit to 32-bit

To change the kernel back to 32-bit from 64-bit, enter the following commands.

```
ln -sf /usr/lib/boot/unix_mp /unix
ln -sf /usr/lib/boot/unix_mp /usr/lib/boot/unix
lslv -m hd5
bosboot -ad /dev/ipldevice
shutdown -Fr
```

32-bit and 64-bit performance comparisons

In most cases, running 32-bit applications on 64-bit hardware is not a problem, because 64-bit hardware can run both 64-bit and 32-bit software. However, 32-bit hardware cannot run 64-bit software.

To find out if any performance issues exist for applications that are running on the system, refer to those application's user guides for their recommended running environment.

Changing the default browser

You can change the default browser.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

This procedure changes the default browser that is used by applications that use the **defaultbrowser** command to open a browser window. The default browser is the browser that is launched when users use the **infocenter** command. This scenario describes how to use SMIT to change the default browser. You can also use Web-based System Manager to change the default browser.

1. Change to root user.
2. On a command line, type:

```
smit change_documentation_services
```
3. In the **DEFAULT_BROWSER** field, type the command that launches your new web browser. Include any flags that are required when a URL is included in the command.

```
anybrowser -u http://www.ibm.com
```

To open your anybrowser with the www.ibm.com web address open inside, you would type anybrowser -u in the field. Many browsers (for example, the Mozilla Web browser) do not require a flag. The browser change takes effect the next time that you log in to the computer.

System environment variable setup

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 login. Most of these variables are set during system initialization. Their definitions are read from the **/etc/profile** file or set by default.

Testing the system battery

If your system is losing track of time, the cause might be a depleted or disconnected battery.

1. To determine the status of your system battery, type the following **diag** command:

```
diag -B -c
```
2. When the Diagnostics main menu appears, select the **Problem Determination** option. If the battery is disconnected or depleted, a problem menu will be displayed with a service request number (SRN). Record the SRN on Item 4 of the Problem Summary Form and report the problem to your hardware service organization.

If your system battery is operational, your system time might have been reset incorrectly because either the **date** or **setclock** command was run incorrectly or unsuccessfully. Refer to “System clock setup” to correct the problem.

System clock setup

The system clock records the time of system events, allows you to schedule system events (such as running hardware diagnostics at 3:00 a.m.), and tells when you first created or last saved files.

Use the **date** command to set your system clock. Use the **setclock** command to set the time and date by contacting a time server.

date command

The **date** command displays or sets the date and time.

Enter the following command to determine what your system recognizes as the current date and time:

```
/usr/bin/date
```

Attention: Do not change the date when the system is running with more than one user.

The following formats can be used when setting the date with the *Date* parameter:

- *mmddHHMM[YYyy]* (default)
- *mmddHHMM[yy]*

The variables to the *Date* parameter are defined as follows:

<i>mm</i>	Specifies the number of the month.
<i>dd</i>	Specifies the number of the day in the month.
<i>HH</i>	Specifies the hour in the day (using a 24-hour clock).
<i>MM</i>	Specifies the minute number.
<i>YY</i>	Specifies the first two digits of a four-digit year.
<i>yy</i>	Specifies the last two numbers of the year.

With root authority, you can use the **date** command to set the current date and time. For example:

```
date 021714252002
```

Sets the date to Feb. 17, 2002, and time to 14:25. For more information about the **date** command, see its description in *AIX 5L Version 5.3 Commands Reference, Volume 2*.

setclock command

The **setclock** command displays or sets the time and date by requesting the current time from a time server on a network.

To display your system's date and time, enter:

```
/usr/sbin/setclock
```

The **setclock** command takes the first response from the time server, converts the calendar clock reading found there, and shows the local date and time. If no time server responds, or if the network is not operational, the **setclock** command displays a message to that effect and leaves the date and time settings unchanged.

Note: Any host running the **inetd** daemon can act as a time server.

With root authority, you can use the **setclock** command to send an Internet TIME service request to a time server host and sets the local date and time accordingly. For example:

```
setclock TimeHost
```

Where *TimeHost* is the host name or IP address of the time server.

See **setclock** for more information.

Message of the day setup

The message of the day is displayed every time a user logs in to the system.

It is a convenient way to communicate information to all users, such as installed software version numbers or current system news. To change the message of the day, use your favorite editor to edit the `/etc/motd` file.

Dynamic Processor Deallocation

AIX can detect and automatically stop using a faulty processor.

Starting with machine type 7044 model 270, the hardware of all systems with two or more processors is able to detect correctable errors, which are gathered by the firmware. These errors are not fatal and, as long as they remain rare occurrences, can be safely ignored. However, when a pattern of failures seems to be developing on a specific processor, this pattern might indicate that this component is likely to exhibit a fatal failure in the near future. This prediction is made by the firmware based on the failure rates and threshold analysis.

On these systems, AIX implements continuous hardware surveillance and regularly polls the firmware for hardware errors. When the number of processor errors hits a threshold and the firmware recognizes that there is a distinct probability that this system component will fail, the firmware returns an error report. In all cases, the error is logged in the system error log. In addition, on multiprocessor systems, depending on the type of failure, 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.

Chapter 8. Process management

The process is the entity that the operating system uses to control the use of system resources. *Threads* can control processor-time consumption, but most system management tools still require you to refer to the process in which a thread is running, rather than to the thread itself.

Tools are available to:

- Observe the creation, cancellation, identity, and resource consumption of processes
 - The **ps** command is used to report process IDs, users, CPU-time consumption, and other attributes.
 - The **who -u** command reports the shell process ID of logged-on users.
 - The **svmon** command is used to report process real-memory consumption. (See *Performance Toolbox Version 2 and 3 for AIX: Guide and Reference* for information on the **svmon** command.)
 - The **acct** command mechanism writes records at process termination summarizing the process's resource use. (See how to set up an accounting system in Chapter 11, "System accounting," on page 245.)
- Control the priority level at which a process contends for the CPU.
 - The **nice** command causes a command to be run with a specified process priority.
 - The **renice** command changes the priority of a given process.
- Terminate processes that are out of control.
 - The **kill** command sends a termination signal to one or more processes.

Process monitoring

You, as the system administrator, can manage processes.

The **ps** command is the primary tool for observing the processes in the system. Most of the flags of the **ps** command fall into one of two categories:

- Flags that specify which types of processes to include in the output
- Flags that specify which attributes of those processes are to be displayed

The most widely useful variants of **ps** for system-management purposes are:

ps -ef	Lists all nonkernel processes, with the userid, process ID, recent CPU usage, total CPU usage, and the command that started the process (including its parameters).
ps -fu <i>UserID</i>	Lists all of the processes owned by <i>UserID</i> , with the process ID, recent CPU usage, total CPU usage, and the command that started the process (including its parameters).

To identify the current heaviest users of CPU time, you could enter:

```
ps -ef | egrep -v "STIME|LOGNAME" | sort +3 -r | head -n 15
```

This command lists, in descending order, the 15 most CPU-intensive processes other than those owned by you.

For more specialized uses, the following two tables are intended to simplify the task of choosing **ps** flags by summarizing the effects of the flags.

Process-Specifying Flags

	-A	-a	-d	-e	-G -g	-k	-p	-t	-U -u	a	g	t	x
All processes	Y	-	-	-	-	-	-	-	-	-	Y	-	-

Process-Specifying Flags

	-A	-a	-d	-e	-G -g	-k	-p	-t	-U -u	a	g	t	x
Not processes group leaders and not associated with a terminal	-	Y	-	-	-	-	-	-	-	-	-	-	-
Not process group leaders	-	-	Y	-	-	-	-	-	-	-	-	-	-
Not kernel processes	-	-	-	Y	-	-	-	-	-	-	-	-	-
Members of specified-process groups	-	-	-	-	Y	-	-	-	-	-	-	-	-
Kernel processes	-	-	-	-	-	Y	-	-	-	-	-	-	-
Those specified in process number list	-	-	-	-	-	-	Y	-	-	-	-	-	-
Those associated with tty(s) in the list	-	-	-	-	-	-	-	Y (n ttys)	-	-	-	Y (1 tty)	-
Specified user processes	-	-	-	-	-	-	-	-	Y	-	-	-	-
Processes with terminals	-	-	-	-	-	-	-	-	-	Y	-	-	-
Not associated with a tty	-	-	-	-	-	-	-	-	-	-	-	-	Y

Column-Selecting Flags

Default1	-f	-l	-U -u	Default2	e	l	s	u	v	
PID	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
TTY	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
TIME	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
CMD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
USER	-	Y	-	-	-	-	-	-	Y	-
UID	-	-	Y	Y	-	-	Y	-	-	-
PPID	-	Y	Y	-	-	-	Y	-	-	-
C	-	Y	Y	-	-	-	Y	-	-	-

Column-Selecting Flags

Default1	-f	-l	-U -u	Default2	e	l	s	u	v	
STIME	-	Y	-	-	-	-	-	-	Y	-
F	-	-	Y	-	-	-	-	-	-	-
S/STAT	-	-	Y	-	Y	Y	Y	Y	Y	Y
PIR	-	-	Y	-	-	-	Y	-	-	-
NI/NICE	-	-	Y	-	-	-	Y	-	-	-
ADDR	-	-	Y	-	-	-	Y	-	-	-
SIZE	-	-	-	-	-	-	-	-	Y	-
SZ	-	Y	-	-	-	Y	-	Y	-	-
WCHAN	-	-	Y	-	-	-	Y	-	-	-
RSS	-	-	-	-	-	-	Y	-	Y	Y
SSIZ	-	-	-	-	-	-	-	Y	-	-
%CPU	-	-	-	-	-	-	-	-	Y	Y
%MEM	-	-	-	-	-	-	-	-	Y	Y
PGIN	-	-	-	-	-	-	-	-	-	Y
LIM	-	-	-	-	-	-	-	-	-	Y
TSIZ	-	-	-	-	-	-	-	-	-	Y
TRS	-	-	-	-	-	-	-	-	-	Y
<i>Environment</i> (following the command)	-	-	-	-	-	Y	-	-	-	-

If **ps** is given with no flags or with a process-specifying flag that begins with a minus sign, the columns displayed are those shown for Default1. If the command is given with a process-specifying flag that does not begin with minus, Default2 columns are displayed. The **-u** or **-U** flag is both a process-specifying and column-selecting flag.

The following are brief descriptions of the contents of the columns:

PID	Process ID
TTY	Terminal or pseudo-terminal associated with the process
TIME	Cumulative CPU time consumed, in minutes and seconds
CMD	Command the process is running
USER	Login name of the user to whom the process belongs
UID	Numeric user ID of the user to whom the process belongs
PPID	ID of the parent process of this process
C	Recently used CPU time
STIME	Time the process started, if less than 24 hours. Otherwise the date the process is started
F	Eight-character hexadecimal value describing the flags associated with the process (see the detailed description of the ps command)
S/STAT	Status of the process (see the detailed description of the ps command)
PRI	Current priority value of the process
NI/NICE	Nice value for the process
ADDR	Segment number of the process stack
SIZE	(-v flag) The virtual size of the data section of the process (in kilobytes)
SZ	(-l and l flags) The size in kilobytes of the core image of the process.
WCHAN	Event on which the process is waiting
RSS	Sum of the numbers of working-segment and code-segment pages in memory times 4

SSIZ	Size of the kernel stack
%CPU	Percentage of time since the process started that it was using the CPU
%MEM	Nominally, the percentage of real memory being used by the process, this measure does not correlate with any other memory statistics
PGIN	Number of page ins caused by page faults. Since all I/O is classified as page faults, this is basically a measure of I/O volume
LIM	Always xx
TSIZ	Size of the text section of the executable file
TRS	Number of code-segment pages times 4
<i>Environment</i>	Value of all the environment variables for the process

Process priority alteration

Basically, if you have identified a process that is using too much CPU time, you can reduce its effective priority by increasing its nice value with the **renice** command.

For example:

```
renice +5 ProcID
```

The nice value of the *ProcID*'s would increase process from the normal 20 of a foreground process to 25. You must have root authority to reset the process *ProcID*'s nice value to 20. Type:

```
renice -5 ProcID
```

Process termination

Normally, you use the **kill** command to end a process.

The **kill** command sends a signal to the designated process. Depending on the type of signal and the nature of the program that is running in the process, the process might end or might keep running. The signals you send are:

SIGTERM	(signal 15) is a request to the program to terminate. If the program has a signal handler for SIGTERM that does not actually terminate the application, this kill may have no effect. This is the default signal sent by kill .
SIGKILL	(signal 9) is a directive to kill the process immediately. This signal cannot be caught or ignored.

It is typically better to issue SIGTERM rather than SIGKILL. If the program has a handler for SIGTERM, it can clean up and terminate in an orderly fashion. Type:

```
kill -term ProcessID
```

(The **-term** could be omitted.) If the process does not respond to the SIGTERM, type:

```
kill -kill ProcessID
```

You might notice occasional defunct processes, also called *zombies*, in your process table. These processes are no longer executing, have no system space allocated, but still retain their PID number. You can recognize a zombie process in the process table because it displays <defunct> in the CMD column. For example:

```
UID  PID  PPID  C   STIME  TTY  TIME CMD
      .
      .
      .
lee 22392 20682  0   Jul 10   -   0:05 xclock
lee 22536 21188  0   Jul 10 pts/0   0:00 /bin/ksh
lee 22918 24334  0   Jul 10 pts/1   0:00 /bin/ksh
lee 23526 22536 22           0:00 <defunct>
```

```

lee 24334 20682 0 Jul 10 ? 0:00 aixterm
lee 24700 1 0 Jul 16 ? 0:00 aixterm
root 25394 26792 2 Jul 16 pts/2 0:00 ksh
lee 26070 24700 0 Jul 16 pts/3 0:00 /bin/ksh
lee 26792 20082 0 Jul 10 pts/2 0:00 /bin/ksh
root 27024 25394 2 17:10:44 pts/2 0:00 ps -ef

```

Zombie processes continue to exist in the process table until the parent process dies or the system is shut down and restarted. In the example shown above, the parent process (PPID) is the **ksh** command. When the Korn shell is exited, the defunct process is removed from the process table.

Sometimes a number of these defunct processes collect in your process table because an application has forked several child processes and has not exited. If this becomes a problem, the simplest solution is to modify the application so its **sigaction** subroutine ignores the **SIGCHLD** signal. For more information, see the **sigaction** subroutine in *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 2*.

Binding or unbinding a process

You can bind a process to a processor or unbind a previously bound process.

You must have root user authority to bind or unbind a process you do not own.

On multiprocessor systems, you can bind a process to a processor or unbind a previously bound process from:

- Web-based System Manager
- SMIT
- command line

Note: While binding a process to a processor might lead to improved performance for the bound process (by decreasing hardware-cache misses), overuse of this facility could cause individual processors to become overloaded while other processors are underused. The resulting bottlenecks could reduce overall throughput and performance. During normal operations, it is better to let the operating system assign processes to processors automatically, distributing system load across all processors. Bind only those processes that you know can benefit from being run on a single processor.

Binding or Unbinding a Process Tasks

<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Binding a Process	smit bindproc	bindprocessor -q
Unbinding a Process	smit ubindproc	bindprocessor -u

Fixes for stalled or unwanted processes

Stalled or unwanted processes can cause problems with your terminal. Some problems produce messages on your screen that give information about possible causes.

To perform the following procedures, you must have either a second terminal, a modem, or a network login. If you do not have any of these, fix the terminal problem by rebooting your machine.

Choose the appropriate procedure for fixing your terminal problem:

- “Freeing a terminal taken over by processes” on page 192
- “Responding to screen messages” on page 192

Freeing a terminal taken over by processes

You can stop stalled or unwanted process.

Identify and stop stalled or unwanted processes by doing the following:

1. Determine the active processes running on the screen by typing the following **ps** command:

```
ps -ef | pg
```

The **ps** command shows the process status. The **-e** flag writes information about all processes (except kernel processes), and the **f** flag generates a full listing of processes including what the command name and parameters were when the process was created. The **pg** command limits output to a single page at a time, so information does not quickly scroll off the screen.

Suspicious processes include system or user processes that use up excessive amounts of a system resource such as CPU or disk space. System processes such as **sendmail**, **routed**, and **lpd** frequently become runaways. Use the **ps -u** command to check CPU usage.

2. Determine who is running processes on this machine by using the **who** command:

```
who
```

The **who** command displays information about all users currently on this system, such as login name, workstation name, date, and time of login.

3. Determine if you need to stop, suspend, or change the priority of a user process.

Note: You must have root authority to stop processes other than your own. If you terminate or change the priority of a user process, contact the process owner and explain what you have done.

- Stop the process using the **kill** command. For example:

```
kill 1883
```

The **kill** command sends a signal to a running process. To stop a process, specify the process ID (PID), which is 1883 in this example. Use the **ps** command to determine the PID number of commands.

- Suspend the process and run it in the background by using the ampersand (&). For example:

```
/u/bin1/prog1 &
```

The **&** signals that you want this process to run in the background. In a background process, the shell does not wait for the command to complete before returning the shell prompt. When a process requires more than a few seconds to complete, run the command in background by typing an **&** at the end of the command line. Jobs running in the background appear in the normal **ps** command.

- Change the priority of the processes that have taken over by using the following **renice** command:

```
renice 20 1883
```

The **renice** command alters the scheduling priority of one or more running processes. The higher the number, the lower the priority with 20 being the lowest priority.

In the previous example, **renice** reschedules process number 1883 to the lowest priority. It will run when there is a small amount of unused processor time available.

Responding to screen messages

Use this procedure to respond to and recover from screen messages.

1. Make sure the **DISPLAY** environment variable is set correctly. Use either of the following methods to check the **DISPLAY** environment:

- Use the **setse** command to display the environment variables.

```
setse
```

The **setse** command displays the protected state environment when you logged in.

Determine if the **DISPLAY** variable has been set. In the following example, the **DISPLAY** variable does not appear, which indicates that the **DISPLAY** variable is not set to a specific value.

```
SYSENVIRON:  
NAME=casey  
TTY=/dev/pts/5  
LOGNAME=casey  
LOGIN=casey
```

OR

- Change the value of the **DISPLAY** variable. For example, to set it to the machine named bastet and terminal 0, enter:

```
DISPLAY=bastet:0  
export DISPLAY
```

If not specifically set, the **DISPLAY** environment variable defaults to `unix:0` (the console). The value of the variable is in the format *name:number* where *name* is the host name of a particular machine, and *number* is the X server number on the named system.

2. Reset the terminal to its defaults using the following **stty** command:

```
stty sane
```

The **stty sane** command restores the “sanity” of the terminal drivers. The command outputs an appropriate terminal resetting code from the `/etc/termcap` file (or `/usr/share/lib/terminfo` if available).

3. If the Return key does not work correctly, reset it by typing:

```
^J stty sane ^J
```

The `^J` represents the Ctrl-J key sequence.

RT_MPC and RT_GRQ

The use of multiple queues increases the processor affinity of threads, but there is a special situation where you might want to counteract this effect.

When there is only one run queue, a thread that has been awakened (the waking thread) by another running thread (the waker thread) would normally be able to use the CPU immediately on which the waker thread was running. With multiple run queues, the waking thread may be on the run queue of another CPU which cannot notice the waking thread until the next scheduling decision. This may result in up to a 10 ms delay.

This is similar to scenarios in earlier releases of this operating system which might have occurred using the `bindprocessor` option. If all CPUs are constantly busy, and there are a number of interdependent threads waking up, there are two options available.

- The first option, which uses one run queue, is to set the environment variable **RT_GRQ=ON** which forces unbound selected threads to be dispatched off the global run queue.
- Alternatively, users can choose the real time kernel option (type the command `bosdebug -R on` and then `bosboot`) and the **RT_MPC=ON** environment variable for selected processes. It is essential to maintain a performance log of your systems to closely monitor the impact of any tuning you attempt.

Chapter 9. Workload management

Workload Manager (WLM) is designed to provide the system administrator with increased control over how the scheduler virtual memory manager (VMM) and the disk I/O subsystem 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.

Attention: Efficient use of WLM requires extensive knowledge of existing system processes and performance. If the system administrator configures WLM with extreme or inaccurate values, performance will be significantly degraded.

WLM is primarily intended for use with large systems. Large systems are often used for server consolidation, in which workloads from many different server systems (such as printer, database, general user, and transaction processing systems) are combined into a single large system to reduce the cost of system maintenance. These workloads often interfere with each other and have different goals and service agreements.

WLM also provides isolation between user communities with very different system behaviors. This can prevent effective starvation of workloads with certain behaviors (for example, interactive or low CPU usage jobs) by workloads with other behaviors (for example, batch or high memory usage jobs).

Also, WLM ties into the accounting subsystem (see Chapter 11, “System accounting,” on page 245) allowing users to do resource usage accounting per WLM class in addition to the standard accounting per user or group.

Workload management concepts

With WLM, you can create different classes of service for jobs, as well as specify attributes for those classes.

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

Terminology for workload management

Common terms associated with workload management are listed and described in this table.

class	A <i>class</i> is a collection of processes and their associated threads. A class has a single set of resource-limitation values and target shares. <i>class</i> is used to describe both subclasses and superclasses.
superclass	A <i>superclass</i> is a class that has subclasses associated with it. No processes can belong to a superclass without also belonging to a subclass. A superclass has a set of class-assignment rules that determines which processes are assigned to the superclass. A superclass also has a set of resource-limitation values and resource target shares that determines the amount of resources that can be used by processes which belong to the superclass. These resources are divided among the subclasses based on the resources limitation values and resource target shares of the subclasses.

subclasses	<p>A <i>subclass</i> is a class associated with exactly one superclass. Every process in a subclass is also a member of its superclass. Subclasses have access only to resources that are available to the superclass. A subclass has a set of class assignment rules that determines which of the processes assigned to the superclass belong to the subclass. A subclass also has a set of resource-limitation values and resource target shares that determines the resources that can be used by processes in the subclass.</p> <p>These resource-limitation values and resource target shares indicate how much of the resources available to the superclass (the target for the superclass) can be used by processes in the subclass.</p> <p>WLM administration can be done using either the Web-based System Manager, SMIT, or the WLM command-line interface.</p>
classification mechanism	<p>A <i>classification mechanism</i> is a set of class assignment rules that determines which processes are assigned to which classes (superclasses or subclasses within superclasses).</p>
class assignment rule	<p>A <i>class assignment rule</i> indicates which values within a set of process attributes result in a process being assigned to a particular class (superclass or subclass within a superclass).</p>
process attribute value	<p>A <i>process attribute value</i> is the value that a process has for a process attribute. The process attributes can include attributes such as user ID, group ID, and application path name.</p>
resource-limitation values	<p><i>Resource-limitation values</i> are a set of values that WLM maintains for a set of resource utilization values. These limits are completely independent of the resource limits specified with the setrlimit subroutine.</p>
resource target share	<p><i>Resource target shares</i> are the shares of a resource that are available to a class (subclass or superclass). These shares are used with other class shares (subclass or superclass) at the same level and tier to determine the desired distribution of the resources between classes at that level and tier.</p>
resource-utilization value	<p>A <i>resource-utilization value</i> is the amount of a resource that a process or set of processes is currently using in a system. Whether it is one process or a set of processes is determined by the scope of process resource collection.</p>
scope-of-resource collection	<p>The <i>scope-of-resource collection</i> is the level at which resource utilization is collected and the level at which resource-limitation values are applied. This might be at the level of each process in a class, the level of the sum across every process in a class owned by each user, or the level of the sum across every process in a class. The only scope currently supported is the latter.</p>
process class properties	<p>The <i>process class properties</i> are the set of properties that are given to a process based on the classes (subclass and superclass) to which it is assigned.</p>
class authorizations	<p>The <i>class authorizations</i> are a set of rules that indicates which users and groups are allowed to perform operations on a class or processes and threads in a class. This includes the authorization to manually assign processes to a class or to create subclasses of a superclass.</p>
class tier	<p>The <i>class tier</i> value is the position of the class within the hierarchy of resource limitation desirability for all classes. The resource limits (including the resource targets) for all classes in a tier are satisfied before any resource is provided to lower tier classes. Tiers are provided at both the superclass and subclass levels. Resources are provided to superclasses based on their tiers. Within a superclass, resources are given to subclasses based on their tier values within the superclass. Thus, superclass tier is the major differentiator in resource distribution; the subclass tier provides an additional smaller differentiator within a superclass.</p>

Classes for workload management

WLM allows system administrators to define classes and define for each class a set of attributes and resource limits.

The processes are assigned to classes based on criteria provided by the system administrator. The resource entitlements and limits are enforced at the class level. This method of defining classes of service and regulating the resource utilization of each class of applications prevents applications with very different resource use patterns from interfering with each other when they share a single server.

WLM supports a hierarchy of classes with two levels:

- The resources of the system are distributed among superclasses according to the resource entitlements for each superclass. The system administrator defines resource entitlements.
- In turn, each superclass can have subclasses. The resources allocated to the superclass are distributed among the subclasses according to the resource entitlements given to each subclass.
- The system administrator can delegate the administration of the subclasses of each superclass to a *superclass administrator* or to a group of superclass administrators.
- In AIX 5.2 and later, WLM supports up to 69 superclasses (64 user-defined) and 64 subclasses per superclass (61 user-defined).
- Depending on the needs of the organization, a system administrator can decide to use only superclasses or to use superclasses and subclasses.

Note: Throughout this discussion of WLM, the term *class* applies to both superclasses and subclasses. If discussion applies only to a specific class type, that type is explicitly mentioned.

Process assignment to classes for workload management

The processes are assigned to a class, using class-assignment rules provided by the system administrator. The classification criteria are based on the value of a set of attributes of the process such as user ID, group ID, name of the application file, type of process, and application tag.

A defined set of rules is used to determine the superclass a process is assigned to. If this superclass has subclasses defined, there is another set of rules for this superclass to determine which subclass is assigned to which process. This automatic assignment process also takes into account the **inheritance** attributes of both the superclass and the subclass. (For information on class attributes, see “Class attributes” on page 215.)

The automatic class assignment is done when a process calls the **exec** subroutine. The class assignment is reevaluated when a process uses a subroutine that can alter a process attribute used for classification purposes. Examples are the **setuid**, **setgid**, **setpri**, and **plock** subroutines.

In addition to this automatic class assignment, a user with the proper authority can manually assign processes or groups of processes to a specific superclass or subclass.

Resource control

WLM allows management of resources in two ways: as a percentage of available resources or as total resource usage.

Resources that can be controlled on a percentage basis include the following:

- Processor use of the threads of type SCHED_OTHER in a class. This is the sum of all of the processor cycles consumed by every thread in the class. Fixed-priority threads are non-adjustable. Therefore, they cannot be altered, and they can exceed the processor usage target.
- Physical memory utilization of the processes in a class. This is the sum of all the memory pages that belong to the processes in the class.

- Disk I/O bandwidth of the class. This is the bandwidth (in 512-byte blocks per second) of all the I/Os started by threads in the class on each disk device accessed by the class.

Resources that can be controlled on a total usage basis fall into one of two categories: class totals or process totals. The class totals category includes:

Number of processes in a class

This is the number of processes that are active in a class at one time.

Number of threads in a class

This is the number of threads that are active in a class at one time.

Number of logins in a class

This is the number of login sessions that are active in a class at one time.

The process totals category includes:

Total CPU time

This is the total accumulated CPU time for a single process.

Total disk I/O

This is the total accumulated blocks of disk I/O for a single process.

Total connect time

This is total amount of time that a login session can be active.

Resource entitlements

WLM allows system administrators to specify per-class resource entitlements independently for each resource type.

These entitlements can be specified by indicating the following:

- The target for usage of different types of resources. This target is specified with shares. The shares are specified as relative amounts of usage between different classes. For instance, if two classes have respectively 1 and 3 shares of CPU and are the only classes active at this time, their percentage goal used by WLM for its CPU regulation will be 25% and 75%, respectively. The target percentages are calculated for classes in each tier based on the number of active shares in the tier and the amount of resource available to the tier.
- Minimum and maximum limits. These limits are specified as percentages of the total resource available. WLM supports two kinds of maximum limits:
 - A soft maximum limit indicates the maximum amount of the resource that can be made available when there is contention for the resource. This maximum can be exceeded if there is no contention; that is, if no one else requires the resource.
 - A hard maximum limit indicates the maximum amount of the resource that can be made available regardless of whether there is contention on the resource. Fixed-priority threads, however, are not subject to these same rules and therefore can exceed the limit.
- Total limits. The total limits are strictly enforced. If a process exceeds one of its total consumption limits, it will be terminated. If a class is at one of its total limits, any operation that would result in the creation of another instance of that resource will fail.

In most cases, soft maximum limits are sufficient to ensure that resource entitlements are met and enforced. Using hard maximum limits may result in unused system resources since these are strictly enforced, even when there is no contention for the resource. Careful consideration must be made when using hard maximum limits since these can greatly affect system or application performance if set too low. Total limits should also be used with caution, since these could result in process termination or failure to function as intended.

In active mode, WLM attempts to keep active classes close to their targets. Since there are few constraints on the values of the various limits, the sum of any of the limits across all classes could far

exceed 100%. In this case, if all of the classes are active, the limit cannot be reached by all classes. WLM regulates the processor consumption by adjusting the scheduling priorities of the non-fixed priority threads in the system according to how the class they belong to is performing, relative to its limits and target. This approach guarantees a processor consumption averaged over a given period of time, rather than the processor consumption over very short intervals (for example, 10 ms).

For example, if class A is the only active class, with a processor minimum of 0% and a processor target of 60 shares, then it gets 100% of the processor. If class B, with a processor minimum limit of 0% and a processor target of 40 shares, becomes active, then the class A processor utilization progressively decreases to 60% and the class B processor utilization increases from 0% to 40%. The system stabilizes at 60% and 40% processor utilization, respectively, in a matter of seconds.

This example supposes that there is no memory contention between the classes. Under regular working conditions, the limits you set for processor and memory are interdependent. For example, a class may be unable to reach its target or even its minimum processor allocation if the maximum limit on its memory usage is too low compared to its working set.

To help refine the class definition and class limits for a given set of applications, WLM provides the **wlmstat** reporting tool, which shows the amount of resource currently being used by each class. A graphical display tool, **wlmmmon**, is also provided for system monitoring.

Workload Manager virtual memory limits

Workload Manager (WLM) virtual memory limits provide administrators a means to prevent system degradation or system failure due to excessive paging by providing a virtual memory limit on a class or a process.

When a limit is exceeded, WLM takes action by doing one of the following:

- killing all processes under the WLM class that exceeded its limit
- killing only the process that caused the WLM class usage to exceed its limit
- killing the process that exceeded its process limit

Virtual memory limits can be specified for any user-defined class, any default subclass under a user-defined super class, and the default super class.

For accounting purposes, WLM will only consider the following as virtual memory when determining WLM total class or process usage:

- heap
- loader initialized data, BSS, shared library, and privately loaded segments
- UBLOCK and mmap areas
- large and pinned user space pages

An administrator can specify a WLM virtual memory limit for a class or for each process in the class. When a class limit is exceeded, WLM can either kill all processes assigned to the class, or only kill the process that caused the limit to be exceeded, depending on whether the **vmenforce** class attribute is set to `class` or `proc`, respectively. The default behavior is to only kill the process that caused the limit to be exceeded. A process limit is killed if the virtual memory use of the process surpasses the limit.

Modes of operation for Workload Manager

WLM can be used to regulate resource consumption as per-class percentages, per-class totals, or per-process totals. Regulation for all resource types can be enabled by running WLM in active mode.

Optionally, you can start a mode of WLM that classifies new and existing processes and monitors the resource usage of the various classes, without attempting to regulate this usage. This mode is called the *passive mode*.

The passive mode can be used when configuring WLM on a new system to verify the classification and assignment rules, and to establish a base line of resource utilization for the various classes when WLM does *not* regulate the processor and memory allocation. This should give a basis for system administrators to decide how to apply the resource shares and resource limits (if needed) to favor critical applications and restrict less important work in order to meet their business goals.

If processor time is the only resource that you are interested in regulating, you can choose to run WLM in active mode for processor and passive mode for all other resources. This mode is called *cpu only* mode. If you want to regulate per-class percentages, but neither of the total resource types, the total resource accounting and regulation can be disabled for per-class totals, per-process totals, or both. In all modes, you have the option of disabling resource set binding.

Dynamic control of Workload Manager

When WLM is active, any parameter of the current configuration can be modified at any time, including the attributes of a class, its resource shares and limits, the assignment rules, and adding new classes or deleting existing classes.

This can be done in several ways, such as:

- Modifying the property files for the currently active configuration (directory pointed to by the symbolic link **/etc/wlm/current**) and refreshing WLM by using the **wlmcntrl** command to use the new parameters.
- Creating another configuration with a different set of parameters and updating WLM to load the parameters of the new configuration, thus making it the current configuration.
- Modifying some of the parameters of the currently active configuration using the WLM command line interface (the **mkclass**, **chclass**, and **rmclass** commands).
- Modifying some of the parameters of the currently active configuration from an application using the WLM APIs.

Automatic switches to a new configuration at specified times of day can be accomplished using *configuration sets*. Configuration sets allow the administrator to specify a set of configurations to be used, and a time range for which each will be active.

Monitoring tools

Use these WLM commands to display WLM statistics and monitor the operation of WLM.

- The **wlmstat** command is text oriented and displays statistics as text (percentage of resource utilization per class for all the resource types managed by WLM).
- The **wlmmmon** command gives a graphical view of per-class resource utilization and WLM regulation.
- The **wlmpperf** command is an optional tool available with the Performance Toolbox and provides more capabilities, such as long-term record and replay.

Per class accounting

The AIX accounting system utility lets you collect and report the use of various system resources by user, group, or WLM class.

When process accounting is turned on, the operating system records statistics about the process resource usage in an accounting file when the process exits. Beginning with AIX 5.1, this accounting record includes a 64-bit numeric key representing the name of the WLM class that the process belonged to. (See Chapter 11, "System accounting," on page 245 for more information about the accounting system utility.)

The accounting subsystem uses a 64-bit key instead of the full 34-character class name to save space (otherwise the change would practically double the size of the accounting record). When the accounting command is run to extract the per-process data, the key is translated back into a class name using the above-mentioned routine. This translation uses the class names currently in the WLM configuration files.

So, if a class has been deleted between the time the accounting record was written, when the process terminated, and the time the accounting report is run, the class name corresponding to the key cannot be found, and the class displays as Unknown.

To keep accurate records of the resource usage of classes deleted during an accounting period, do one of the following:

- Instead of deleting the class, keep the class name in the classes file and remove the class from the rules file so that no process can be assigned to it. Then you can delete the class after the accounting report has been generated at the end of the accounting period.
- Or, delete the class from the configuration it belongs to, and keep the class name in the classes file in a "dummy" configuration (one that is never activated) until after the accounting records for the period have been generated.

Workload Manager

Workload Manager (WLM) gives system administrators more control over how the scheduler and the virtual memory manager (VMM) allocate resources to processes. Using WLM, you can prevent different classes of jobs from interfering with each other and you can allocate resources based on the requirements of different groups of users.

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

WLM is part of the base operating system and is installed with the base operating system, but it is an optional service. It must be configured to suit your system environment, started when you want to use it, and stopped when you want to suspend or end WLM service.

This section contains procedures for configuring WLM with classes and rules that are appropriate for your site and suggestions for troubleshooting unexpected resource consumption behavior. For information on troubleshooting WLM, see *Workload Manager troubleshooting guidelines*. Also, "Configuring Workload Manager to consolidate workloads" on page 207 provides instructions for creating an example configuration.

The tasks in this section assume you are familiar with WLM concepts provided in *Workload management concepts*.

Attention: Efficient use of WLM requires extensive knowledge of existing system processes and performance. If the system administrator configures WLM with extreme or inaccurate values, performance will be significantly degraded.

Startup and shutdown

WLM is an optional service that must be started and stopped.

It is recommended that you use one of the system management interfaces, Web-based System Manager or SMIT, to start or stop WLM.

- To start or stop WLM using Web-based System Manager, select the Workload Manager icon from the session window.
- To start or stop WLM using SMIT, use the `smit wlmmanage` fast path.

The key difference between these options is permanence. In Web-based System Manager or SMIT, you can start or stop WLM three ways:

current session

If you request to stop WLM with this option, WLM will be stopped for this session only and restarted at next reboot. If you request a start with this option, WLM will be started for this session only and not restarted at next reboot.

next reboot

If you request to stop WLM with this option, WLM will remain running for this session only and *will not be* restarted at next reboot. If you request a start with this option, WLM will not be available for this session, but will be started at next reboot.

both

If you request to stop WLM with this option, WLM will be stopped for this session only and *will not be* restarted at next reboot. If you request a start with this option, WLM will be started for this session only *and* will be restarted at next reboot.

You can also use the **wlmcntrl** command, but the **wlmcntrl** command allows you to start or stop WLM for the current session only. If you want to use the command line interface and you want the change to remain in effect when the machine is rebooted, you must edit the **/etc/inittab** file.

WLM can be used to regulate resource consumption as per-class percentages, per-class totals, or per-process totals. Regulation for all resource types can be enabled by running WLM in *active* mode. Optionally, you can start a mode of WLM that classifies new and existing processes and monitors the resource usage of the various classes, without attempting to regulate this usage. This mode is called the *passive* mode. If CPU time is the only resource that you are interested in regulating, you can choose to run WLM in active mode for CPU and passive mode for all other resources. This mode is called *cpu only* mode.

All processes existing in the system before WLM is started are classified according to the newly loaded assignment rules, and are monitored by WLM.

Workload Manager properties

You can specify the properties for the WLM configuration by using the Web-based System Manager, SMIT, the WLM command line interface, or by creating flat ASCII files. The Web-based System Manager and SMIT interfaces use the WLM commands to record the information in the same flat ASCII files, called *property files*.

A set of WLM property files defines a WLM configuration. You can create multiple sets of property files, defining different configurations of workload management. These configurations are located in subdirectories of **/etc/wlm**. The WLM property files describing the superclasses of the *Config* configuration are the file's *classes*, *description*, *limits*, *shares* and *rules* in **/etc/wlm/Config**. Then, the property file's describing the subclasses of the superclass *Super* of this configuration are the file's *classes*, *limits*, *shares* and *rules* in directory **/etc/wlm/Config/Super**. Only the root user can start or stop WLM, or switch from one configuration to another.

The property files are named as follows:

classes	Class definitions
description	Configuration description text
groupings	Attribute value groupings
limits	Class limits
shares	Class target shares
rules	Class assignment rules

The command to submit the WLM property files, **wlmcntrl**, and the other WLM commands allow users to specify an alternate directory name for the WLM properties files. This allows you to change the WLM properties without altering the default WLM property files.

A symbolic link, **/etc/wlm/current**, points to the directory containing the current configuration files. Update this link with the **wlmcntrl** command when you start WLM with a specified configuration or configuration set. The sample configuration files shipped with the operating system are in **/etc/wlm/standard**.

Creating an attribute value grouping

You can group attribute values and represent them with a single value in the **rules** file. These *attribute value groupings* are defined in a **groupings** file within the WLM configuration directory.

By default, a configuration has no **groupings** file. There is no command or management interface to create one. To create and use attribute value groupings, use the following procedure:

1. With root authority, change to the appropriate configuration directory, as shown in the following example:

```
cd /etc/wlm/MyConfig
```

2. Use your favorite editor to create and edit a file named **groupings**. For example:

```
vi groupings
```

3. Define attributes and their associated values using the following format:

```
attribute = value, value, ...
```

All values must be separated by commas. Spaces are not significant. Ranges and wild cards are allowed. For example:

```
trusted = user[0-9][0-9], admin*
nottrusted = user23, user45
shell = /bin/?sh, \
        /bin/sh, \
        /bin/tcsh
rootgroup=system,bin,sys,security,cron,audit
```

4. Save the file.
5. To use attribute groupings within the selection criteria for a class, edit the **rules** file. The attribute grouping name must be preceded by a dollar sign (\$) to include the corresponding values or the exclamation point (!) to exclude the values. The exclamation point cannot be used in the members of the group (step 3), and it is the only modifier that can be used in front of the grouping in this rules file. In the following example, the asterisk (*) signals a comment line:

```
*class resvd user group application type tag
classA - $trusted,!$nottrusted - - -
classB - - - $shell,!/bin/zsh - -
classC - - $rootgroup - - -
```

6. Save the file.

At this point, your classification rules includes attribute value groupings. When the rules are parsed, if an element begins with a \$, the system looks for that element within the **groupings** file. If an element is syntactically invalid or if the **groupings** file does not exist, the system displays a warning message and continues processing other rules.

Creating a time-based configuration set

You can create a set of specialty configurations and assign each configuration within the set to days and times when you want a specific configuration to be in effect.

These sets, called time-based *configuration sets*, are completely separate from but compatible with your normal configuration. You can use the **wlmcntrl -u** command to switch between a configuration set and your normal configuration as needed.

When using a configuration set, you associate existing named configurations, typically with a specific time range. Because only one configuration can be used at any given time, each specified time range must be unique; time ranges cannot overlap or be duplicated.

The **wlmd** daemon alerts WLM when a specified configuration goes out of time range and another configuration needs to be used. Only the root user can manage these time ranges, which are specified within the configuration set's directory in an ASCII file called **.times**.

Use the following procedure to create a time-based configuration set:

1. With root authority, create a configuration set directory then change to that directory. For example:

```
mkdir /etc/wlm/MyConfigSet
cd /etc/wlm/MyConfigSet
```

2. Use your favorite editor to create the configuration set's **.times** file and specify the configuration and time ranges in the following format:

```
ConfigurationName:
    time = "N-N,HH:MM-HH:MM"
```

or

```
ConfigurationName:
    time = -
```

(no time value specified) Where *N* is a numeral representing a day of the week in the range of 0 (Sunday) through 6 (Saturday), *HH* represents the hour in the range of 00 (midnight) to 23 (11 p.m.), and *MM* represents the minutes in the range of 00 to 59. You can specify the day only or not at all. An hour value of 24 is valid for the ending hour of the day, provided that the minute value is 00. If you type a dash (-) instead of a time range for a particular configuration, that configuration will be used when the other configurations' time ranges are not in effect. Only one configuration can be specified without a time range.

For example:

```
conf1:
    time =
conf2:
    time = "1-5,8:00-17:00"
conf2
    time = "6-0,14:00-17:00"
conf3
    time = "22:00-6:00"
```

3. Use the **wlmcntrl -u** command to update WLM with the new configuration set. For example:

```
wlmcntrl -u /etc/wlm/MyConfigSet
```

At this point, WLM's current configuration is your new time-based configuration set.

You can also use the **confsetcntrl** and **lswlmconf** commands to create and manipulate configuration sets. For example:

To create the **confset1** configuration set with a default configuration of **conf1**, use the following command:

```
confsetcntrl -C confset1 conf1
```

To add **conf2** to **confset1** and make it the active configuration from 8:00 AM to 5:00 PM daily, use the following command:

```
confsetcntrl -d confset1 -a conf2 "0-6,08:00-17:00"
```

To make this configuration set the active configuration, use the following command:

```
wlmcntrl -d confset1
```

Creating a resource set

Using resource sets (rsets) is an effective way to isolate workloads from one another as far as the CPU is concerned. By separating two different workloads into two classes and giving each class a different subset of the CPUs, you can make sure that the two workloads never compete with each other for CPU resources, even though they still compete for physical memory and I/O bandwidth.

The simplest way to create a resource set is to use the SMIT interface (**smit addrsetcntrl** fast path) or the **mkrset** command.

For instructional purposes, the following example illustrates each step of creating and naming a resource set on a 4-way system. Its goal is to create a resource set containing processors 0 to 2, and use it in WLM configuration to restrict all processes of a superclass to these three processors.

1. With root authority, view the available building blocks (from which to create the resource sets) using the following command:

```
lsrset -av
```

The output for this example is the following:

T	Name	Owner	Group	Mode	CPU	Memory	Resources
r	sys/sys0	root	system	r-----	4	98298	sys/sys0
r	sys/node.00000	root	system	r-----	4	98298	sys/sys0
r	sys/mem.00000	root	system	r-----	0	98298	sys/mem.00000
r	sys/cpu.00003	root	system	r-----	1	0	sys/cpu.00003
r	sys/cpu.00002	root	system	r-----	1	0	sys/cpu.00002
r	sys/cpu.00001	root	system	r-----	1	0	sys/cpu.00001
r	sys/cpu.00000	root	system	r-----	1	0	sys/cpu.00000

In the output, **sys/sys0** represents the whole system (in this case, a 4-way SMP). When a WLM class does not specify an **rset** attribute, this is the default set that its processes potentially can access.

2. Create and name the resource set using the following SMIT fast path:

```
smit addrsetcntrl
```

For this example, fill in the fields as follows:

Name Space

admin

Resource Set Name

proc0_2

Resources

Select from the list those lines that correspond to the memory and CPUs 0 to 2 (sys/cpu.00000 to sys.cpu.00002).

All other fields

Select from the lists.

When you finish entering the fields and exit SMIT, the **admin/proc0_2** rset is created in **/etc/rsets**.

3. To use the new rset, add it into the kernel data structures using the following SMIT fast path:

```
smit reloadrsetcntrl
```

This menu gives you the option to reload the data base now, at next boot or both. Because this is the first time you are using the new resource set, select both so that this rset will be loaded now and after each reboot. (If you had changed an existing rset, you would probably have selected now.)

4. Add the new rset to a WLM class using the following SMIT fast path:

```
smit wlmclass_ga1
```

Select the class (in this example, **super1**) then select **admin/proc0_2** from the list available for the **Resource Set** field. After you make your selection and exit SMIT, the **classes** file on disk is changed.

5. Do one of the following:

- If WLM is running, update the configuration using the following SMIT fast path:

```
smit wlmupdate
```

- If WLM is not running, start it using the following SMIT fast path:

```
smit wlmstart
```

6. Monitor the effect of the new resource set on the class. For example:

a. Start 90 CPU loops (program executing an infinite loop) in class **super1**.

b. Type `wlmstat` on the command line. The output for this example is the following

```
          CLASS CPU MEM BIO
Unclassified  0  0  0
Unmanaged    0  0  0
Default      8  0  0
Shared       0  0  0
System       0  0  0
super1       75  0  0
super2       0  0  0
super2.Default  0  0  0
super2.Shared  0  0  0
super2.sub1   0  0  0
super2.sub2   0  0  0
```

This output shows that the 90 CPU bound processes, which otherwise unconstrained would take up 100% of the CPU, now use only 75% because the resource set limits them to run on CPUs 0 to 2.

c. To verify what resource set a process (identified by its PID) has access to, use the following SMIT fast path:

```
smit lsrsetproc
```

Enter the PID of the process you are interested in or select it from the list. The following output is for one of the loop processes:

```
  CPU Memory Resources
   3  98298 sys/mem.00000 sys/cpu.00002 sys/cpu.00001 sys/cpu.00000
```

Compare this with a process from a class without a specified **rset** attribute. (When no **rset** is specified for a class, it uses the **Default** resource set.) The following output is from the **init** process, which is in a class that does not specify a resource set:

```
  CPU Memory Resources
   4  98298 sys/sys0
```

At this point, your resource set exists and is being used by at least one class within WLM. For additional information see the **lsrset** command.

Note: WLM will not set its **rset** attachment for a process that currently has a **bindprocessor** subroutine binding or another **rset** attachment. When the other attachment no longer exists, WLM will assign its **rset** automatically.

Troubleshooting guidelines

If you are not seeing the desired behavior with your current configuration, you might need to adjust your WLM configuration.

The consumption values for each class can be monitored using tools such as **wlmstat**, **wlmmmon** or **wlmpperf**. This data can be collected and analyzed to help determine what changes might need to be made to the configuration. After you update the configuration, update the active WLM configuration using the **wlmcntrl -u** command.

The following guidelines can help you decide how to change your configuration:

- If the number of active shares in a tier varies greatly over time, you can give a class no shares for a resource so it can have a consumption target that is independent from the number of active shares. This technique is useful for important classes that require high-priority access to a resource.
- If you need to guarantee access to a certain amount of a resource, specify minimum limits. This technique is useful for interactive jobs that do not consume a lot of resources, but must respond quickly to external events.
- If you need to limit access to resources but shares do not provide enough control, specify maximum limits. In most cases, soft maximum limits are adequate, but hard maximums can be used for strict enforcement. Because hard maximum limits can result in wasted system resources, and they can increase paging activity when used for memory regulation, you should impose minimum limits for the other classes before imposing any hard limits.
- If less-important jobs are interfering with more-important jobs, put the less-important jobs in a lower tier. This technique ensures less-important jobs have lower priority and cannot compete for available resources while the more-important jobs are running.
- If a class cannot reach its consumption target for a resource, check whether this condition is caused by contention for another resource. If so, change the class allocation for the resource under contention.
- If processes within a class vary greatly in their behaviors or resource consumption, create more classes to gain more granular control. Also, it might be desirable to create a separate class for each important application.
- If your analysis shows the resource required by one class is dependent on the consumption of another class, reallocate your resources accordingly. For example, if the amount of resource required by ClassZ is dependent on the number of work requests that can be handled by ClassA, then ClassA must be guaranteed access to enough resources to provide what ClassZ needs.
- If one or more applications are consistently not receiving enough resources to perform adequately, your only option might be to reduce the workload on the system.

Note: You can define an *adminuser* for a superclass to reduce the amount of work that is required of the WLM administrator. After the top-level configuration has been tested and tuned, subsequent changes (including creating and configuring subclasses) can be made by the superclass adminusers to suit their particular needs.

Configuring Workload Manager to consolidate workloads

Workload Manager (WLM) gives you control over the resources used by jobs on your system.

A default WLM configuration template exists on every installed AIX operating system. The following procedure updates the WLM configuration template to implement a resource-management policy on a shared server. The resulting configuration can be used as a starting point for testing. Exactly how you configure WLM will depend on the workload and policy requirements for your environment.

Note:

1. Efficient use of WLM requires extensive knowledge of existing system processes and performance. Repeated testing and tuning will probably be needed before you can develop a configuration that works well for your workload. If you configure WLM with extreme or inaccurate values, you can significantly degrade system performance.
2. The process of configuring WLM is simpler when you already know one or more of the classification attributes of a process (for example, user, group, or application name). If you are

unfamiliar with the current use of resources, use a tool such as **topas** to identify the processes that are the primary resource users and use the resulting information as the starting point for defining classes and rules.

3. The following scenario assumes you are familiar with the basic Workload Manager concepts as described in “Workload management concepts” on page 195.

The WLM configuration files exist in the `/etc/wlm/ConfigurationName` directory. Each subclass for each superclass is defined in a configuration file named `/etc/wlm/ConfigurationName/SuperClassName`. For more information about these files, see the *AIX 5L Version 5.3 Files Reference*.

In the following procedure, you consolidate the workloads from two separate department servers onto one larger server. This example edits the configuration files, but you can also create a configuration using SMIT (use the **smit wlmconfig_create** fast path) or Web-based System Manager (select the **Workload Manager** container, select the **Configuration/Classes** container, then from the **Workload** menu, select **New Configuration**). Briefly, in this procedure, you will do the following:

1. Identify the resource requirements of the applications you want to consolidate. This will help you determine how many applications you can move to the larger server.
2. Define tiers, as well as resource shares and limits, to begin testing with the consolidated workload.
3. Fine-tune the configuration until you achieve your desired results.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

Step 1. Identify application requirements

In this scenario, the workload is typical of what you might see on a database server. Assume the jobs fall into the following general categories:

Listeners

These are processes that sleep most of the time and wake up periodically in response to a request. Although these processes do not consume a lot of resources, response time can be critical.

Workers

These are processes that do the work on behalf of a request, whether the request is local or remote. These processes probably use a lot of CPU time and memory.

Reporters

These are processes that do automated tasks. They might require a lot of CPU time or memory, but they can tolerate a slower response time.

Monitors

These are processes that typically run periodically to verify the state of the system or applications. These processes might use a significant amount of resource, but only for a short time.

Commands

These are commands or other applications that system users might run at any time. Their resource requirements are unpredictable.

In addition to this work, scheduled jobs fall into one of the following categories:

SysTools

These are processes that perform automated tasks. These jobs are not critical to system operation but need to run periodically and within certain time constraints.

SysBatch

These are processes that run infrequently, are not critical to system operation, and need not finish in a timely manner.

The first step of creating a configuration is to define classes and rules. In the following steps, you will use the general job categories listed above to define your classes. Use the following procedure:

1. Make a new configuration within the **/etc/wlm** directory called **MyConfig** using the following command:

```
mkdir /etc/wlm/MyConfig
```

2. Copy the template files into the **/etc/wlm/MyConfig** directory using the following command:

```
cp -pr /etc/wlm/template/* /etc/wlm/MyConfig
```

3. To create the superclasses, use your favorite editor to modify the **/etc/wlm/MyConfig/classes** file to contain the following:

```
System:
```

```
Default:
```

```
DeptA:
```

```
DeptB:
```

```
SysTools:
```

```
SysBatch:
```

As a starting point, you define one superclass for each department (because two departments will be sharing the server). The SysTool and SysBatch superclasses will handle the scheduled jobs outlined in the general categories above. The System and Default superclasses are always defined.

4. Within the **MyConfig** directory, create a directory for each the DeptA and DeptB superclasses. (When creating a configuration, you must create a directory for every superclass that has subclasses.) In the following step, you define five subclasses (one for each category of work) for each department's superclass.
5. To create subclasses for each general category of jobs, edit the **/etc/wlm/MyConfig/DeptA/classes** and **/etc/wlm/MyConfig/DeptB/classes** files to contain the following:

```
Listen:
```

```
Work:
```

```
Monitor:
```

```
Report:
```

```
Command:
```

Note: The contents of the **classes** file can be different for each superclass.

After the classes are identified, in the following step, you create the classification rules that are used to classify processes at the superclass and subclass levels. For the sake of simplicity, assume that all applications run from known locations, that all processes from one department run under the deptAUNIX group, and that processes from the other department run under the deptB UNIX group.

6. To create the superclass assignment rules, modify the **/etc/wlm/MyConfig/rules** file to contain the following:

```
DeptA - - deptA - -  
DeptB - - deptB - -  
SysTools - root,bin - /usr/sbin/tools/* -  
SysBatch - root,bin - /usr/sbin/batch/* -  
System - root - - -  
Default - - - - -
```

Note: If more than one instance of the same application can be running and all classification attributes (other than the tag) are the same, use the **wlmassign** command or **wlm_set_tag** subroutine to differentiate between them by assigning them to different classes.

7. To create more specific subclass rules, create the `/etc/wlm/MyConfig/DeptA/rules` and `/etc/wlm/MyConfig/DeptB/rules` files with the following content:

```
Listen - - - /opt/myapp/bin/listen* -  
Work - - - /opt/myapp/bin/work* -  
Monitor - - - /opt/bin/myapp/bin/monitor -  
Report - - - /opt/bin/myapp/report* -  
Command - - - /opt/commands/* -
```

8. To determine the resource-consumption behavior of each class, start WLM in passive mode using the following command:

```
wlmcntrl -p -d MyConfig
```

After starting WLM in passive mode, you can run each application separately at first to gain a finer perspective of its resource requirements. You can then run all applications simultaneously to better determine the interaction among all classes.

An alternative method of identifying the application resource requirements might be to first run WLM in passive mode on the separate servers from which you are consolidating applications. The disadvantages to this approach are that you would have to re-create the configurations on the larger system, and the required percentage of resources will likely be different on the larger system.

Step 2. Define Tiers, Shares, and Limits

A WLM configuration is an implementation of a resource-management policy. Running WLM in passive mode provides information that helps you determine whether your resource-management policy is reasonable for the given workload. You can now define tiers, shares, and limits to regulate your workload based on your resource-management policy.

For this scenario, assume you have the following requirements:

- The System class must have the highest priority and be guaranteed access to a percentage of system resources at all times.
- The SysTools class must have access to a certain percentage of resources at all times, but not so much that it will significantly impact the applications that are running in DeptA and DeptB.
- The SysBatch class cannot interfere with any of the other work on the system.
- DeptA will receive 60% of the available resources (meaning resources that are available to the classes with shares) and DeptB will receive 40%. Within DeptA and DeptB:
 - Processes in the Listen class must respond to requests with a low latency, but must not consume a lot of resources.
 - The Work class must be allowed to consume the most resources. The Monitor and Command classes must consume some resource, but less than the Work class.
 - The Report class cannot interfere with any of the other work.

In the following procedure, you define tiers, shares, and limits:

1. To create the superclass tiers, use your favorite editor to modify the `/etc/wlm/MyConfig/classes` file to contain the following:

```
System:
```

```
Default:
```

```
DeptA:
```

```
localshm = yes  
adminuser = adminA  
authuser = adminA  
inheritance = yes
```

```
DeptB:
```

```
localshm = yes
```



```
adminuser = adminB
authuser = adminB
inheritance = yes
```

```
SysTools:
    localshm = yes
```

```
SysBatch:
    tier = 1
    localshm = yes
```

The SysBatch superclass is put in tier 1 because this class contains very low-priority jobs that you do not want to interfere with the rest of the work on the system. (When a tier is not specified, the class defaults to tier 0.) Administration of each department's superclass is defined by the adminuser and authuser attributes. The inheritance attribute is enabled for DeptA and DeptB. All new processes started in a class with inheritance will remain classified in that class.

2. To create subclass tiers for each group of jobs, modify the **/etc/wlm/MyConfig/DeptA/classes** and **/etc/wlm/MyConfig/DeptB/classes** files to contain the following:

```
Listen:
```

```
Work:
```

```
Monitor:
```

```
Report:
    tier = 1
```

```
Command:
```

3. To assign the initial shares for the superclasses, edit the **/etc/wlm/MyConfig/shares** file to contain the following:

```
DeptA:
    CPU = 3
    memory = 3
```

```
DeptB:
    CPU = 2
    memory = 2
```

Because you assigned a CPU total of 5 shares, DeptA processes will have access to three out of five shares (or 60%) of the total CPU resources and DeptB processes will have access to two out of five (or 40%). Because you did not assign shares to the SysTools, System, and Default classes, their consumption targets will remain independent from the number of active shares, which gives them higher-priority access to resources than the DeptA and DeptB (until their limit is reached). You did not assign the SysBatch class any shares because it is the only superclass in tier 1, and therefore any share assignment is irrelevant. Jobs in the SysBatch class can only consume resources that are unused by all classes in tier 0.

4. To assign the initial shares for the subclasses, edit the **/etc/wlm/MyConfig/DeptA/shares** and **/etc/wlm/MyConfig/DeptB/shares** files to contain the following:

```
Work:
    CPU = 5
    memory = 5
```

```
Monitor:
    CPU = 4
    memory = 1
```

```
Command:
    CPU = 1
    memory = 1
```

Because you did not assign shares to the Listen class, it will have the highest-priority access (in the superclass) to resources when it requires them. You assigned the largest number of shares to the Work class because it has the greatest resource requirements. Accordingly, you assigned shares to the Monitor and Command classes based on their observed behavior and relative importance. You did not assign shares to the Report class because it is the only subclass in tier 1, and therefore any share assignment is irrelevant. Jobs in the Report class can only consume resources that are unused by subclasses in tier 0.

In the following step of this example, you assign limits to classes that were not assigned shares. (You can also assign limits to classes with shares. See *Managing Resources with WLM* for more information.)

5. To assign limits to the superclasses, edit the `/etc/wlm/MyConfig/limits` file to contain the following:

Default:

```
CPU = 0%-10%;100%
memory = 0%-10%;100%
```

SysTools:

```
CPU = 0%-10%;100%
memory = 0%-5%;100%
```

System:

```
CPU = 5%-50%;100%
memory = 5%-50%;100%
```

You assigned soft maximum limits to the System, SysTools, and Default classes to prevent them from significantly interfering with other work on the system. You assigned minimum limits to the System class for CPU and memory because this class contains processes that are essential to system operation, and it must be able to consume a guaranteed amount of resource.

6. To assign limits to the subclasses, edit the `/etc/wlm/MyConfig/DeptA/limits` and `/etc/wlm/MyConfig/DeptB/limits` files to contain the following:

Listen:

```
CPU = 10%-30%;100%
memory = 10%-20%;100%
```

Monitor:

```
CPU = 0%-30%;100%
memory = 0%-30%;100%
```

Note: The limits can be different for each subclass file.

You assigned soft maximum limits to the Listen and Monitor classes to prevent them from significantly interfering with the other subclasses in the same superclass. In particular, you do not want the system to continue accepting requests for jobs within the Work class, if the Work class does not have access to the resources to process them. You also assigned minimum limits to the Listen class to ensure fast response time. The minimum limit for memory ensures that pages used by this class will not be stolen by page replacement, resulting in faster execution time. The minimum limit for CPU ensures that when these processes can be run, they will have the highest-priority access (in the superclass) to the CPU resources.

Step 3. Fine-tune the Workload Manager configuration

1. Monitor the system using the `wlmstat` command and verify that the regulation done by WLM aligns with your goals and does not unduly deprive some applications of resources while others get more than they should. If this is the case, adjust the shares and refresh WLM.
2. As you monitor and adjust the shares, limits, and tier numbers, decide whether you want to delegate the administration for the subclasses for some or all of the superclasses. The administrator can then monitor and set up the subclass shares, limits, and tier number.

The administrator of each superclass can repeat this process for the subclasses of each superclass. The only difference is that WLM cannot run in passive mode at the subclass level only. The subclass

configuration and tuning has to be done with WLM in active mode. One way not to impact users and applications in the superclass is to start the tier number, and the shares and limits for the subclasses at their default value ('-' (hyphen) for shares, 0% for minimum, and 100% for soft and hard maximum). With these settings, WLM will not regulate the resource allocation between the subclasses.

For more information

- Workload Manager.
- Workload Management.
- Workload Management Diagnosis in *Performance management*.
- Analyzing WLM with wlmperf in *Performance Toolbox Version 2 and 3 for AIX: Guide and Reference*.
- The descriptions for the **classes**, **limits**, **rules**, and **shares** files in the *AIX 5L Version 5.3 Files Reference*.
- The **topas**, **wlmassign**, **wlmcheck**, **wlmcntrl**, **wlmmmon**, **wlmperf**, and **wlmstat**.
- The WLM subroutine descriptions, especially **wlm_set_tag**.
- *AIX 5L Workload Manager (WLM)*, an IBM Redbook
- *Server Consolidation on RS/6000®*, an IBM Redbook

Classes

Workload Manager helps you control the allocation of system resources by defining classes of service and allocating resources to each of these classes.

Each class has a set of attributes that determine what its resource entitlements are, as well as other behaviors. Every process on the system is classified into a service class, and is thus subject to enforcement of the resource entitlements and behaviors for that class. Processes are assigned to a class either manually using manual assignment, or automatically according to user-defined classification rules.

WLM supports two levels of classes: *superclasses* and *subclasses*. Superclasses are given resource entitlements based on available system resources, and subclasses are given resource entitlements relative to the entitlements of their associated superclass. Optionally, you can define subclasses to allow for more granular control of the processes in a superclass. You can also delegate the responsibility of defining subclasses by specifying an adminuser or admingroup for a superclass.

For both the superclass and subclass levels, you can define classes, resource shares and limits, and rules using SMIT, Web-based System Manager, or the command-line interface. Applications can use the WLM APIs. Configuration definitions are kept in a set of text files called the WLM *property files*.

A class name is up to 16 characters in length and can contain only uppercase and lowercase letters, numbers and underscores (_). For a given WLM configuration, each superclass name must be unique. Each subclass name must be unique within that superclasses, but it can match subclass names in other superclasses. To uniquely identify every subclass, the full name of a subclass is composed of the superclass name and the subclass name separated by a dot; for example: *Super.Sub*.

Superclasses

The system administrator can define up to 64 superclasses.

In addition, the following five superclasses are automatically created:

Default superclass

Is the default superclass and is always defined. All non-root processes that are not automatically assigned to a specific superclass are assigned to the Default superclass. Other processes can also be assigned to the *Default* superclass by providing specific assignment rules.

System superclass

Has all privileged (root) processes assigned to it if those processes are not assigned by rules to a specific class. This superclass also collects the memory pages belonging to kernel memory segments and kernel processes. Other processes can also be assigned to the System superclass by providing specific assignment rules for this superclass. This superclass has a memory minimum limit of 1% as the default.

Shared superclass

Receives the memory pages that are shared by processes in more than one superclass. This includes pages in shared memory regions and pages in files that are used by processes in more than one superclass (or in subclasses of different superclasses). Shared memory and files that are used by multiple processes that all belong to a single superclass (or subclasses of the same superclass) are associated with that superclass. Only when a process from a different superclass accesses the shared memory region or file are the pages placed in the Shared superclass. This superclass can have only physical memory shares and limits applied to it. It cannot have shares or limits for the other resource types, subclasses, or assignment rules specified. Whether a memory segment shared by processes in different subclasses of the same superclass is classified into the *Shared* subclass or remains in its original subclass depends on the value of the **localshm** attribute of the original subclass.

Unclassified superclass

Is a memory allocation for unclassified processes. The processes in existence at the time that WLM is started are classified according to the assignment rules of the WLM configuration being loaded. During this initial classification, all the memory pages attached to each process are "charged" either to the superclass that the process belongs to (when not shared, or shared by processes in the same superclass), or to the *Shared* superclass when shared by processes in different superclasses.

However, a few pages cannot be directly tied to any processes (and thus to any class) at the time of this classification, and this memory is charged to the *Unclassified* superclass. Most of this memory is correctly reclassified over time, when it is either accessed by a process, or released and reallocated to a process after WLM is started. There are no processes in the *Unclassified* superclass. This superclass can have physical memory shares and limits applied to it. It cannot have shares or limits for the other resource types, subclasses, or assignment rules specified.

Unmanaged superclass

A special superclass, named *Unmanaged*, is always defined. No processes are assigned to this class. This class accumulates the memory usage for all pinned pages in the system that are not managed by WLM. The CPU utilization for the waitproc processes is not accumulated in any class to prevent the system from appearing to be at 100% utilization. This superclass cannot have shares or limits for any resource types, subclasses, or specified assignment rules.

Subclasses

The system administrator or a superclass administrator can define up to 61 subclasses.

In addition, two special subclasses, *Default* and *Shared*, are always defined.

Default subclass

Is the default subclass and is always defined. All processes that are not automatically assigned to a specific subclass of the superclass are assigned to the *Default* subclass. You can also assign other processes to the *Default* subclass by providing specific assignment rules.

Shared subclass

Receives all the memory pages that are used by processes in more than one subclass of the superclass. Included are pages in shared memory regions and pages in files that are used by processes in more than one subclass of the same superclass. Shared memory and files that are used by multiple processes that all belong to a single subclass are associated with that subclass. Only when a process from a different subclass of the same superclass accesses the shared

memory region or file are the pages placed in the *Shared* subclass of the superclass. There are no processes in the *Shared* subclass. This subclass can have only physical memory shares and limits applied to it, and it cannot have shares or limits for the other resource types or assignment rules specified. Whether a memory segment shared by processes in different subclasses of the same superclass is classified into the *Shared* subclass or remains in its original subclass depends on the value of the **localshm** attribute of the original subclass.

Class attributes

This describes the attributes of a WLM class.

Class Name

Can be up to 16 characters in length and can only contain uppercase and lowercase letters, numbers and underscores (_).

Tier A number between 0 and 9 used to prioritize resource allocation between classes.

Inheritance

Specifies whether a child process inherits the class assignment from its parent.

localshm

Prevents memory segments belonging to one class from migrating to Shared class.

Administrator (adminuser, admingroup, authgroup) (superclass only)

Delegates the administration of a superclass.

Authorization (authuser, authgroup)

Delegates the right to manually assign a process to a class.

Resource Set (rset)

Limits the set of resources a given class has access to in terms of CPUs (processor set).

delshm

Deletes the shared memory segments if the last referencing process is killed due to the virtual memory limit.

vmenforce

Indicates whether to kill all processes in a class, or only the offending process, when a class reaches its virtual memory limit.

Tier attribute

Tiers represent the order in which system resources are allocated to WLM classes.

The administrator can define classes in up to 10 tiers, numbered 0 through 9, with 0 being the highest or most important tier. The amount of resources available to tier 0 is all available system resources. The amount of resources available to the lower (higher number) tiers, is the amount of resources that is unused by all higher tiers. Target consumption percentages for classes are based on the number of active shares in its tier, and the amount of resource available to the tier. Because tier 0 is the only tier that is guaranteed to always have resources available to it, it is recommended that processes that are essential to system operation be classified in a class in this tier. If no tier value is specified for a class, it will be put in tier 0.

A tier can be specified at both the superclass and the subclass levels. Superclass tiers are used to specify resource allocation priority between superclasses. Subclass tiers are used to specify resource allocation priority between subclasses of the same superclass. There is no relationship among sub-tiers of different superclasses.

Inheritance attribute

The **inheritance** attribute of a class indicates whether processes in the class should be automatically reclassified when one of the classification attributes of the process changes.

When a new process is created with the **fork** subroutine, it automatically inherits its parent's class, whether or not inheritance is enabled. One exception is when the parent process has a tag, has its **inherit tag at fork** set to off, and class inheritance is off for the parent's class. In this case, the child process is reclassified according to the classification rules.

When inheritance is not enabled for a class, any process in the class is automatically classified according to the classification rules after calling any service that changes a process attribute that is used in the rule. The most common of these calls is the **exec** subroutine, but other subroutines that can change classification include **setuid**, **setgid**, **plock**, **setpri**, and **wlm_set_tag**. When inheritance is enabled, the process is not subject to reclassification based on the classification rules, and will remain in its current class. Manual assignment has priority over inheritance and can be used to reclassify processes that are in a class with inheritance enabled.

The specified value for the **inheritance** attribute can be either yes or no. If unspecified, inheritance will not be enabled for a class.

This attribute can be specified at both superclass and subclass level. For a subclass of a given superclass:

- If the **inheritance** attribute is set to yes at both the superclass and the subclass levels, a child of a process in the subclass will remain in the same subclass.
- If the **inheritance** attribute is set to yes for the superclass and no (or unspecified) for the subclass, a child of a process in the subclass will remain in the same superclass and will be classified in one of its subclasses according to the assignment rules for the superclass.
- If the **inheritance** attribute is no (or is unspecified) for the superclass and is set to yes for the subclass, a child of a process in the subclass will be submitted to the automatic assignment rules for the superclasses.
 - If the process is classified by the rules in the same superclass, then it will remain in the subclass (it will not be submitted to the subclass's assignment rules).
 - If the process is classified by the superclass's rules in a different superclass, then the subclass assignment rules of the new superclass are applied to determine the subclass of the new superclass the process will be assigned to.
- If both superclass and subclass **inheritance** attributes are set to no (or are unspecified), then a child of a process in the subclass will be submitted to the standard automatic assignment.

localshm attribute

The **localshm** attribute can be specified at the superclass and the subclass levels.

The **localshm** attribute is used to prevent memory segments belonging to one class from migrating to the *Shared* superclass or subclass when accessed by processes in other classes. The possible values for the attribute are yes or no. A value of yes means that shared memory segments in this class must remain local to the class and not migrate to the appropriate *Shared* class. A value of no is the default when the attribute is not specified.

Memory segments are classified on page faults. When a segment is created, it is marked as belonging to the *Unclassified* superclass. On the first page fault on the segment, this segment is classified into the same class as the faulting process. If, later on, a process belonging to a different class than the segment page faults on this segment, WLM considers whether the segment needs to be reclassified into the appropriate *Shared* class (superclass or subclass). If the faulting process and the segment belong to different superclasses, one of the following occurs:

- If the segment's superclass has the **localshm** attribute set to yes, the segment remains in its current superclass. If the segment's subclass has the **localshm** attribute set to yes, the segment remains in its current subclass. If the superclass **localshm** attribute is set to yes but its subclass attribute is set to no, it goes into the *Shared* subclass of the current superclass.

- If the segment's superclass has the **localshm** attribute set to no, the segment goes to *Shared* superclass. This is the default action.

If the faulting process and the segment belong to different subclasses of the same superclass, and the segment's subclass has the **localshm** attribute set to yes, the segment remains in the current class (superclass and subclass). Otherwise, the segment goes to the *Shared* subclass of the superclass.

Of course, if the faulting process and the segment belong to the same class (same superclass and same subclass), the segment is not reclassified regardless of the values of the **localshm** attributes.

Administrator attribute

The **adminuser** and **admingroup** attributes are used to delegate the superclass administration to a user or group of users.

Note: These attributes are valid only for superclasses.

The **adminuser** attribute specifies the name of the user (as listed in */etc/passwd*) authorized to perform administration tasks on the superclass. The **admingroup** attribute specifies the name of the group of users (as listed in */etc/group*) authorized to perform administration tasks on the superclass.

Only one value (user or group) is allowed for each attribute. Either of them, none, or both can be specified. The user or group will have authority to do the following:

- Create and delete subclasses.
- Change the attributes and resource shares and limits for the subclasses.
- Define, remove or modify subclass assignment rules.
- Refresh (update) the active WLM configuration for the superclass.

Authorization attribute

The **authuser** and **authgroup** attributes are valid for all classes. They are used to specify the user or group authorized to manually assign processes to the class (superclass or subclass).

When manually assigning a process (or a group of processes) to a superclass, the assignment rules for the superclass are used to determine to which subclass of the superclass each process will be assigned.

Only one value (user or group) is allowed for each attribute. Either of them, none, or both can be specified.

Resource set attribute

The resource set attribute (called *rset*) can be specified for any class. Its value is the name of a resource set defined by the system administrator.

The *rset* attribute represents a subset of the CPU resource available on the system (processor set). The default is "system," which gives access to all the CPU resources available on the system. The only restriction is that if an *rset* is specified for a subclass, the set of CPUs in the the set must be a subset of the CPUs available to the superclass. (For detailed information, see the **mkrset** command.)

Note: Carefully consider assigning resource sets to any class that is not in tier 0. Because lower tiers only have access to the resources that are unused by the higher tiers, restricting a non-tier-0 class to a subset of the CPUs on the system could result in starvation if there is no CPU time available on those CPUs.

Process classifications in Workload Manager

In WLM, processes can be classified in either of two ways.

- A process is automatically assigned using assignment rules when process classification attributes change. When WLM is running in active mode, this automatic assignment is always in effect (it cannot be turned off). This is the most common way that processes are classified.
- A selected process or group of processes can be manually assigned to a class by a user with the required authority on both the processes and the target class. Manual assignment can be done using by a WLM command, which could be invoked directly or through SMIT or Web-based System Manager, or by an application using a function of the WLM Application Programming Interface. This manual assignment overrides the automatic assignment and inheritance.

Automatic class assignment in Workload Manager

The automatic assignment of processes to classes uses a set of class-assignment rules specified by a WLM administrator.

There are two levels of assignment rules:

- A set of assignment rules at the WLM configuration level used to determine which superclass a given process is assigned to.
- Each superclass with subclasses defined, in turn has a set of assignment rules used to determine which subclass of the superclass the process is assigned to.

The assignment rules at both levels are based on the values of a set of process attributes. These attributes are as follows:

- Process user ID
- Process group ID
- Path name of the application (program) executed
- Type of the process (32bit or 64bit, for example)
- Process tag.

The tag is a process attribute, defined as a character string, that an application can set by program, using the WLM API.

The classification is done whenever an attribute changes by comparing the value of these process attributes against lists of possible values given in the class assignment rules file (called **rules**). The comparison determines which rule is a match for the current value of the process attributes.

A class assignment rule is a text string that includes the following fields, separated by one or more spaces:

Name	Must contain the name of a class which is defined in the class file corresponding to the level of the rules file (superclass or subclass). Class names can contain only uppercase and lowercase letters, numbers, and underscores and can be up to 16 characters in length. No assignment rule can be specified for the system defined classes Unclassified, Unmanaged and Shared.
Reserved	Reserved for future extension. Its value must be a hyphen (-), and it must be present.
User	Can contain either a hyphen (-) or at least one valid user name (as defined in the /etc/passwd file). The list is composed of one or more names, separated by a comma (.). An exclamation mark (!) can be used before a name to exclude a given user from the class. Patterns can be specified to match a set of user names, using full Korn shell pattern-matching syntax. If there are no valid user names, the rule is ignored.
Group	Can contain either a hyphen (-) or at least one valid group name (as defined in the /etc/group file). The list is composed of one or more names, separated by a comma (.). An exclamation mark (!) can be used before a name to exclude a given group from the class. Patterns can be specified to match a set of group names using full Korn shell pattern matching syntax. If there are no valid group names, the rule is ignored.

Application	<p>Can contain either a hyphen (-) or a list of application path names. This is the path name of the applications (programs) executed by processes included in the class. The application names will be either full path names or Korn shell patterns that match path names. The list is composed of one or more path names, separated by a comma (.). An exclamation mark (!) can be used before a name to exclude a given application.</p> <p>At least one application in the list must be found at load time or the rule is ignored. Rules that are initially ignored for this reason might become effective later on if a file system is mounted that contains one or more applications in the list.</p>
Type	<p>Can contain either a hyphen (-) or a list of process attributes. The possible values for these attributes are:</p> <ul style="list-style-type: none"> • 32bit: the process is a 32-bit process • 64bit: the process is a 64-bit process • plock: the process called the plock subroutine to pin memory • fixed: the process is a fixed priority process (SCHED_FIFO or SCHED_RR) <p>The fixed type is for classification purposes only. WLM does not regulate the processor use of fixed priority processes or threads. Because fixed priority processes have the potential to cause deprivation among other processes in a class, this classification attribute is provided to allow isolation of these jobs. This attribute can also be used to report consumption of such processes.</p> <p>The value of the type field can be a combination of one or more of the above attributes separated by a plus (+). The 32bit and 64bit values are mutually exclusive.</p>
Tag	<p>May contain either a hyphen (-) or a list of application tags. An application tag is a string of up to 30 alphanumeric characters. The list is composed of one or more application tag values separated by commas.</p>

The User, Group, Application, and Tag attributes can be an attribute value grouping.

When a process is created (fork), it remains in the same class as its parent. Reclassification happens when the new process issues a system call which can modify one of the attributes of the process used for classification; for example, *exec*, *setuid* (and related calls), *setgid* (and related calls), *setpri* and *plock*.

To classify the process, WLM examines the top-level **rules** file for the active configuration to determine which superclass the process belongs. For each rule in the file, WLM checks the current values for the process attributes against the values and lists of values specified in the rule. Rules are checked in the order that they appear in the file. When a match is found, the process is assigned to the superclass named in the first field of the rule. Then the rules file for the superclass is examined in the same way to determine to which subclass the process should be assigned.

For a process to match one of the rules, each of its attributes must match the corresponding field in the rule. The following is a list of the criteria used to determine whether the value of an attribute matches the values in the field of the **rules** file:

- If the field in the rules file has a value of hyphen (-), then any value of the corresponding process attribute is a match.
- For all the attributes except *type*, if the value of the process attribute matches one of the values in the list in the rules file that is not excluded (prefaced by a "!"), then a match has occurred.
- For the *type* attribute, if one of the values in the rule is comprised of two or more values separated by a plus (+), then a process is a match only if its characteristics match all the values.

At both superclass and subclass levels, WLM goes through the rules in the order in which they appear in the **rules** file, and classifies the process in the class corresponding to the first rule for which the process is a match. The order of the rules in the rules file is therefore extremely important. Use caution when you create or modify the rules file.

Manual class assignment in Workload Manager

A process or a group of processes can be manually assigned to a superclass and/or subclass by using Web-based System Manager, SMIT, or the **wlmassign** command.

See **wlmassign** for more information. An application can assign processes through the **wlm_assign** API function.

To manually assign processes to a class or to cancel an existing manual assignment, a user must have the appropriate level of privilege. (See “Security considerations for Workload Manager” on page 221 for further details.) A manual assignment can be made or canceled separately at the superclass level, the subclass level, or both. This assignment is specified by flags for the programming interface and a set of options for the command line interface used by the WLM administration tools. So, a process can be manually assigned to a superclass only, a subclass only, or to a superclass and a subclass of that superclass. In the latter case, the dual assignment can be done simultaneously (with a single command or API call) or at different times, by different users.

Assignment is very flexible, but can be confusing. Following are two examples of the possible cases.

Example 1: First Assignment of Processes

A system administrator manually assigns *Process1* from *superclassA* to *superclassB* (superclass-level-only assignment). The automatic assignment rules for the subclasses of *superclassB* are used by WLM to determine to which subclass the process is ultimately assigned. *Process1* is assigned to *superclassB.subclassA* and is flagged as having a “superclass only” assignment.

A user with the appropriate privileges assigns *Process2* from its current class *superclassA.subclassA* to a new subclass of the same superclass, *superclassA.subclassB*. *Process2* is assigned to its new subclass and flagged as having a “subclass only” assignment.

A WLM administrator of the subclasses of *superclassB* manually reassigns *Process1* to *subclassC*, which is another subclass of *superclassB*. *Process1* is reclassified into *superclassB.subclassC* and is now flagged as having both superclass and subclass level assignment.

Example 2: Reassignment or Cancellation of Manual Assignment

The reassignment and cancellation of a manual assignment at the subclass level is less complex and affects only the subclass level assignment.

Suppose that the system administrator wants *Process2* to be in a superclass with more resources and decides to manually assign *Process2* to *superclassC*. In Example 1, *Process2* was manually assigned to *subclassB* of *superclassA*, with a “subclass only” assignment. Because *Process2* is assigned to a different superclass, the previous manual assignment becomes meaningless and is canceled. *Process2* now has a “superclass only” manual assignment to *superclassC*, and in the absence of inheritance, is assigned to a subclass of *superclassC* using the automatic assignment rules.

Now, the system administrator decides to terminate the manual assignment from *Process1* to *superclassB*. The “superclass level” manual assignment of *Process1* is canceled, and in the absence of inheritance, *Process1* is assigned a superclass using the top level automatic assignment rules.

If the rules have not changed, *Process1* is assigned to *superclassA*, and its subclass level manual assignment to *superclassB.subclassC* becomes meaningless and is canceled.

If for some reason the top level rules assign *Process1* to *superclassB*, then the subclass level assignment to *superclassB.subclassC* is still valid and remains in effect. *Process1* now has a “subclass only” manual assignment.

Updates to the Workload Manager

When WLM is updated (with the **wlmctrl -u** command), the updated configuration can load a new set of classification rules.

When this happens, processes are often reclassified using the new rules. WLM does not reclassify processes that have been manually assigned or that are in a class with inheritance enabled, unless their class does not exist in the new configuration.

Security considerations for Workload Manager

To assign a process to a class or to cancel a prior manual assignment, the user must have authority both on the process and on the target class.

These constraints translate into the following rules:

- The root user can assign any process to any class.
- A user with administration privileges on the subclasses of a given superclass (that is, the user or group name matches the user or group names specified in the attributes **adminuser** and **admingroup** of the superclass) can manually reassign any process from one of the subclasses of this superclass to another subclass of the superclass.
- Users can manually assign their own processes (ones associated with the same real or effective user ID) to a subclass for which they have manual assignment privileges (that is, the user or group name matches the user or group names specified in the attributes **authuser** and **authgroup** of the superclass or subclass).

To modify or terminate a manual assignment, users must have at least the same level of privilege as the person who issued the last manual assignment.

Resource management with Workload Manager

WLM monitors and regulates the resource utilization, on a per-class basis, of the threads and processes active on the system. You can set minimum or maximum limits per class for each resource type managed by WLM, as well as a target value per class for each resource. This target is representative of the amount of the resource that is optimal for the jobs in the class.

The shares and limits at the superclass level refer to the total amount of each resource available on the system. At the subclass level, shares and limits refer to the amount of each resource made available to the superclass that the subclass is in (superclass target). The hierarchy of classes is a way to divide the system resources between groups of users (superclasses) and delegate the administration of this share of the resources to superclass administrators. Each superclass administrator can then redistribute this amount of resources between the users in the group by creating subclasses and defining resource entitlements for these subclasses.

Resource types in Workload Manager

WLM manages three types of resources on a percentage consumption basis.

CPU utilization of the threads in a class	This is the sum of all the CPU cycles consumed by every thread in the class.
Physical memory utilization for the processes in a class	This is the sum of all the memory pages which belong to the processes in the class.
Disk I/O bandwidth for the class	This is the bandwidth (in 512-byte blocks per second) of all the I/Os started by threads in the class on each disk device accessed by the class.

Every second, WLM calculates the per-class utilization for each resource during the last second, as a percentage of the total resource available, as follows:

- For the CPU, the total amount of CPU time available every second is equal to 1 second times the number of CPUs on the system. For instance, on an eight-way SMP, if all the threads of a class combined consumed 2 seconds of CPU time during the last second, this represents a percentage of $2/8 = 25\%$. The percentage used by WLM for regulation is a decayed average over a few seconds of this "instantaneous" per-second resource utilization.
- For physical memory, the total amount of physical memory available for processes at any given time is equal to the total number of memory pages physically present on the system minus the number of pinned pages. Pinned pages are not managed by WLM because these pages cannot be stolen from a class and given to another class to regulate memory utilization. The memory utilization of a class is the ratio of the number of non-pinned memory pages owned by all the processes in the class to the number of pages available on the system, expressed as a percentage.
- For disk I/O, the main problem is determining a meaningful available bandwidth for a device. When a disk is 100% busy, its throughput, in blocks per second, is very different if one application is doing sequential I/Os than if several applications are generating random I/Os. If you only used the maximum throughput measured for the sequential I/O case (as a value of the I/O bandwidth available for the device) to compute the percentage of utilization of the device under random I/Os, you might be misled to think that the device is at 20% utilization, when it is in fact at 100% utilization.

To get more accurate and reliable percentages of per-class disk utilization, WLM uses the statistics provided by the disk drivers (displayed using the **iostat** command) giving for each disk device the percentage of time that the device has been busy during the last second. WLM counts the number of total blocks which have been read or written on a device during the last second by all the classes accessing the device, and how many blocks have been read or written by each class and what was the percentage of utilization of the device. WLM then calculates the percentage of the disk throughput consumed by each class.

For instance, if the total number of blocks read or written during the last second was 1000 and the device had been 70% busy, this means that a class reading or writing 100 blocks used 7% of the disk bandwidth. Similarly to the CPU time (another renewable resource), the values used by WLM for its disk I/O regulation are also a decayed average over a few seconds of these per second percentages.

For the disk I/O resource, the shares and limits apply to each disk device individually accessed by the class. The regulation is done independently for each device. This means that a class could be over its entitlement on one device, and the I/Os to this device will be regulated while it is under its entitlement on another disk and the I/Os to this other device will not be constrained.

In AIX 5.2 and later, WLM supports accounting and regulation of resources on the basis of total consumption. There are two types of resources that can be regulated in this manner: class totals and process totals.

class totals

Per-class limits can be specified for the number of processes, threads, and login sessions in the class. These limits are specified as the absolute number of each resource that can exist in the class at any instant. These limits are strictly enforced; when a class has reached its limit for one of these resources, any attempt to create another instance of the resource will fail. The operation will continue to fail for any process in the class until the class is below its specified limit for the resource.

process totals

Per-process limits can be specified for the total amount of CPU time, blocks of disk I/O, and connect time for a login session. These limits are specified at the class level, but apply to each process in the class individually (each process can consume this amount). These consumption values are cumulative and thus represent the total amount of each particular resource that has been consumed by the process during its lifetime. Once a process has exceeded its total limit for any resource, the process will be terminated. The process will be sent a SIGTERM signal, and if it catches this signal and does not exit after a 5 second grace period, will be sent a SIGKILL signal.

When a login session has reached 90% of its connect time limit, a warning message will be written to the controlling terminal to warn the user that the session will soon be terminated.

Target shares in Workload Manager

The target (or desired) resource consumption percentage for a class is determined by the number of shares it has for a particular resource.

The shares represent how much of a particular resource a class should get, relative to the other classes in its tier. A class's target percentage for a particular resource is simply its number of shares divided by the number of active shares in its tier. If limits are also being used, the target is limited to the range [minimum, soft maximum]. If the calculated target outside this range, it is set to the appropriate upper/lower bound (see Resource Limits). The number of active shares is the total number of shares of all classes that have at least one active process in them. Because the number of active shares is dynamic, so is the target. If a class is the only active class in a tier, its target will be 100% of the amount of resource available to the tier.

For example, assume three active superclasses classes in tier 0—A, B, and C—with shares for a particular resource of 15, 10, and 5, respectively, the targets would be:

$$\begin{aligned}\text{target(A)} &= 15/30 = 50\% \\ \text{target(B)} &= 10/30 = 33\% \\ \text{target(C)} &= 5/30 = 17\%\end{aligned}$$

If some time later, class B becomes inactive (no active processes), the targets for class A and C will be automatically adjusted:

$$\begin{aligned}\text{target(A)} &= 15/20 = 75\% \\ \text{target(C)} &= 5/20 = 25\%\end{aligned}$$

As you can see, the shares represent a self-adapting percentage which allow the resources allocated to a class to be evenly distributed to or taken from the other classes when it becomes active/inactive.

To allow a high degree of flexibility, the number of shares for a class can be any number between 1 and 65535. Shares can be specified for superclasses and subclasses. For superclasses, the shares are relative to all other active superclasses in the same tier. For subclasses, the shares are relative to all other active subclasses in the same superclass, in the same tier. The shares for a subclass one superclass have no relationship to the shares for a subclass of another superclass.

In some cases, it might be desirable to make the target for a class independent from the number of active shares. To accomplish this, a value of "-" can be specified for the number of shares. In this case, the class will be unregulated for that resource, meaning that it has no shares, and its target is not dependent on the number of active shares. Its target will be set to (resource available to the tier - the sum of mins for all other classes in the tier). This target, or actual consumption (whichever is lower) is then taken out of what is available to other classes in the same tier.

For example, assume classes A, B, C, and D have shares for a particular resource of "-", 200, 150, and 100, respectively. All classes are active, and class A is consuming 50% of the resource:

$$\begin{aligned}\text{target(A)} &= \text{unregulated} = 100\% \\ \text{target(B)} &= 200/450 * \text{available} = 44\% * 50\% = 22\% \\ \text{target(C)} &= 150/450 * \text{available} = 33\% * 50\% = 17\% \\ \text{target(D)} &= 100/450 * \text{available} = 22\% * 50\% = 11\%\end{aligned}$$

Because class A is unregulated and is consuming 50% of the available resource, the other classes only have 50% available to them, and their targets are calculated based on this percentage. Since class A will

always be below its target (100%), it will always have a higher priority than all other classes in the same tier that are at or above their targets (see “Class priority in Workload Manager” on page 226 for more information).

Note: Making a class unregulated for a resource is not the same as putting it in a higher tier. The following behaviors, listed here, are true for an unregulated class (in the same tier), and are not true if the class is put in a higher tier:

- Because the shares are defined on a per-resource basis, a class can be unregulated for one or more resources, and regulated for others.
- The minimum limits for other classes in the same tier are honored. Higher tiers do not honor minimums specified in the lower tiers.
- Even in the absence of minimum limits for the classes with shares, the consumption of unregulated classes is somewhat dependent on classes with shares since they are competing for some of the resource available to the tier. Some experimentation is required to see what the behavior is with a given workload.

If the number of shares is unspecified, a default value of “-” will be used, and the class will be unregulated for that resource. Note that in the first version of WLM, the default share value, if unspecified, was 1.

The shares are specified per class for all the resource types. Shares are specified in stanzas of the **shares** file. For example:

```
shares
classname:
  CPU      = 2
  memory   = 4
  diskIO   = 3
```

Resource limit specification in Workload Manager

In addition to using shares to define relative resource entitlements, WLM provides the ability to specify resource limits for a class. Resource limits allow the administrator to have more control over resource allocation. These limits are specified as percentages and are relative to the amount of resource available to the tier that the class is in.

There are three type of limits for percentage-based regulation:

Minimum

This is the minimum amount of a resource that should be made available to the class. If the actual class consumption is below this value, the class will be given highest priority access to the resource. The possible values are 0 to 100, with 0 being the default (if unspecified).

Soft Maximum

This is the maximum amount of a resource that a class can consume when there is contention for that resource. If the class consumption exceeds this value, the class will be given the lowest priority in its tier. If there is no contention for the resource (from other classes in the same tier), the class will be allowed to consume as much as it wants. The possible values are 1 to 100, with 100 being the default (if unspecified).

Hard Maximum

This is the maximum amount of a resource that a class can consume, even when there is no contention. If the class reaches this limit, it will not be allowed to consume any more of the resource until its consumption percentage falls below the limit. The possible values are 1 to 100, with 100 being the default (if unspecified).

Resource limit values are specified in the resource limit file by resource type within stanzas for each class. The limits are specified as a minimum to soft maximum range, separated by a hyphen (-) with white space ignored. The hard maximum when specified follows the soft maximum, and is separated by a semicolon (;). Each limit value is followed by a percent sign (%).

The following are examples using the rules files:

- If user joe from group acct3 executes **/bin/vi**, then the process will be put in superclass acctg.
- If user sue from group dev executes **/bin/emacs**, then the process will be in the superclass devlt (group ID match), but will not be classified into the editors subclass, because user sue is excluded from that class. The process will go to devlt. Default.
- When a DB administrator starts **/usr/sbin/oracle** with a user ID of oracle and a group ID of dbm, to serve the data base DB1, the process will be classified in the Default superclass. Only when the process sets its tag to **_DB1** will it be assigned to superclass db1.

Limits are specified for all resource types, per class, in stanzas of the **limits** file. For example:

```
shares
classname:
  CPU      = 0%-50%;80%
  memory   = 10%-30%;50%
```

In this example, no limits are set for disk I/O. Using the system defaults, this translates to the following:

```
diskIO    = 0%-100%;100%
```

All of the preceding examples assume that the superclasses and subclasses described do not have the inheritance attribute set to yes. Otherwise, the new processes would simply inherit the superclass or subclass from their parent.

The following are the only constraints that WLM places on resource limit values:

- The minimum limit must be less than or equal to the soft maximum limit.
- The soft maximum limit must be less than or equal to the hard maximum limit.
- The sum of the minimum of all the superclasses within a tier cannot exceed 100.
- The sum of the minimum of all the subclasses of a given superclass within a tier cannot exceed 100.

When a class with a hard memory limit has reached this limit and requests more pages, the VMM page replacement algorithm (LRU) is initiated and "steals" pages from the limited class, thereby lowering its number of pages below the hard maximum, before handing out new pages. This behavior is correct, but extra paging activity, which can take place even where there are plenty of free pages available, impacts the general performance of the system. Minimum memory limits for other classes are recommended before imposing a hard memory maximum for any class.

Because classes under their minimum have the highest priority in their tier, the sum of the minimums should be kept to a reasonable level, based on the resource requirements of the other classes in the same tier.

The constraint of the sum of the minimum limits within a tier being less than or equal to 100 means that a class in the highest priority tier is always allowed to get resources up to its minimum limit. WLM does not guarantee that the class will actually reach its minimum limit. This depends on how the processes in the class use their resources and on other limits that are in effect. For example, a class might not reach its minimum CPU entitlement because it cannot get enough memory.

For physical memory, setting a minimum memory limit provides some protection for the memory pages of the class's processes (at least for those in the highest priority tier). A class should not have pages stolen when it is below its minimum limit unless all the active classes are below their minimum limit and one of them requests more pages. A class in the highest tier should never have pages stolen when it is below its minimum. Setting a memory minimum limits for a class of interactive jobs helps make sure that their pages will not all have been stolen between consecutive activations (even when memory is tight) and improves response time.

Attention: Using hard maximum limits can have a significant impact on system or application performance if not used appropriately. Because imposing hard limits can result in unused system resources, in most cases, soft maximum limits are more appropriate. One use for hard maximum limits might be to limit the consumption of a higher tier to make some resource available to a lower tier, though putting applications that need resources in the higher tier may be a better solution.

Beginning with AIX 5.2, the total limits can be specified in the limits file with values and units summarized in the following table:

Table 9. Resource Limits for Workload Manager

Resource	Allowed Units	Default Unit	Maximum Value	Minimum Value
totalCPU	s, m, h, d, w	s	$2^{30} - 1$ s	10 s
totalDiskIO	KB, MB, TB, PB, EB	KB	$(2^{63} - 1) * 512/1024$ KB	1 MB
totalConnectTime	s, m, h, d, w	s	$2^{63} - 1$ s	5 m
totalProcesses	-	-	$2^{63} - 1$	2
totalThreads	-	-	$2^{63} - 1$	2
totalLogins	-	-	$2^{63} - 1$	1

Note: The unit specifiers are not case sensitive. s = seconds, m = minutes, h = hours, d = days, w = weeks, KB = kilobytes, MK = megabytes, ... etc.

An example limits stanza follows:

```
BadUserClass:
    totalCPU = 1m
    totalConnectTime = 1h
```

The total limits can be specified using any value in the above table with the following restrictions:

- If specified, the value for totalThreads must be at least the value of totalProcesses.
- If totalThreads is specified and totalProcesses is not, the limit for totalProcesses will be set to the value of totalThreads.

The total limits can be specified at the superclass and subclass level. When checking the limits, the subclass limit is checked before the superclass limit. If both limits are specified, the lower of the two is enforced. If the specified subclass limit is greater than its associated superclass limit, a warning will be issued when the configuration is loaded, but it will be loaded. This is significant for the class total limits since the limit is absolute (not relative to the superclass) and one subclass could consume all resources available to the superclass. If unspecified, the default value for all total limits is "-" which means no limit. By default, class and process total accounting and regulation will be enabled when WLM is running. The **-T [classproc]** option of the **wlmcntrl** command can be used to disable total accounting and regulation.

Class priority in Workload Manager

WLM allocates resources to classes by giving a priority to each class for each resource.

The priority of a class is dynamic and is based on the tier, shares, limits, and the current consumption of the class. At any given time, the highest priority class or classes will be given preferential access to resources. At the highest level, tiers represent non-overlapping ranges of class priority. Classes in tier 0 will always be guaranteed to have a higher priority than classes in tier 1 (unless over hard maximum).

When determining class priority, WLM enforces its constraints with the following precedence (highest to lowest):

hard limits

If class consumption exceeds the hard maximum for a resource, the class is given the lowest-possible priority for the resource and will be denied access until its consumption falls below this limit.

tier In the absence of hard limits, a class's priority will be bounded by the minimum and maximum allowed priority for its tier.

soft limits

If class consumption is below the minimum of the soft maximum limits for a resource, the class is given the highest-priority in the tier. If class consumption is above the soft maximum, the class is given the lowest-priority in the tier.

shares

These are used to calculate the class consumption targets for each resource. The class priority is increased as class consumption falls below the target, and is decreased as it rises above the target. Note that soft limits have higher precedence and the class's priority will be determined based on them, when applicable.

Even though both shares and limits can be used for each class and for each resource, results are more predictable if only one or the other is used per class.

Resource sets in Workload Manager

WLM uses resource sets (or *rsets*) to restrict the processes in a given class to a subset of the system's physical resources. In WLM, the physical resources managed are the memory and the processors. A valid resource set is composed of memory and at least one processor.

Using SMIT or Web-based System Manager, a system administrator can define and name resource sets containing a subset of the resources available on the system. Then, using the WLM administration interfaces, root user or a designated superclass administrator can use the name of the resource set as the **rset** attribute of a WLM class. From then on, every process assigned to this WLM class is dispatched only on one of the processors in the resource set, effectively separating workloads for the CPU resource.

All of the current systems have only one memory domain shared by all the resource sets, so this method does not physically separate workloads in memory.

Resource set registry in Workload Manager

The **rset** registry services enable system administrators to define and name resource sets so that they can then be used by other users or applications.

To alleviate the risks of name collisions, the registry supports a two-level naming scheme. The name of a resource set is in the form *name_space/rset_name*. Both the *name_space* and *rset_name* can each be 255 characters in length, are case-sensitive, and may contain only uppercase and lowercase letters, numbers, underscores, and periods (.). The *name_space* of **sys** is reserved by the operating system and used for **rset** definitions that represent the resources of the system.

The **rset** definition names are unique within the registry name space. Adding a new **rset** definition to the registry using the same name as an existing **rset** definition causes the existing definition to be replaced with the new definition, given the proper permission and privilege. Only root can create, modify, and delete resource sets and update the in-core **rset** database, using SMIT or Web-based System Manager.

Each **rset** definition has an owner (user ID), group (group ID), and access permissions associated with it. These are specified at the time the **rset** definition is created and exist for the purpose of access control. As is the case for files, separate access permissions exist for the owner, group and others that define whether read and/or write permission has been granted. Read permission allows an **rset** definition to be retrieved while write permission allows an rset definition to be modified or removed.

System Administrator defined **rset** definitions are kept in the **/etc/rsets** stanza file. The format of this file is not described, and users must manipulate **rset** through the SMIT or Web-based System Manager interfaces to prevent future potential compatibility problems if the file format is modified. As is the case for WLM class definitions, the **rset** definitions must be loaded into kernel data structures before they can be used by WLM.

For an example that details what is in a resource set and how to create new one, see [Creating a Resource Set](#).

Setup

Class definitions, class attributes, the shares and limits, and the automatic class assignment rules, can be entered using Web-based System Manager, SMIT, or the WLM command-line interface. These definitions and rules are kept in plain text files, that can also be created or modified using a text editor.

These files (called *WLM property files*) are kept in the subdirectories of **/etc/wlm**. A set of files describing superclasses and their associated subclasses define a WLM configuration. The files for the WLM **Config** configuration are in **/etc/wlm/Config**. This directory contains the definitions of the WLM parameters for the superclasses. The files are named **description**, **classes**, **shares**, **limits**, and **rules**. This directory might also contain subdirectories with the name of the superclasses where the subclass definitions are kept. For example, for the superclass *Super* of the WLM configuration **Config**, the directory **/etc/wlm/Config/Super** contains the property files for the subclasses of the superclass *Super*. The files are named **description**, **classes**, **shares**, **limits**, and **rules**.

After a WLM configuration has been defined by the system administrator, it can be made the active configuration using Web-based System Manager, the **smit wlmmanage** fast path, or the **wlmcntrl** command.

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.

Instructions for creating a WLM configuration are described in [Configure Workload Manager \(WLM\) to Consolidate Workloads](#). This topic describes the concepts and requirements of each phase of the procedure.

Application requirements for Workload Manager configuration

The first phase of defining a configuration requires an understanding of your users and their computing needs, as well as the applications on your system, their resource needs and the requirements of your business (for example, which tasks are critical and which can be given a lower priority). Based on this understanding, you define your superclasses and then your subclasses.

Setting priorities is dependent on what function WLM serves in your organization. In the case of server consolidation, you might already know the applications, the users and their resource requirements, and you might be able to skip or shorten some of the steps.

WLM allows you to classify processes by user or group, application, type, tag, or a combination of these attributes. Because WLM regulates the resource utilization among classes, system administrators should group applications and users with the same resource utilization patterns into the same classes. For instance, you might want to separate the interactive jobs that typically consume very little CPU time but require quick response time from batch-type jobs that typically are very CPU- and memory-intensive. This is the same in a database environment in which you need to separate the OLTP-type traffic from the heavy-duty queries of data mining.

This step is done using Web-based System Manager, SMIT, or command-line interface. For the first few times, it is probably a good idea to use Web-based System Manager or SMIT to take you through the steps of creating your first WLM configuration, including defining the superclasses and setting their attributes. For the first pass, you can set up some of the attributes and leave the others at their default value. This is the same for the resource shares and limits. All these class characteristics can be dynamically modified at a later time.

You can then start WLM in passive mode, check your classification, and start reviewing the resource utilization patterns of your applications.

Verify your configuration, using the **wlmcheck** command or the corresponding SMIT or Web-based System Manager menus. Then start WLM in passive mode on the newly defined configuration. WLM will classify all the existing processes (and all processes created from that point on) and start compiling statistics on the CPU, memory, and disk I/O utilization of the various classes. WLM will not try to regulate this resource usage.

Verify that the various processes are classified in the appropriate class as expected by the system administrator (using the **-o** flag of the **ps** command). If some of the processes are not classified as you expect, you can modify your assignment rules or set the inheritance bit for some of the classes (if you want the new processes to remain in the same class as their parent) and update WLM. You can repeat the process until you are satisfied with this first level of classification (superclasses).

Running WLM in passive mode and refreshing WLM (always in passive mode) involves low risk, has low overhead operation, and can be done safely on a production system without disturbing normal system operation. To activate and refresh WLM, use the **wlmcntrl** command, invoked either from the command line or from SMIT or Web-based System Manager.

Run WLM in passive mode to gather statistics by using the **wlmstat** command. The **wlmstat** command can be used at regular time intervals to display the per-class resource utilization as a percentage of the total resource available, for superclasses). This allows you to monitor your system for extended periods of time to review the resource utilization of your main applications.

Tiers, shares, and limits in Workload Manager

Using the data gathered from running WLM in passive mode and your business goals, decide which tier number will be given to every superclass and what share of each resource should be given to the various classes.

For some classes, you might want to define minimum or maximum limits. Adjust the shares and tier numbers to reach your resource-allocation goals. Reserve limits for cases that cannot be solved only with shares. Also, you might decide you need to add subclasses.

- Use minimum limits for applications that typically have low resource usage but need a quick response time when activated by an external event. One of the problems faced by interactive jobs in situations where memory becomes tight is that their pages get stolen during the periods of inactivity. A memory minimum limit can be used to protect some of the pages of interactive jobs if the class is in tier 0.
- Use maximum limits to contain some resource-intensive, low-priority jobs. Unless you partition your system resources for other reasons, a hard maximum will make sense mostly for a nonrenewable resource such as memory. This is because of the time it takes to write data out to the paging space if a higher priority class needs pages that the first class would have used. For CPU usage, you can use tiers or soft maximum to make sure that if a higher priority class is immediately assigned CPU time.

When creating and adjusting the parameters of subclasses, you can refresh WLM only for the subclasses of a given superclass that do not affect users and applications in the other superclasses, until you are satisfied with the system behavior.

You can also define other configurations with different parameters, according to the needs of the business. When doing so, you can save time by copying and modifying existing configurations.

Fine-tuning the Workload Manager configuration

Monitor the system using the **wlmstat** command and verify that the regulation done by WLM aligns with your goals and does not unduly deprive some applications of resources while others get more than they should. If this is the case, adjust the shares and refresh WLM.

As you monitor and adjust the shares, limits, and tier numbers, decide whether you want to delegate the administration for the subclasses for some or all of the superclasses. The administrator can then monitor and set up the subclass shares, limits, and tier number.

The administrator of each superclass can repeat this process for the subclasses of each superclass. The only difference is that WLM cannot run in passive mode at the subclass level only. The subclass configuration and tuning has to be done with WLM in active mode. One way not to impact users and applications in the superclass is to start the tier number, and the shares and limits for the subclasses at their default value ('-' (hyphen) for shares, 0% for minimum, and 100% for soft and hard maximum). With these settings, WLM will not regulate the resource allocation between the subclasses.

Application programming interface

Applications can use the WLM APIs, a set of routines in the **/usr/lib/libwlm.a** library, to perform all the tasks that a WLM administrator can perform using the WLM command-line interface.

These include:

- Create, modify, or delete classes
- Change class attributes or resource shares and limits
- Removing classes
- Manually assign processes to classes
- Retrieving WLM statistics

The API allows applications to set an application-defined classification attribute called a *tag*. Setting this tag using a set of values provided by the system administrator (through the application user documentation) allows discrimination between several instances of the same application. The different classes can therefore be classified with different resource entitlements.

In addition, the **wlm_set_tag** routine allows an application to set up an application tag and specify whether this tag should be inherited by child processes at **fork** or **exec**. Threads can also be assigned application tags with the **wlm_set_thread** tag. A thread's application tag can be inherited across the **fork**, **exec** or **pthread_create** subroutines. The library provides support for multi-threaded 32-bit or 64-bit applications.

Application tag

The application tag is a string of characters and is used as one of the classification criteria for the automatic classification of processes or threads (using the **rules** file). This tag basically provides an application-defined classification criteria in addition to the system-defined criteria such as *user*, *group*, *application* and *type*.

When an application process or thread sets its tag, it is immediately reclassified using the superclass and subclass rules in effect for the currently active WLM configuration. WLM then reviews the assignment rules looking for a match, using all the process attributes, including the new tag.

To be effective, this tag must appear in one or more of the assignment rules. The format and the use of the various tags by each application must be clearly specified in the application's administration

documentation and well-known to the WLM administrators so that they can use the various values of the tags in their assignment rules to distinguish between different instances of the same application.

Because different users might have different requirements regarding what set of characteristics of the application processes they want to use to classify those processes, it is recommended that the application provide a set of configuration or run time attributes that can be used to build the tag. The application administrator can specify the format of this tag to the application. The attributes that can be used for the tag and the syntax to specify the format of the WLM tag are application-dependent and are the responsibility of the application provider.

For example, an instance of a database server is able to determine which database it is working on (*db_name*) and through which TCP port a given user is connected (*port_num*). Administrators might have any of the following priorities:

- To create different classes for processes accessing different databases to give each class different resource entitlement
- To separate the processes serving remote requests from different origins and to use the port number as a classification attribute
- To create one superclass for each database and subclass per port number in each superclass.

One way to accommodate these different needs is to specify the content and format of the tag. In this example, assume that the tag can be passed to the application in a configuration file or run time parameter such as `WLM_TAG=$db_name` or `WLM_TAG=$db_name_$port_num`.

When setting its tag, an application can specify whether this tag will be inherited by its children so that all the processes spawned by a specific instance of an application can be classified in the same class. Setting the tag inheritance is how the application tag is most commonly used.

The following is an example of how application tags could be used. In this example, the tag of the database is the same as the database name. Then two instances of the server working on two different databases would set up two different tags, for instance `db1` and `db2`.

A system administrator could create two different classes `dbserv1` and `dbserv2` and classify the two database servers (and all their children if tag inheritance is used) in these classes using the tags. It would then be possible to give each class different resource entitlement according to specific business goals.

The corresponding assignment rules could look similar to the following:

```
* class  resvd user  group      application  type  tag
*
dbserv1  -   -   dbadm     /usr/sbin/dbserv  -   db1
dbserv2  -   -   dbadm     /usr/sbin/dbserv  -   db2
```

Application programming interface types

The following are the Workload Manager (WLM) application programming interface types.

Class Management APIs

The WLM API provides applications with the ability to:

- Query the names and characteristics of the existing classes of a given WLM configuration (**wlm_read_classes**).
- Create a new class for a given WLM configuration, define the values of the various attributes of the class (such as tier and inheritance) and the shares and limits for the resources managed by WLM, such as CPU, physical memory, and block I/O (**wlm_create_class**).
- Change the characteristics of an existing class of a given WLM configuration, including the class attributes and resource shares and limits (**wlm_change_class**).
- Delete an existing class of a given configuration (**wlm_delete_class**).

The changes will be applied only to the property files of the specified WLM configuration. Optionally, by specifying an empty string as the configuration name, it is possible to apply the change only to the in-core classes, resulting in an immediate update of the active configuration state.

The API calls require from the caller the same level of privilege that would be required for the command line or for the SMIT or Web-based System Manager interfaces, as follows:

- Any user can read the class names and characteristics
- Only the root user can create, modify, or delete superclasses
- Only the root user or designated superclass administrators (superclass attributes **adminuser** or **admingroup**) can create, modify, or delete subclasses of a given superclass.

In cases where WLM administration is accomplished, both through the command line and administration tools by WLM administrators, and by application(s) through the API, some caution must be applied. Both interfaces share the same name space for the superclass and subclass names, and the total number of superclasses and subclasses.

In addition, when the API directly modifies the in-core WLM data (create new classes, for example), WLM administrators are not aware of this until classes that they did not create appear on the output of commands such as the **wlmstat** command. To avoid conflicts that would confuse the applications using this API when the system administrator updates WLM, the classes created through the API that are not defined in the WLM property files are not automatically removed from the in-core data. They remain in effect until explicitly removed through the **wlm_delete_class** routine or through an invocation of the **rmclass** command (invoked directly or through SMIT or Web-based System Manager by the system administrator).

The WLM API also provides applications with the ability to:

- Query or change the mode of operation of WLM using the **wlm_set** function
- Query the current status of WLM
- Stop WLM
- Toggle between active to passive mode
- Turn the **rset** binding on and off
- Start or update WLM with the current or an alternate configuration by using the **wlm_load** routine
- Assign a process or a group of processes to a class by using the **wlm_assign** routine.

The API requires the same levels of privilege as the corresponding **wlmcntrl** and **wlmassign** commands:

- Any user can query the state of WLM
- Only the root user can change the mode of operation of WLM
- Only the root user can update or refresh a whole configuration
- root or an authorized superclass administrator (**adminuser** or **admingroup**) can update WLM for the subclasses of a given superclass
- root, an authorized user (specified by **authuser** or **authgroup**), or an authorized superclass administrator (**adminuser** or **admingroup**) can assign processes to a superclass or subclass. See the **wlmassign** command for details.

WLM Statistics API

The WLM API routines and **wlm_get_bio_stats** provide application access to the WLM statistics displayed by the **wlmstat** commands.

WLM Classification API

The **wlm_check** routine allows the user to verify the class definitions and the assignment rules for a given WLM configuration. The API routine **wlm_classify** allows an application to determine to which class a process with a specified set of attributes would be classified.

Binary compatibility

To provide binary compatibility if there are future changes to the data structures, each API call is passed a version number as a parameter.

This allows the library to determine with which version of the data structures the application has been built.

Commands

WLM offers commands that allow system administrators to perform a variety of functions.

These include:

- Create, modify, and delete superclasses and subclasses, using the **mkclass**, **chclass**, and **rmclass** commands. These commands update the file's *classes*, *shares* and *limits*.
- Start, stop, and update WLM, using the **wlmcntrl** command.
- Check the WLM property files for a given configuration and determine to which class (superclass and subclass) a process with a given set of attributes is assigned using the **wlmcheck** command.
- Monitor the per-class resource utilization using the **wlmstat** (ASCII) command. Most of the performance tools, such as those started by the **svmon** and **topas** commands, have extensions to take into account the WLM classes and provide per-class and per-tier statistics using new command-line options.
- Flags in the **ps** command allow the user to display which class a process and its application tag is in. The **ps** command also allows the user to list all the processes belonging to a given superclass or subclass.
- There is no command line interface to manage the assignment rules. You must use the SMIT or Web-based System Manager administration tools, or a text editor.

Examples of Workload Manager classification, rules, 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 of Workload Manager assignment rules

This example shows a top-level **rules** file for the configuration *Config* (*/etc/wlm/Config/rules* file).

```
* This file contains the rules used by WLM to
* assign a process to a superclass
*
* class resvd user      group  application      type  tag
db1      -      -      -      /usr/bin/oracle*  -     _DB1
db2      -      -      -      /usr/bin/oracle*  -     _DB2
devlt    -      -      dev      -              -     -
VPs      -      bob,ted  -          -              -     -
acctg    -      -      acct*     -              -     -
System   -      root     -          -              -     -
Default  -      -      -          -              -     -
```

The following is an example of the **rules** file for the **devlt** superclass in the */etc/wlm/Config/devlt/rules* file:

```
* This file contains the rules used by WLM to
* assign a process to a subclass of the
* superclass devlt
*
* class resvd user      group  application      type  tag
hackers  -      jim,liz  -          -              -     -
hogs     -      -      -          -              64bit+plock  -
```

```

editors - !sue - /bin/vi,/bin/emacs - -
build - - - /bin/make,/bin/cc - -
Default - - - - -

```

Note: The asterisk (*) is the comment character used in the **rules** file.

The following are examples using this rules file. The following examples assume that the superclasses and subclasses described do not have the inheritance attribute set to yes. If inheritance were enabled, new processes would inherit the superclass or subclass from their parent processes.

- If user joe from group acct3 executes **/bin/vi**, then the process will be put in superclass acctg.
- If user sue from group dev executes **/bin/emacs**, then the process will be put in the superclass devlt (group ID match), but will not be classified into the editors subclass, because the user is excluded from that class. The process will go to devlt by default.
- When a database administrator with a user ID of oracle and a group ID of dbm starts **/usr/sbin/oracle** to serve the DB1 database, the process will be classified in the Default superclass. Only when the process sets its tag to **_DB1** will it be assigned to the superclass db1.

Example of Workload Manager classes with shares and limits

For this example, assume classes A, B, C, and D have shares of 3, 2, 1, and 1 respectively.

If classes A, C, and D are active, the calculated targets would be:

```

target(A) = 3/5 = 60%
target(C) = 1/5 = 20%
target(D) = 1/5 = 20%

```

If during testing it was found that the applications in class A perform adequately when allowed to use 50% of the resource, it might be desirable to make the other 50% of the resource available to the other classes. This can be accomplished by giving class A a soft maximum of 50% for this resource. Since the current calculated target of 60% is over this limit, it will be adjusted down to the soft maximum value. When this happens, the target or actual consumption (whichever is lower) of class A is subtracted from the amount of resource available. Because this class now has a target constrained by its limit (and not its shares), the shares for the class are also subtracted from the number of active shares. Assuming class A has a current consumption of 48%, the targets will now be:

```

target(A) = 3/5 = 60%, softmax = 50, = 50%
target(C) = 1/2 * (100 - 48) = 26%
target(D) = 1/2 * (100 - 48) = 26%

```

Some time later, all classes may become active, and the targets will again be automatically adjusted:

```

target(A) = 3/7 = 42%
target(B) = 2/7 = 28%
target(C) = 1/7 = 14%
target(D) = 1/7 = 14%

```

Example of Workload Manager classes with CPU limits

This example examines CPU allocation, assuming that each class can consume all the CPU that it is given.

Two classes, A and B, are in the same tier. CPU limits for A are [30% - 100%]. CPU limits for B are [20% - 100%]. When both classes are running and are using sufficient CPU, WLM first makes sure that they both get their minimum percentages of each second (averaged over several seconds). Then WLM distributes the remaining CPU cycles according to any CPU target share values.

If the CPU target shares for A and B are 60% and 40% respectively, then the CPU utilization for A and B stabilize at 60% and 40% respectively.

A third class, C, is added. This class is a group of CPU-bound jobs and should run with about half (or more) of the available CPU. Class C has limits of [20% - 100%] and CPU target shares of 100%. If C is in the same tier as A and B, then when C is starting, A and B see their CPU allocation decrease steeply and the three classes stabilize at 30%, 20% and 50%, respectively. Their targets in this case are also the minimum for A and B.

A system administrator might not want batch jobs to consume 50% of the CPU when other jobs, possibly of higher priority, are also running. In a situation like the previous example, C is placed in a lower priority tier. C then receives whatever CPU remains after A and B receive their needs. In the above example, C receives no CPU time, because A and B are each capable of absorbing 100% of the CPU. In most situations, however, A and B, in a high-priority tier, are composed of interactive or transaction-oriented jobs, which do not use all of the CPU all of the time. C then receives some share of the CPU, for which it competes with other classes in the same or lower tiers.

Example of Workload Manager classes 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.

Backward compatibility in Workload Manager

When starting WLM with configurations created with versions earlier than AIX 5.1, only superclasses are used.

The default output of the **wlmstat** command lists only the superclasses and is similar to those of previous versions. For example:

```
# wlmstat
      CLASS  CPU  MEM  DKIO
Unclassified  0   0   0
  Unmanaged  0   0   0
    Default  0   0   0
      Shared  0   2   0
      System  2  12   0
      class1  0   0   0
      class2  0   0   0
#
```

If some of the superclasses have subclasses defined by a WLM administrator, then the subclasses are listed. For example:

```
# wlmstat
      CLASS  CPU  MEM  DKIO
Unclassified  0   0   0
  Unmanaged   0   0   0
    Default   0   0   0
    Shared    0   2   0
    System    3  11   7
  class1     46   0   0
class1.Default 28   0   0
class1.Shared   0   0   0
  class1.sub1  18   0   0
    class2    48   0   0
#
```

The output is the same when you use the **ps** command. For processes in a superclass without any subclasses, the **ps** command lists the superclass name as the process class name.

Exclusive use processor resource sets

Exclusive use processor resource sets (XRSETs) allow administrators to guarantee resources for important work. An XRSET is a named resource set that changes the behavior of all the CPUs that it includes. Once a CPU is exclusive, it only runs programs explicitly directed to it.

Creating an XRSET

You must be a root user to create an XRSET. Use the **mkrset** command to create a resource set in the **sysxrset** namespace. For example, the command **mkrset -c 1-3 sysxrset/set1** creates an XRSET for CPUs 1, 2, and 3. The **rs_registername()** subroutine can also be used to create an XRSET.

Determining if XRSETs have been Defined on a System

The **lsrset -v -n sysxrset** command displays all XRSETs defined on a system. (There is currently no programming API to do this.)

Deleting an XRSET

You must be a root user to delete an XRSET. The **rmrset** command deletes an XRSET. The **rs_discardname()** subroutine may also be used to delete an XRSET.

Rebooting the System

When you reboot the system, any XRSETs that were set are removed from the registry and are no longer in effect.

Specifying Work for XRSETs

There are multiple ways for work to be marked as eligible to use exclusive use processors. The **attachrset** and **execrset** commands can be used to specify resource sets that contain exclusive use processors. Resource sets containing exclusive use processors can be associated with WLM classes. Work classified into such WLM classes will use the exclusive use processors specified in the resource set.

Using XRSETs with Bindprocessor and `_system_configuration.ncpus`

You cannot use **bindprocessor** to cause work to run on exclusive use processors. Only resource set-based attachments can cause work to run on exclusive use processors.

The number of CPUs in the system configuration (the `_system_configuration.ncpus` field) is not changed when XRSETs are created. There are still NCPUs in the system.

When programs use the **bindprocessor** system call to NCPUs, CPUs in XRSETs will fail with the EINVAL error. You can bind to any ID returned by the query option of the **bindprocessor** command. The query option (**bindprocessor -q**) will only return valid bind IDs, excluding those that are associated with exclusive CPUs.

For example, if there are 10 CPUs online in a system and three of them are in XRSETs, a **bindprocessor** to CPUs with bind IDs in the range of 0 to 6 will succeed. A **bindprocessor** to CPUs with bind IDs in the range of 7 to 9 will receive an EINVAL error.

Using XRSETs with Dynamic CPU Reconfiguration Operations

In general, dynamic CPU reconfiguration is not affected by exclusive use processors. However, the creation of XRSETs and assignment of work to those processors may prevent removal of a CPU. CPUs that are dynamically added to the system may enter the system as either general use or exclusive use processors. They will enter the system as exclusive use if there is an XRSET containing the logical CPU ID when it enters the system.

Chapter 10. System Resource Controller

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

The following are the properties and components of a subsystem.

A subsystem can have one or more of the following properties:

- Is known to the system by name
- Requires a more complex execution environment than a subroutine or nonprivileged program
- Includes application programs and libraries as well as subsystem code
- Controls resources that can be started and stopped by name
- Requires notification if a related process is unsuccessful to perform cleanup or to recover resources
- Requires more operational control than a simple daemon process
- Needs to be controlled by a remote operator
- Implements subservers to manage specific resources
- Does not put itself in the background.

A few subsystem examples are ypserv, ntsd, qdaemon, inetd, syslogd, and sendmail.

Note: See each specific subsystem for details of its SRC capabilities.

Use the **lssrc -a** command to list active and inactive subsystems on your system.

The following defines subsystem groups and subservers:

Subsystem Group

A *subsystem group* is a group of any specified subsystems. Grouping subsystems together allows the control of several subsystems at one time. A few subsystem group examples are TCP/IP, SNA Services, Network Information System (NIS), and Network File Systems (NFS).

Subserver

A *subserver* is a program or process that belongs to a subsystem. A subsystem can have multiple subservers and is responsible for starting, stopping, and providing status of subservers. Subservers can be defined only for a subsystem with a communication type of IPC message queues and sockets. Subsystems using signal communications do not support subservers.

Subservers are started when their parent subsystems are started. If you try to start a subserver and its parent subsystem is not active, the **startsrc** command starts the subsystem as well.

SRC hierarchy

The System Resource Controller hierarchy begins with the operating system followed by a subsystem group (such as **tcpip**), which contains a subsystem (such as the **inetd** daemon), which in turn can own several subservers (such as the **ftp** daemon and the **finger** command).

SRC administration commands

You can administer SRC from the command line.

The SRC administration commands are:

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.

Starting the System Resource Controller

The System Resource Controller (SRC) is started during system initialization with a record for the **/usr/sbin/srcmstr** daemon in the **/etc/inittab** file.

The following are the prerequisites for starting the SRC:

- Reading and writing the **/etc/inittab** file requires root user authority.
- The **mkinitab** command requires root user authority.
- The **srcmstr** daemon record must exist in the **/etc/inittab** file.

The default **/etc/inittab** file already contains such a record, so this procedure might be unnecessary. You can also start the SRC from the command line, a profile, or a shell script, but there are several reasons for starting it during initialization:

- Starting the SRC from the **/etc/inittab** file allows the **init** command to restart the SRC if it stops for any reason.
- The SRC is designed to simplify and reduce the amount of operator intervention required to control subsystems. Starting the SRC from any source other than the **/etc/inittab** file is counterproductive to that goal.
- The default **/etc/inittab** file contains a record for starting the print scheduling subsystem (**qdaemon**) with the **startsrc** command. Typical installations have other subsystems started with **startsrc** commands in the **/etc/inittab** file as well. Because the **srcmstr** command requires the SRC be running, removing the **srcmstr** daemon from the **/etc/inittab** file causes these **startsrc** commands to fail.

See the **srcmstr** command for the configuration requirements to support remote SRC requests.

Note: This procedure is necessary only if the **/etc/inittab** file does not already contain a record for the **srcmstr** daemon.

1. Make a record for the **srcmstr** daemon in the **/etc/inittab** file using the **mkitab** command. For example, to make a record identical to the one that appears in the default **/etc/inittab** file, type:

```
mkitab -i fbcheck srcmstr:2:respawn:/usr/sbin/srcmstr
```

The **-i fbcheck** flag ensures that the record is inserted before all subsystems records.

2. Tell the **init** command to reprocess the **/etc/inittab** file by typing:

```
telinit q
```

When **init** revisits the **/etc/inittab** file, it processes the newly entered record for the **srcmstr** daemon and starts the SRC.

Starting or stopping a subsystem, subsystem group, or subserver

Use the **startsrc** command to start a System Resource Controller (SRC) resource such as a subsystem, a group of subsystems, or a subserver. Use the **stopsrc** command to stop an SRC resource such as a subsystem, a group of subsystems, or a subserver.

The following are the prerequisites for starting or stopping a subsystem, subsystem group, or subserver:

- To start or stop an SRC resource, the SRC must be running. The SRC is normally started during system initialization. The default **/etc/inittab** file, which determines what processes are started during initialization, contains a record for the **srcmstr** daemon (the SRC). To see if the SRC is running, type **ps -A** and look for a process named **srcmstr**.
- The user or process starting an SRC resource must have root user authority. The process that initializes the system (**init** command) has root user authority.
- The user or process stopping an SRC resource must have root user authority.

The **startsrc** command can be used:

- From the **/etc/inittab** file so the resource is started during system initialization
- From the command line
- With SMIT.

When you start a subsystem group, all of its subsystems are also started. When you start a subsystem, all of its subservers are also started. When you start a subserver, its parent subsystem is also started if it is not already running.

When you stop a subsystem, all its subservers are also stopped. However, when you stop a subserver, the state of its parent subsystem is not changed.

Both the **startsrc** and **stopsrc** commands contain flags that allow requests to be made on local or remote hosts. See the **srcmstr** command for the configuration requirements to support remote SRC requests.

Starting/Stopping a Subsystem Tasks

<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Start a Subsystem	smit startssys	/bin/startsrc -s SubsystemName OR edit /etc/inittab
Stop a Subsystem	smit stopssys	/bin/stopsrcSubsystemName

Displaying status of a subsystem or subsystems

Use the **lssrc** command to display the status of a System Resource Controller (SRC) resource such as a subsystem, a group of subsystems, or a subserver.

All subsystems can return a short status report that includes which group the subsystem belongs to, whether the subsystem is active, and what its process ID (PID) is. If a subsystem does not use the signals communication method, it can be programmed to return a long status report containing additional status information.

The **lssrc** command provides flags and parameters for specifying the subsystem by name or PID, for listing all subsystems, for requesting a short or long status report, and for requesting the status of SRC resources either locally or on remote hosts.

See the **srcmstr** command for the configuration requirements to support remote SRC requests.

Displaying the Status of Subsystems Tasks

<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Display the status of a subsystem (long format)	smit qssys	lssrc -l -s <i>SubsystemName</i>
Display the status of all subsystems	smit lssys	lssrc -a
Display the status of all subsystems on a particular host		lssrc -hHostName -a

Refreshing a subsystem or subsystem group

Use the **refresh** command to tell a System Resource Controller (SRC) resource such as a subsystem or a group of subsystems to refresh itself.

The following are the prerequisites for refreshing a subsystem or subsystem group:

- The SRC must be running. See “Starting the System Resource Controller” on page 240 for details.
- The resource you want to refresh must not use the signals communications method.
- The resource you want to refresh must be programmed to respond to the refresh request.

The **refresh** command provides flags and parameters for specifying the subsystem by name or PID. You can also use it to request a subsystem or group of subsystems be refreshed, either locally or on remote hosts. See the **srcmstr** command for the configuration requirements to support remote SRC requests.

Refreshing a Subsystem or Subsystem Group

Task	SMIT Fast Path	Command or File
Refresh a Subsystem	smit refresh	refresh -s Subsystem

Subsystem control

The **traceson** command can be used to turn on, and the **traceoff** command can be used to turn off, tracing of a System Resource Controller (SRC) resource such as a subsystem, a group of subsystems, or a subserver.

Use the **traceson** command to turn on tracing of a System Resource Controller (SRC) resource such as a subsystem, a group of subsystems, or a subserver.

Use the **traceoff** command to turn off tracing of a System Resource Controller (SRC) resource such as a subsystem, a group of subsystems, or a subserver.

The **traceson** and **traceoff** commands can be used to remotely turn on or turn off tracing on a specific host. See the **srcmstr** command for the configuration requirements for supporting remote SRC requests.

Prerequisites

- To turn the tracing of an SRC resource either on or off , the SRC must be running. See “Starting the System Resource Controller” on page 240 for details.
- The resource you want to trace must not use the signals communications method.
- The resource you want to trace must be programmed to respond to the trace request.

Turning On/Off Subsystem, Subsystem Group, or Subserver Tasks

<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Turn on Subsystem Tracing (short format)	smit tracessyson	traceson -s Subsystem
Turn on Subsystem Tracing (long format)	smit tracessyson	traceson -l -s Subsystem
Turn off Subsystem Tracing	smit tracessysoff	tracesoff -s Subsystem

Chapter 11. System accounting

The system accounting utility allows you to collect and report on individual and group use of various system resources.

Note: A new advanced accounting subsystem is available beginning with AIX 5.3. For more information, see *AIX 5L Version 5.3 Understanding the accounting subsystem*

This accounting information can be used to bill users for the system resources they utilize, and to monitor selected aspects of the system operation. To assist with billing, the accounting system provides the resource-usage totals defined by members of the adm group, and, if the **chargefee** command is included, factors in the billing fee.

The accounting system also provides data to assess the adequacy of current resource assignments, set resource limits and quotas, forecast future needs, and order supplies for printers and other devices.

The following information should help you understand how to implement the accounting utility in your system.

Setting up an accounting system

You can set up an accounting system.

You must have root authority to complete this procedure.

The information below is an overview of the steps you must take to set up an accounting system. Refer to the commands and files noted in these steps for more specific information.

1. Use the **nulladm** command to ensure that each file has the correct access permission: read (r) and write (w) permission for the file owner and group and read (r) permission for others by typing:

```
/usr/sbin/acct/nulladm wtmp pacct
```

This provides access to the **pacct** and **wtmp** files.

2. Update the **/etc/acct/holidays** file to include the hours you designate as prime time and to reflect your holiday schedule for the year.

Note: Comment lines can appear anywhere in the file as long as the first character in the line is an asterisk (*).

- a. To define prime time, fill in the fields on the first data line (the first line that is not a comment), using a 24-hour clock. This line consists of three 4-digit fields, in the following order:

- 1) Current year
- 2) Beginning of prime time (*hhmm*)
- 3) End of prime time (*hhmm*)

Leading blanks are ignored. You can enter midnight as either 0000 or 2400.

For example, to specify the year 2000, with prime time beginning at 8:00 a.m. and ending at 5:00 p.m., enter:

```
2000 0800 1700
```

- b. To define the company holidays for the year, fill in the next data line. Each line contains four fields, in the following order:

- 1) Day of the year
- 2) Month
- 3) Day of the month

4) Description of holiday

The day-of-the-year field contains the number of the day on which the holiday falls and must be a number from 1 through 365 (366 on leap year). For example, February 1st is day 32. The other three fields are for information only and are treated as comments.

A two-line example follows:

```
1 Jan 1 New Year's Day
332 Nov 28 Thanksgiving Day
```

3. Turn on process accounting by adding the following line to the **/etc/rc** file or by deleting the comment symbol (#) in front of the line if it exists:

```
/usr/bin/su - adm -c /usr/sbin/acct/startup
```

The **startup** procedure records the time that accounting was turned on and cleans up the previous day's accounting files.

4. Identify each file system you want included in disk accounting by adding the following line to the stanza for the file system in the **/etc/filesystems** file:

```
account = true
```

5. Specify the data file to use for printer data by adding the following line to the queue stanza in the **/etc/qconfig** file:

```
acctfile = /var/adm/qacct
```

6. As the adm user, create a **/var/adm/acct/nite**, a **/var/adm/acct/fiscal**, a and **/var/adm/acct/sum** directory to collect daily and fiscal period records:

```
su - adm
cd /var/adm/acct
mkdir nite fiscal sum
exit
```

For long usernames, use the following commands instead:

```
su - adm
cd /var/adm/acct
mkdir nitex fiscalx sumx
exit
```

7. Set daily accounting procedures to run automatically by editing the **/var/spool/cron/crontabs/adm** file to include the **dodisk**, **ckpacct**, and **runacct** commands. For example:

```
0 2 * * 4 /usr/sbin/acct/dodisk
5 * * * * /usr/sbin/acct/ckpacct
0 4 * * 1-6 /usr/sbin/acct/runacct
    2>/var/adm/acct/nite/accterr
```

For long usernames, add the following lines instead:

```
0 2 * * 4 /usr/sbin/acct/dodisk -X
5 * * * * /usr/sbin/acct/ckpacct
0 4 * * 1-6 /usr/sbin/acct/runacct -X
    2>/var/adm/acct/nitex/accterr
```

The first line starts disk accounting at 2:00 a.m. (0 2) each Thursday (4). The second line starts a check of the integrity of the active data files at 5 minutes past each hour (5 *) every day (*). The third line runs most accounting procedures and processes active data files at 4:00 a.m. (0 4) every Monday through Saturday (1-6). If these times do not fit the hours your system operates, adjust your entries.

Note: You must have root user authority to edit the **/var/spool/cron/crontabs/adm** file.

8. Set the monthly accounting summary to run automatically by including the **monacct** command in the **/var/spool/cron/crontabs/adm** file. For example, type:

```
15 5 1 * * /usr/sbin/acct/monacct
```

For long usernames, add the following line instead:

```
15 5 1 * * /usr/sbin/acct/monacct -X
```

Be sure to schedule this procedure early enough to finish the report. This example starts the procedure at 5:15 a.m. on the first day of each month.

9. To submit the edited **cron** file, type:

```
crontab /var/spool/cron/crontabs/adm
```

Displaying Accounting system activity

You can display formatted information about system activity with the **sar** command.

To display system activity statistics, the **sadc** command must be running.

Note: The typical method of running the **sadc** command is to place an entry for the **sa1** command in the root **crontab** file. The **sa1** command is a shell-procedure variant of the **sadc** command designed to work with the **cron** daemon.

To display basic system-activity information, type:

```
sar 2 6
```

where the first number is the number of seconds between sampling intervals and the second number is the number of intervals to display. The output of this command looks something like this:

```
arthurd 2 3 000166021000 05/28/92
14:03:40  %usr  %sys  %wio  %idle
14:03:42    4    9    0    88
14:03:43    1   10    0    89
14:03:44    1   11    0    88
14:03:45    1   11    0    88
14:03:46    3    9    0    88
14:03:47    2   10    0    88
Average    2   10    0    88
```

The **sar** command also offers a number of flags for displaying an extensive array of system statistics. To see all available statistics, use the **-A** flag. For a list of the available statistics and the flags for displaying them, see the **sar** command.

Note: To have a daily system activity report written to **/var/adm/sa/sadd**, include an entry in the root **crontab** file for the **sa2** command. The **sa2** command is a shell procedure variant for the **sar** command designed to work with the **cron** daemon.

Showing Accounting system activity while running a command

You can display formatted information about system activity while a particular command is running.

The **-o** and **-p** flags of the **timex** command require that system accounting be turned on.

You can use the **time** and **timex** commands to display formatted information about system activity while a particular command is running.

To display the elapsed time, user time, and system execution time for a particular command, type:

```
time CommandName
```

OR

```
timex CommandName
```

To display the total system activity (all the data items reported by the **sar** command) during the execution of a particular command, type:

```
timex -s CommandName
```

The **timex** command has two additional flags. The **-o** flag reports the total number of blocks read or written by the command and all of its children. The **-p** flag lists all of the process accounting records for a command and all of its children.

Displaying the process time of active Accounting processes

You can display the process time for active processes.

The **acctcom** command reads input in the total accounting record form (**acct** file format). This implies that you have process accounting turned on or that you have run process accounting in the past. See “Setting up an accounting system” on page 245 for guidelines.

The **ps** command offers a number of flags to tailor the information displayed.

To produce a full list of all active processes except kernel processes, type:

```
ps -ef
```

You can also display a list of all processes associated with terminals. To do this, type:

```
ps -al
```

Both of these usages display a number of columns for each process, including the current CPU time for the process in minutes and seconds.

Displaying the process time of finished Accounting processes

You can display the process time of finished processes.

The **acctcom** command reads input in the total accounting record form (**acct** file format). This implies that you have process accounting turned on or that you have run process accounting in the past. See “Setting up an accounting system” on page 245 for guidelines.

The process accounting functions are turned on with the **startup** command, which is typically started at system initialization with a call in the **/etc/rc** file. When the process accounting functions are running, a record is written to **/var/adm/pacct** (a total accounting record file) for every finished process that includes the start and stop time for the process. You can display the process time information from a **pacct** file with the **acctcom** command. This command has a number of flags that allow flexibility in specifying which processes to display.

For example, to see all processes that ran for a minimum number of CPU seconds or longer, use the **-O** flag, type:

```
acctcom -O 2
```

This displays records for every process that ran for at least 2 seconds. If you do not specify an input file, the **acctcom** command reads input from the **/var/adm/pacct** directory.

Showing the CPU usage for each accounting process

You can display formatted reports about the CPU usage by user with the **acctprc1** command.

The **acctprc1** command requires input in the total accounting record form (**acct** file format). This implies that you have process accounting turned on or that you have run process accounting in the past. See “Setting up an accounting system” on page 245 for guidelines.

To produce a formatted report of CPU usage by process, type:

```
acctprc1 </var/adm/pacct
```

Showing the CPU accounting usage for each user

You can display a formatted report about the CPU usage by user with a combination of the **acctprc1** and **prtacct** commands.

The `../../../../com.ibm.aix.cmds/doc/aixcmds1/acctprc1.htm` command requires input in the total accounting record form (**acct** file format). This implies that you have process accounting turned on or that you have run process accounting in the past. See “Setting up an accounting system” on page 245 for guidelines.

To show the CPU usage for each user, perform the following steps:

1. Produce an output file of CPU usage by process by typing:

```
acctprc1 </var/adm/pacct >out.file
```

The `/var/adm/pacct` file is the default output for process accounting records. You might want to specify an archive **pacct** file instead.

2. Produce a binary total accounting record file from the output of the previous step by typing:

```
acctprc2 <out.file >/var/adm/acct/nite/dayacct
```

Note: The **dayacct** file is merged with other total accounting records by the **acctmerg** command to produce the daily summary record, `/var/adm/acct/sum(x)/acct`.

3. Use the `../../../../com.ibm.aix.cmds/doc/aixcmds4/prtacct.htm` command to display a formatted report of CPU usage summarized by the user by typing:

```
prtacct </var/adm/acct/nite/dayacct
```

Displaying connect time usage for accounting

You can display the connect time of all users, of individual users, and by individual login with the **ac** command.

The **ac** command extracts login information from the `/var/adm/wtmp` file, so this file must exist. If the file has not been created, the following error message is returned:

```
No /var/adm/wtmp
```

If the file becomes too full, additional **wtmp** files are created; you can display connect-time information from these files by specifying them with the **-w** flag. For more information about the **ac** command, see **ac**.

To display the total connect time for all users, type:

```
/usr/sbin/acct/ac
```

This command displays a single decimal number that is the sum total connect time, in minutes, for all users who have logged in during the life of the current **wtmp** file.

To display the total connect time for one or more particular users, type:

```
/usr/sbin/acct/ac User1 User2 ...
```

This command displays a single decimal number that is the sum total connect time, in minutes, for the user or users you specified for any logins during the life of the current **wtmp** file.

To display the connect time by individual user plus the total connect time, type:

```
/usr/sbin/acct/ac -p User1 User2 ...
```

This command displays as a decimal number for each user specified equal to the total connect time, in minutes, for that user during the life of the current **wtmp** file. It also displays a decimal number that is the sum total connect time for all the users specified. If no user is specified in the command, the list includes all users who have logged in during the life of the **wtmp** file.

Displaying disk space utilization for Accounting

You can display disk space utilization information with the **acctmrg** command.

To display disk space utilization information, the **acctmrg** command requires input from a **dacct** file (disk accounting). The collection of disk-usage accounting records is performed by the **dodisk** command. Placing an entry for the **dodisk** command in a **crontabs** file is part of the procedure described in “Setting up an accounting system” on page 245. For more information about the **acctmrg** command, see **acctmrg**.

To display disk space utilization information, type:

```
acctmrg -a1 -2,13 -h </var/adm/acct/nite(x)/dacct
```

This command displays disk accounting records, which include the number of 1 KB blocks utilized by each user.

Note: The **acctmrg** command always reads from standard input and can read up to nine additional files. If you are not piping input to the command, you must redirect input from one file; the rest of the files can be specified without redirection.

Displaying printer or plotter usage accounting records

You can display printer or plotter usage accounting records with the **pac** command.

- To collect printer usage information, you must have an accounting system set up and running. See “Setting up an accounting system” on page 245 for guidelines.
- The printer or plotter for which you want accounting records must have an **acctfile=** clause in the printer stanza of the **/etc/qconfig** file. The file specified in the **acctfile=** clause must grant read and write permissions to the root user or **printq** group.
- If the **-s** flag of the **pac** command is specified, the command rewrites the summary file name by appending **_sum** to the path name specified by the **acctfile=** clause in the **/etc/qconfig** file. This file must exist and grant read and write permissions to the root user or **printq** group.

To display printer usage information for all users of a particular printer, type:

```
/usr/sbin/pac -PPrinter
```

If you do not specify a printer, the default printer is named by the **PRINTER** environment variable. If the **PRINTER** variable is not defined, the default is **lp0**.

To display printer usage information for particular users of a particular printer, type:

```
/usr/sbin/pac -PPrinter User1 User2 ...
```

The **pac** command offers other flags for controlling what information gets displayed.

Fixing tacct errors

If you are using the accounting system to charge users for system resources, the integrity of the `/var/adm/acct/sum/tacct` file is quite important. Occasionally, mysterious **tacct** records appear that contain negative numbers, duplicate user numbers, or a user number of 65,535. You can fix these problems.

You must have root user or adm group authority.

To patch a **tacct** file, perform the following steps:

1. Move to the `/var/adm/acct/sum` directory by typing:

```
cd /var/adm/acct/sum
```

2. Use the **prtacct** command to check the total accounting file, **tacctprev**, by typing:

```
prtacct tacctprev
```

The **prtacct** command formats and displays the **tacctprev** file so that you can check connect time, process time, disk usage, and printer usage.

3. If the **tacctprev** file looks correct, change the latest **tacct** `.mmd` file from a binary file to an ASCII file. In the following example, the **acctmerg** command converts the **tacct.mmd** file to an ASCII file named **tacct.new**:

```
acctmerg -v < tacct.mmd > tacct.new
```

Note: The **acctmerg** command with the **-a** flag also produces ASCII output. The **-v** flag produces more precise notation for floating-point numbers.

The **acctmerg** command is used to merge the intermediate accounting record reports into a cumulative total report (**tacct**). This cumulative total is the source from which the **monacct** command produces the ASCII monthly summary report. Since the **monacct** command procedure removes all the **tacct.mmd** files, you recreate the **tacct** file by merging these files.

4. Edit the **tacct.new** file to remove the bad records and write duplicate user number records to another file by typing:

```
acctmerg -i < tacct.new > tacct.mmd
```

5. Create the **tacct** file again by typing:

```
acctmerg tacctprev < tacct.mmd > tacct
```

Fixing wtmp errors

The `/var/adm/wtmp`, or "who temp" file, might cause problems in the day-to-day operation of the accounting system. You can fix **wtmp** errors.

You must have root user or adm group authority to perform this procedure.

When the date is changed and the system is in multiuser mode, date change records are written to the `/var/adm/wtmp` file. When a date change is encountered, the **wtmpfix** command adjusts the time stamps in the **wtmp** records. Some combinations of date changes and system restarts may slip past the **wtmpfix** command and cause the **acctcon1** command to fail and the **runacct** command to send mail to the **root** and **adm** accounts listing incorrect dates.

To fix **wtmp** errors, perform the following procedure:

1. Move to the `/var/adm/acct/nite` directory by typing:

```
cd /var/adm/acct/nite
```

2. Convert the binary **wtmp** file to an ASCII file that you can edit by typing:

```
fwtmp < wtmp.mmd > wtmp.new
```

The **fwtmp** command converts **wtmp** from binary to ASCII.

3. Edit the ASCII **wtmp.new** file to delete damaged records or all records from the beginning of the file up to the needed date change by typing:

```
vi wtmp.new
```

4. Convert the ASCII **wtmp.new** file back to binary format by typing:

```
fwtmp -ic < wtmp.new > wtmp.mmdd
```

5. If the **wtmp** file is beyond repair, use the **nulladm** command to create an empty **wtmp** file. This prevents any charges in the connect time.

```
nulladm wtmp
```

The **nulladm** command creates the file specified with read and write permissions for the file owner and group, and read permissions for other users. It ensures that the file owner and group are **adm**.

Fixing incorrect Accounting file permissions

To use the Accounting system, file ownership and permissions must be correct.

You must have root user or adm group authority to perform this procedure.

The **adm** administrative account owns the accounting command and scripts, except for **/var/adm/acct/accton** which is owned by root.

To fix incorrect Accounting file permissions, perform the following procedure:

1. To check file permissions using the **ls** command, type:

```
ls -l /var/adm/acct
```

```
-rws--x--- 1 adm adm 14628 Mar 19 08:11 /var/adm/acct/fiscal
-rws--x--- 1 adm adm 14628 Mar 19 08:11 /var/adm/acct/nite
-rws--x--- 1 adm adm 14628 Mar 19 08:11 /var/adm/acct/sum
```

2. Adjust file permissions with the **chown** command, if necessary. The permissions are 755 (all permissions for owner and read and execute permissions for all others). Also, the directory itself should be write-protected from others. For example:

- a. Move to the **/var/adm/acct** directory by typing:

```
cd /var/adm/acct
```

- b. Change the ownership for the **sum**, **nite**, and **fiscal** directories to **adm** group authority by typing:

```
chown adm sum/* nite/* fiscal/*
```

To prevent tampering by users trying to avoid charges, deny write permission for others on these files. Change the **accton** command group owner to **adm**, and permissions to 710, that is, no permissions for others. Processes owned by **adm** can execute the **accton** command, but ordinary users cannot.

3. The **/var/adm/wtmp** file must also be owned by **adm**. If **/var/adm/wtmp** is owned by root, you will see the following message during startup:

```
/var/adm/acct/startup: /var/adm/wtmp: Permission denied
```

To correct the ownership of **/var/adm/wtmp**, change ownership to the **adm** group by typing the following command:

```
chown adm /var/adm/wtmp
```

Fixing Accounting errors

You can correct date and time-stamp inconsistencies.

You must have root user or adm group authority to perform this procedure.

Processing the `/var/adm/wtmp` file might produce some warnings mailed to root. The `wtmp` file contains information collected by `/etc/init` and `/bin/login` and is used by accounting scripts primarily for calculating connect time (the length of time a user is logged in). Unfortunately, date changes confuse the program that processes the `wtmp` file. As a result, the `runacct` command sends mail to root and adm complaining of any errors after a date change since the last time accounting was run.

1. Determine if you received any errors. The `acctcon1` command outputs error messages that are mailed to adm and root by the `runacct` command. For example, if the `acctcon1` command stumbles after a date change and fails to collect connect times, adm might get mail like the following mail message:

```
Mon Jan 6 11:58:40 CST 1992
acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
new: Mon Jan 6 11:57:59 1992
acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
new: Mon Jan 6 11:57:59 1992
acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
new: Mon Jan 6 11:57:59 1992
```

2. Adjust the `wtmp` file by typing:

```
/usr/sbin/acct/wtmpfix wtmp
```

The `wtmpfix` command examines the `wtmp` file for date and time-stamp inconsistencies and corrects problems that could make `acctcon1` fail. However, some date changes slip by `wtmpfix`. See “Fixing `wtmp` errors” on page 251.

3. Run accounting right before shutdown or immediately after startup. Using the `runacct` command at these times minimizes the number of entries with bad times. The `runacct` command continues to send mail to the root and adm accounts, until you edit the `runacct` script, find the `WTMPFIX` section, and comment out the line where the file log gets mailed to the root and adm accounts.

Accounting errors encountered when running the `runacct` command

You might encounter errors when running the `runacct` command.

Note: You must have root user or adm group authority to run the `runacct` command.

The `runacct` command processes files that are often very large. The procedure involves several passes through certain files and consumes considerable system resources while it is taking place. Since the `runacct` command consumes considerable resources, it is normally run early in the morning when it can take over the machine and not disturb anyone.

The `runacct` command is a script divided into different stages. The stages allow you to restart the command where it stopped, without having to rerun the entire script.

When the `runacct` encounters problems, it sends error messages to different destinations depending on where the error occurred. Usually it sends a date and a message to the console directing you to look in the `activeMMDD` file (such as `active0621` for June 21st) which is in the `/usr/adm/acct/nite` directory. When the `runacct` command aborts, it moves the entire `active` file to `activeMMDD` and appends a message describing the problem.

Review the following error message tables for errors that you might encounter when running the `runacct` command.

Note:

- The abbreviation `MMDD` stands for the month and day, such as `0102` for January 2. For example, a fatal error during the `CONNECT1` process on January 2 creates the file `active0102` containing the error message.
- The abbreviation “SE message” stands for the standard error message such as:

```
***** ACCT ERRORS : see active0102 *****
```

Preliminary State and Error Messages from the **runacct** Command

State	Command	Fatal?	Error Message	Destinations
pre	runacct	yes	* 2 CRONS or ACCT PROBLEMS * ERROR: locks found, run aborted	console, mail, active
pre	runacct	yes	runacct: Insufficient space in /usr (<i>nnn</i> blks); Terminating procedure	console, mail, active
pre	runacct	yes	SE message; ERROR: acctg already run for 'date': check lastdate	console, mail, activeMMDD
pre	runacct	no	* SYSTEM ACCOUNTING STARTED *	console
pre	runacct	no	restarting acctg for 'date' at STATE	console active, console
pre	runacct	no	restarting acctg for 'date' at state (argument \$2) previous state was STATE	active
pre	runacct	yes	SE message; Error: runacct called with invalid arguments	console, mail, activeMMDD

States and Error Messages from the **runacct** Command

State	Command	Fatal?	Error Message	Destinations
SETUP	runacct	no	ls -l fee pacct* /var/adm/wtmp	active
SETUP	runacct	yes	SE message; ERROR: turnacct switch returned rc=error	console, mail, activeMMDD
SETUP	runacct	yes	SE message; ERROR: SpacctMMDD already exists file setups probably already run	activeMMDD
SETUP	runacct	yes	SE message; ERROR: wtmpMMDD already exists: run setup manually	console, mail, activeMMDD
WTMPFIX	wtmpfix	no	SE message; ERROR: wtmpfix errors see xtmperrorMMDD	activeMMDD, wtmperrorMMDD
WTMPFIX	wtmpfix	no	wtmp processing complete	active
CONNECT1	acctcon1	no	SE message; (errors from acctcon1 log)	console, mail, activeMMDD
CONNECT2	acctcon2	no	connect acctg complete	active

States and Error Messages from the **runacct** Command

State	Command	Fatal?	Error Message	Destinations
PROCESS	runacct	no	WARNING: accounting already run for pacctN	active
PROCESS	acctprc1 acctprc2	no	process acctg complete for SpacctMMDD	active
PROCESS	runacct	no	all process acctg complete for date	active
MERGE	acctmerg	no	tacct merge to create dayacct complete	active
FEES	acctmerg	no	merged fees OR no fees	active
DISK	acctmerg	no	merged disk records OR no disk records	active
MERGEACCT	acctmerg	no	WARNING: recreating sum/tacct	active
MERGEACCT	acctmerg	no	updated sum/tacct	active
CMS	runacct	no	WARNING: recreating sum/cms	active
CMS	acctcms	no	command summaries complete	active
CLEANUP	runacct	no	system accounting completed at 'date'	active
CLEANUP	runacct	no	*SYSTEM ACCOUNTING COMPLETED*	console
<wrong>	runacct	yes	SE message; ERROR: invalid state, check STATE	console, mail, activeMMDD

Note: The label <wrong> in the previous table does not represent a state, but rather a state other than the correct state that was written in the state file **/usr/adm/acct/nite/statefile**.

Summary of Message Destinations

Destination	Description
console	The /dev/console device
mail	Message mailed to root and adm accounts
active	The /usr/adm/acct/nite/active file
activeMMDD	The /usr/adm/acct/nite/activeMMDD file
wtmperrMMDD	The /usr/adm/acct/nite/wtmperrorMMDD file
STATE	Current state in /usr/adm/acct/nite/statefile file
fd2log	Any other error messages

Holidays file update

The Holidays file is out of date after the last holiday listed has passed or the year has changed. You can update the Holidays file.

The **acctcon1** command (started from the **runacct** command) sends mail to the **root** and **adm** accounts when the **/usr/lib/acct/holidays** file gets out of date.

Update the out-of-date Holidays file by editing the **/var/adm/acct/holidays** file to differentiate between prime and nonprime time.

Prime time is assumed to be the period when your system is most active, such as workdays. Saturdays and Sundays are always nonprime times for the accounting system, as are any holidays that you list.

The holidays file contains three types of entries: comments, the year and prime-time period, and a list of holidays as in the following example:

```
* Prime/Non-Prime Time Table for Accounting System
*
* Curr      Prime      Non-Prime
* Year      Start      Start
* 1992      0830      1700
*
* Day of    Calendar    Company
* Year      Date        Holiday
*
* 1         Jan 1         New Year's Day
* 20        Jan 20        Martin Luther King Day
* 46        Feb 15        President's Day
* 143       May 28        Memorial Day
* 186       Jul 3         4th of July
* 248       Sep 7         Labor Day
* 329       Nov 24        Thanksgiving
* 330       Nov 25        Friday after
* 359       Dec 24        Christmas Eve
* 360       Dec 25        Christmas Day
* 361       Dec 26        Day after Christmas
```

The first noncomment line must specify the current year (as four digits) and the beginning and end of prime time, also as four digits each. The concept of prime and nonprime time only affects the way that the accounting programs process the accounting records.

If the list of holidays is too long, the **acctcon1** command generates an error, and you will need to shorten your list. You are safe with 20 or fewer holidays. If you want to add more holidays, just edit the holidays file each month.

System data collection and reporting

You can set up the system to automatically collect data and generate reports.

For data to be collected automatically, a member of the adm group needs to follow the procedures described in "Setting up an accounting system" on page 245. These procedures enable the **cron** daemon to run the commands that generate data on:

- The amount of time each user spends logged in to the system
- Usage of the processing unit, memory, and I/O resources
- The amount of disk space occupied by each user's files
- Usage of printers and plotters
- The number of times a specific command is given.

The system writes a record of each session and process after they are completed. These records are converted into total accounting (**tacct**) records arranged by user and merged into a daily report. Periodically, the daily reports are combined to produce totals for the defined fiscal period. Methods for collecting and reporting the data and the various accounting commands and files are discussed in the following sections.

Although most of the accounting data is collected and processed automatically, a member of the **adm** group can enter certain commands from the keyboard to obtain specific information. These commands are discussed in “Keyboard commands” on page 264.

Connect-time accounting data

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. For information about the **init** and **login** commands, see **init** and **login**.

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. For information about the **utmp.h** file, see **utmp.h**.

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.”

Connect-time reports

Accounting records include login, logout, system-shutdown, and lastlogin records.

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(x)/lineuse** file. The **acctcon2** command then converts the session records into a total accounting record, **/var/adm/logacct**, that the **acctmrg** command adds to daily reports. For information about these commands, see **runacct**, **acctcon1**, and **acctcon2**.

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(x)/lineuse**. To produce an overall session report for the accounting period, **/var/adm/acct/nite(x)/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. For information about the **lastlogin** command, see **lastlogin**.

Process accounting data

The Accounting system collects data on resource usage for each process as it runs.

This data includes:

- The user and group numbers under which the process runs
- The first eight characters of the name of the command

- A 64-bit numeric key representing the Workload Manager class that the process belongs to
- The elapsed time and processor time used by the process
- Memory use
- The number of characters transferred
- The number of disk blocks read or written on behalf of the process

The **accton** command records these data in a specified file, usually the **/var/adm/pacct** file. For more information about the **accton** command, see **accton**.

Related commands are the **startup** command, the **shutacct** command, the **dodisk** command, the **ckpacct** command, and the **turnacct** command. For information about these commands, see **startup**, **shutacct**, **dodisk**, **ckpacct**, and **turnacct**.

For more information, see “Accounting data reports” on page 260.

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. For more information about the **acctmerg** command, see **acctmerg**.

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. For more information about the **acctcms** command, see **acctcms**.

The **acctcom** command handles the same data as the **acctcms** command, but provides detailed information about each process. You can display all process accounting records or select records of particular interest. Selection criteria include the load imposed by the process, the time period when the process ended, the name of the command, the user or group that invoked the process, the name of the WLM class the process belonged to, and the port at which the process ran. Unlike other accounting commands, **acctcom** can be run by all users. For more information about the **acctcom** command, see **acctcom**.

Disk-usage accounting data

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(x)/dacct** file.

To accomplish this, the **dodisk** command calls other commands. Depending upon the thoroughness of the accounting search, the **diskusg** command or the **acctdusg** command can be used to collect data. The **acctdisk** command is used to write a total accounting record. The total accounting record, in turn, is used by the **acctmerg** command to prepare the daily accounting report.

The **dodisk** command charges a user for the links to files found in the user’s login directory and evenly divides the charge for each file between the links. This distributes the cost of using a file over all who use it and removes the charges from users when they relinquish access to a file. For more information about the **dodisk** command and **cron** daemon, see **dodisk** and **cron**.

For more information, see “Connect-time reports” on page 257.

Disk-usage accounting report

The disk-usage records collected in the `/var/adm/acct/nite(x)/dacct` file are merged into the daily accounting reports by the `acctmerg` command.

For information about the `acctmerg` command, see `acctmerg`.

Printer-usage accounting data

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 about these commands, see `enq` and `qdaemon`.

For more information, see “Printer-Usage accounting report.”

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.

For information about the `acctmerg` command, see `acctmerg`.

Fee accounting data

You can produce an ASCII total accounting record in the `/var/adm/fee` file.

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 information about the `chargefee` and `acctmerg` commands, see `chargefee` and `acctmerg`.

For more information, see “Fee accounting report.”

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.

For information about the `chargefee` and `acctmerg` commands, see `chargefee` and `acctmerg`.

Fiscal accounting reports

The Fiscal Accounting Reports generally collected monthly by using the `monacct` command.

The report is stored in `/var/adm/acct/fiscal(x)/fiscrptMM` where `MM` is the month that the `monacct` command was executed. This report includes information similar to the daily reports summarized for the entire month.

Accounting system activity reports

You can generate a report that shows Accounting system activity.

To generate a report on system activity, use the **prtacct** command. This command reads the information in a total accounting file (**tacct** file format) and produces formatted output. Total accounting files include the daily reports on connect time, process time, disk usage, and printer usage.

Prerequisites

The **prtacct** command requires an input file in the **tacct** file format. This implies that you have an accounting system set up and running or that you have run the accounting system in the past. See “Setting up an accounting system” on page 245 for guidelines.

Procedure

Generate a report on system activity by entering:

```
prtacct -f Specification -v Heading File
```

Specification is a comma-separated list of field numbers or ranges used by the **acctmerg** command. The optional **-v** flag produces verbose output where floating-point numbers are displayed in higher precision notation. *Heading* is the title you want to appear on the report and is optional. *File* is the full path name of the total accounting file to use for input. You can specify more than one file.

Accounting data reports

After the various types of accounting data are collected, the records are processed and converted into reports.

Accounting commands automatically convert records into scientific notation when numbers become large. A number is represented in scientific notation in the following format:

*Base***e**+*Exp*

*Base***e**-*Exp*

which is the number equal to the *Base* number multiplied by 10 to the **+Exp** or **-Exp** power. For example, the scientific notation 1.345e+9 is equal to 1.345x10⁹, or 1,345,000,000. And the scientific notation 1.345e-9 is equal to 1.345x10⁻⁹ or, 0.000000001345.

Daily accounting reports

To generate a daily report, use the **runacct** command.

This command summarizes data into an ASCII file named **/var/adm/acct/sum(x)/rprtMMDD**. *MMDD* specifies the month and day the report is run. The report covers the following:

- Daily report
- Daily Usage report
- Daily Command Summary
- Monthly Total Command Summary
- Last Login

Daily report

Daily accounting reports contain data on connect-time, processes, disk usage, printer usage, and fees to charge.

The **acctmerg** command merges raw accounting data on connect-time, processes, disk usage, printer usage, and fees to charge into daily reports. Called by the **runacct** command as part of its daily operation, the **acctmerg** command produces the following:

/var/adm/acct/nite(x)/dacct

An intermediate report that is produced when one of the input files is full.

/var/adm/acct/sum(x)/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. For more information about the **acctmerg** command, see **acctmerg**.

The first line of the Daily report begins with the start and finish times for the data collected in the report, a list of system-level events including any existing shutdowns, reboots, and run-level changes. The total duration is also listed indicating the total number of minutes included within the accounting period (usually 1440 minutes, if the report is run every 24 hours). The report contains the following information:

LINE	Console, tty, or pty In use
MINUTES	Total number of minutes the line was in use
PERCENT	Percentage of time in the accounting period that the line was in use
# SESS	Number of new login sessions started
# ON	Same as # SESS
# OFF	Number of logouts plus interrupts made on the line

Daily Usage accounting report

The Daily Usage report is a summarized report of system usage per user ID during the accounting period.

Some fields are divided into prime and non-prime time, as defined by the accounting administrator in the **/usr/lib/acct/holidays** directory. The report contains the following information:

UID	User ID
LOGIN NAME	User name
CPU (PRIME/NPRIME)	Total CPU time for all of the user's processes in minutes
KCORE (PRIME/NPRIME)	Total memory used by running processes, in kilobyte-minutes
CONNECT (PRIME/NPRIME)	Total connect time (how long the user was logged in) in minutes
DISK BLOCKS	Average total amount of disk space used by the user on all filesystems for which accounting is enabled
FEES	Total fees entered with chargefee command
# OF PROCS	Total number of processes belonging to this user
# OF SESS	Number of distinct login sessions for this user
# DISK SAMPLES	Number of times disk samples were run during the accounting period. If no DISK BLOCKS are owned, the value will be zero

Daily Command Summary accounting report

The Daily Command Summary report shows each command executed during the accounting period, with one line per each unique command name.

The table is sorted by TOTAL KCOREMIN (described below), with the first line including the total information for all commands. The data listed for each command is cumulative for all executions of the command during the accounting period. The columns in this table include the following information:

COMMAND NAME	Command that was executed
NUMBER CMDS	Number of times the command executed
TOTAL KCOREMIN	Total memory used by running the command, in kilobyte-minutes
TOTAL CPU-MIN	Total CPU time used by the command in minutes
TOTAL REAL-MIN	Total real time elapsed for the command in minutes
MEAN SIZE-K	Mean size of memory used by the command per CPU minute
MEAN CPU-MIN	Mean number of CPU minutes per execution of the command
HOG FACTOR	Measurement of how much the command hogs the CPU while it is active. It is the ratio of TOTAL CPU-MIN over TOTAL REAL-MIN
CHARS TRNSFD	Number of characters transferred by the command with system reads and writes
BLOCKS READ	Number of physical block reads and writes performed by the command

Monthly Total Command Summary accounting report

The Monthly Total Command Summary, created by the **monacct** command, provides information about all commands executed since the previous monthly report.

The fields and information mean the same as those in the Daily Command Summary.

Last login

The Last Login report displays two fields for each user ID. The first field is **YY-MM-DD** and indicates the most recent login for the specified user. The second field is the name of the user account.

A date field of 00-00-00 indicates that the user ID has never logged in.

Accounting report summary

You can generate a report that summarizes raw accounting data.

To summarize raw accounting data, use the **sa** command. This command reads the raw accounting data, usually collected in the **/var/adm/pacct** file, and the current usage summary data in the **/var/adm/savacct** file, if summary data exists. It combines this information into a new usage summary report and purges the raw data file to make room for further data collection.

Prerequisites

The **sa** command requires an input file of raw accounting data such as the **pacct** file (process accounting file). To collect raw accounting data, you must have an accounting system set up and running. See “Setting up an accounting system” on page 245 for guidelines

Procedure

The purpose of the **sa** command is to summarize process accounting information and to display or store that information. The simplest use of the command displays a list of statistics about every process that has run during the life of the **pacct** file being read. To produce such a list, type:

```
/usr/sbin/sa
```

To summarize the accounting information and merge it into the summary file, type:

```
/usr/sbin/sa -s
```

The **sa** command offers many additional flags that specify how the accounting information is processed and displayed. See the **sa** command description for more information.

Monthly report

You can generate a Monthly accounting 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.

Greater than eight character username support

In order to maintain backwards compatibility with all scripts, long username support is not enabled by default within accounting. Instead, all user IDs are truncated to the first eight characters.

In order to enable long username support, most commands have been given the additional **-X** flag, which allows them to accept and output greater than eight-character user IDs (in both ASCII and binary formats). In addition, when long username support is enabled, commands and scripts will process files in the **/var/adm/acct/sumx**, **/var/adm/acct/nitex**, and **/var/adm/acct/fiscalx** directories, instead of using **/var/adm/acct/sum**, **/var/adm/acct/nite**, and **/var/adm/acct/fiscal**.

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. For information about these commands, see **runacct**, **cron** daemon, and **crontab**.
- 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 automatically collect accounting data.

Several commands usually run by the **cron** daemon automatically collect accounting data. These commands are:

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.

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/sadd` file, where *dd* is the day of the month. The command removes reports from the `/var/adm/sa/sadd` file that have been there longer than one week.

Other commands are run automatically by procedures other than the `cron` daemon:

startup

When added to the `/etc/rc` file, the `startup` command initiates startup procedures for the accounting system.

shutacct

Records the time accounting was turned off by calling the `acctwtmp` command to write a line to `/var/adm/wtmp` file. It then calls the `turnacct off` command to turn off process accounting.

Keyboard commands

A member of the `adm` group can enter the following commands from the keyboard.

ac Prints connect-time records. This command is provided for compatibility with Berkeley Software Distribution (BSD) systems.

acctcom

Displays process accounting summaries. This command is also available to users.

acctcon1

Displays connect-time summaries. Either the `-l` flag or the `-o` flag must be used.

accton

Turns process accounting on and off.

chargefee

Charges the user a predetermined fee for units of work performed. The charges are added to the daily report by the `acctmerg` command.

fwtmp Converts files between binary and ASCII formats.

last Displays information about previous logins. This command is provided for compatibility with BSD systems.

lastcomm

Displays information about the last commands that were executed. This command is provided for compatibility with BSD systems.

lastlogin

Displays the time each user last logged in.

pac

Prepares printer/plotter accounting records. This command is provided for compatibility with BSD systems.

prctmp

Displays a session record.

prtacct

Displays total accounting files.

sa

Summarizes raw accounting information to help manage large volumes of accounting information. This command is provided for compatibility with BSD systems.

sadc

Reports on various local system actions, such as buffer usage, disk and tape I/O activity, TTY device activity counters, and file access counters.

sar

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

time

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

timex

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

Accounting files

The two main accounting directories are the **/usr/sbin/acct** directory, where all the C language programs and shell procedures needed to run the accounting system are stored, and the **/var/adm** directory, which contains the data, report and summary files.

The accounting data files belong to members of the **adm** group, and all active data files (such as **wtmp** and **pacct**) reside in the **adm** home directory **/var/adm**.

Accounting data files

The following files are in the **/var/adm** directory.

/var/adm/diskdiag	Diagnostic output during the running of disk accounting programs
/var/adm/dtmp	Output from the acctdusg command
/var/adm/fee	Output from the chargefee command, in ASCII tacct records
/var/adm/pacct	Active process accounting file
/var/adm/wtmp	Active process accounting file
/var/adm/Spacct .mmd	Process accounting files for mmd during the execution of the runacct command.

Accounting report and summary files

Some subdirectories are needed before you enable the Accounting system.

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" on page 245 for more information.

/var/adm/acct/nite(x)

Contains files that the **runacct** command reuses daily

/var/adm/acct/sum(x)

Contains the cumulative summary files that the **runacct** command updates daily

/var/adm/acct/fiscal(x)

Contains the monthly summary files that the **monacct** command creates.

Starting the runacct command for accounting

You can start the **runacct** command.

Prerequisites

1. You must have the accounting system installed.
2. You must have root user or adm group authority.

Notes:

1. If you call the **runacct** command with no parameters, the command assumes that this is the first time that the command has been run today. Therefore, you need to include the *mddd* parameter when you restart the **runacct** program, so that the month and day are correct. If you do not specify a state, the **runacct** program reads the **/var/adm/acct/nite(x)/statefile** file to determine the entry point for processing. To override the **/var/adm/acct/nite(x)/statefile** file, specify the desired state on the command line.
2. When you perform the following task, you might need to use the full path name **/usr/sbin/acct/runacct** rather than the simple command name, **runacct**.

Procedure

To start the **runacct** command, type the following:

```
nohup runacct 2> \  
/var/adm/acct/nite/accterr &
```

This entry causes the command to ignore all **INTR** and **QUIT** signals while it performs background processing. It redirects all standard error output to the **/var/adm/acct/nite/accterr** file.

Restarting the runacct command for Accounting

If the **runacct** command is unsuccessful, you can restart it.

The prerequisites for this procedure are:

- You must have the accounting system installed.
- You must have root user or adm group authority.

Note: The most common reason why the **runacct** command can fail are because:

- The system goes down.
- The **/usr** file system runs out of space.
- The **/var/adm/wtmp** file has records with inconsistent date stamps.

If the **runacct** command is unsuccessful, do the following:

1. Check the **/var/adm/acct/nite(x)/active mddd** file for error messages.
2. If both the active file and lock files exist in **acct/nite**, check the **accterr** file, where error messages are redirected when the **cron** daemon calls the **runacct** command.
3. Perform any actions needed to eliminate errors.
4. Restart the **runacct** command.
5. To restart the **runacct** command for a specific date, type the following:

```
nohup runacct 0601 2>> \  
/var/adm/acct/nite/accterr &
```

This restarts the **runacct** program for June 1 (0601). The **runacct** program reads the **/var/adm/acct/nite/statefile** file to find out with which state to begin. All standard error output is appended to the **/var/adm/acct/nite/accterr** file.

6. To restart the **runacct** program at a specified state, for example, the MERGE state, type the following:

```
nohup runacct 0601 MERGE 2>> \  
/var/adm/acct/nite/accterr &
```

runacct command files

The **runacct** command produces report and summary files.

The following report and summary files, produced by the **runacct** command, are of particular interest:

/var/adm/acct/nite(x)/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(x)/dayacct	Contains the total accounting file for the previous day.
/var/adm/acct/sum(x)/tacct	Contains the accumulation of each day's nite/dayacct file and can be used for billing purposes. The monacct command restarts the file each month or fiscal period.
/var/adm/acct/sum(x)/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(x)/daycms	Contains the daily command summary. An ASCII version is stored in nite/daycms .
/var/adm/acct/sum(x)/loginlog	Contains a record of the last time each user ID was used.
/var/adm/acct/sum(x)/rprrt mmd	This file contains a copy of the daily report saved by the runacct command.

Files in the /var/adm/acct/nite(x) directory

The following files are in the **/var/adm/acct/nite(x)** directory.

active	Used by the runacct command to record progress and print warning and error messages. The file active.mmd 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.mmd	Connect total accounting records.
ctmp	Connect session records.
daycms	ASCII daily command summary used by the prdaily command.
dayacct	Total accounting records for one day.
dacct	Disk total accounting records, created by the ddisk command.
accterr	Diagnostic output produced during the execution of the runacct command.
lastdate	Last day the runacct executed, in <i>date +%m%d</i> 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.
logmmd	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.
wtmperrmmd	Same as wtmperror after the runacct command detects an error.
wtmp.mmd	Contains previous day's wtmp file. Removed during the cleanup of runacct command.

Files in the /var/adm/acct/sum(x) directory

The following files are in the **/var/adm/acct/sum(x)** 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 <i>mmdd</i> . This file is removed after system startup by the remove command. For information about the remove command, see remove .
rprtmmdd	Saved output of the prdaily command.
tacct	Cumulative total accounting file for the current fiscal period.
tacctprev	Same as tacct without the latest update.
tacctmmdd	Total accounting file for <i>mmdd</i> .

Files in the `/var/adm/acct/fiscal(x)` directory

The following files are in the `/var/adm/acct/fiscal(x)` directory.

cms?	Total command summary file for the fiscal period, specified by <i>?</i> , in binary format
fiscrpt?	A report similar to that of the prdaily command for fiscal period, specified by <i>?</i> , in binary format
tacct?	Total accounting file for fiscal period, specified by <i>?</i> , in binary format.

Accounting file formats

The following table summarizes the accounting file output and formats.

wtmp	Produces the active process accounting file. The format of the wtmp file is defined in the utmp.h file. For information about the utmp.h file, see utmp.h .
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 <i>mmdd</i> during the running of the runacct command. The format of these files is defined in the sys/acct.h file.
daytacct	Produces total accounting records for one day. The format of the file is defined in the tacct file format.
sum/tacct	Produces binary file that accumulates each day's command summaries. The format of this file is defined in the /usr/include/sys/acct.h header file.
ptacct	Produces concatenated versions of pacct files. The format of these files are defined in the tacct file.
ctacct	Produces connect total accounting records. The output of this file is defined in the tacct file.
cms	Produces total accounting command summary used by the prdaily command, in binary format. The ASCII version is nite/cms .
daycms	Daily command summary used by the prdaily command, in binary format. The ASCII version is nite/daycms .

Chapter 12. Web-based System Manager

Web-based System Manager is a client-server application that gives the user a powerful graphical user interface to access and manage multiple hosts.

Through Web-based System Manager, you can view users and groups, install software, and printers and devices; manage logical volumes, users and groups, and resources; mount and unmount file systems; configuring the network; and do many other system administration tasks. A plug-in architecture makes it easier to extend the suite. In addition, Web-based System Manager supports dynamic monitoring and administrator notification of system events.

The interface provides point-and-click control of objects, which provides an alternative to learning and using commands or SMIT. You can select a machine to manage, and as you navigate through your desired operation, the panels refresh to display your currently allowable choices.

For a detailed explanation and procedures for using Web-based System Manager, *AIX 5L Version 5.3 Web-based System Manager Administration Guide*.

Chapter 13. System management interface tool

Although the Web-based System Manager is the primary interface for system management, the System Management Interface Tool (SMIT) provides an alternative, natural-language, task-oriented interface. The SMIT facility runs in two interfaces, ASCII (nongraphical) or AIXwindows (graphical).

SMIT steps you through the desired task with the use of menus, selectors, and dialogs, thereby freeing you from the details of complex command syntax, valid parameter values, and system command spelling. In addition, SMIT creates log files that you can use to duplicate system configuration or to learn how to use specific commands.

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

<i>Task</i>	<i>SMIT Fast Path</i>	<i>Selection (ASCII)</i>	<i>Selection (AIXwindows)</i>
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 14. Preparing to install a device

Installing devices on your system consists of identifying where the device is to be attached, connecting the device physically, and configuring the device with Web-based System Manager, the Configuration Manager, or SMIT.

Note: The following procedure requires a shutdown of your system to install the device. Not all device installations require a shutdown of your system. Refer to the documentation shipped with the specific device.

This topic documents installation tasks that are common to all devices. Because of the wide variety of devices that you can install on your system, only a general procedure is provided. For more specific information, see the installation instructions shipped with the specific device.

1. Stop all applications running on the system unit and shut down the system unit using the **shutdown** command.
2. Turn off the system unit and all attached devices.
3. Unplug the system unit and all attached devices.
4. Connect the new device to the system using the procedure described in the setup and operator guide for the device.
5. Plug in the system unit and all attached devices.
6. Turn on all the attached devices leaving the system unit turned off.
7. Turn on the system unit when all the devices complete power-on self-tests (POST).

The Configuration Manager automatically scans the attached devices and configures any new devices it detects. The new devices are configured with default attributes and recorded in the customized configuration database placing the device in **Available** state.

You can manually configure a device using Web-based System Manager (**wsm**, then select **Devices**), or the SMIT fast path, **smit dev**. If you need to customize the device attributes or if the device cannot be configured automatically, see the device documentation that shipped with the device for specific configuration requirements.

Chapter 15. Installing an IDE device

You can install an IDE device on your system. The procedure to install an IDE device is divided into several tasks that must be performed in order.

The prerequisites for installing an IDE device are:

- You must have access to the operator's guide for your system unit and the installation guide for the device to be installed. The documentation must identify how to set the IDE device jumper to configure the device to either the master or slave setting.
- There must be at least one unused IDE device ID on an IDE adapter on the system.
- If you are updating the product topology diskettes, you need the Product Topology System diskette which is kept with important records for the system, and the Product Topology Update diskette which is shipped with the device.
- Verify that the interface of the device is compatible with the interface of the IDE controllers on the system unit.
- There are two classifications for IDE devices, ATA and ATAPI. ATA are disk devices and ATAPI are CD-ROM or tape devices. Up to two devices are allowed to be connected to each IDE controller, one master and one slave. Typically an IDE adapter has two controllers, which allows up to four IDE devices to be attached.

With appropriate cabling, you can attach any of the following device combinations to a single controller:

- 1 ATA device as master
- 1 ATAPI device as master
- 2 ATA devices as master and slave
- 1 ATA device as master and 1 ATAPI device as slave
- 2 ATAPI devices as master and slave

You cannot attach the following:

- 1 ATA device as slave only
- 1 ATAPI device as slave only
- 1 ATAPI device as master and 1 ATA device as slave

Determining the number and location of IDE controllers

You can determine how many IDE controllers are attached to your system unit and where the IDE controllers are located. An IDE adapter may be in an adapter slot or built into the system planar. Remember that IDE adapters have two IDE controllers (IDE buses). Thus, two IDE controllers are found in an adapter slot or built into the system planar.

This topic is part of a series of tasks that allow you to install an IDE device. These tasks are:

1. The prerequisites for this series of tasks is Chapter 15, "Installing an IDE device"
2. Task 1 is this topic ("Determining the number and location of IDE controllers")
3. Task 2 is "Selecting an IDE controller and an IDE address on the controller" on page 276
4. Task 3 is "Setting up the hardware for IDE device installation" on page 278
5. Task 4 is "Adding an IDE device to the customized configuration database" on page 278
6. Task 5 is "Customizing the attributes for an IDE device" on page 279. Task 5 is optional.

You can obtain the number and location of IDE controllers using either of the following two methods:

- Using a software configuration command. This method is available only when the operating system has been installed on the system unit.

- Using the *About Your Machine* document shipped with your system unit. This method is valid only for initial setup and installation of a new system unit.

Using a software configuration command

This method applies to a system that already has the operating system installed.

To list the IDE I/O controllers on the system, type the following commands:

```
lscfg -l ide*
```

Examine the list of IDE controllers that are displayed. The following sample display from the **lscfg -l ide** command shows two IDE I/O controllers. Controller **ide0** and **ide1** are located on the system planar. The planar indicator is the second digit in the location value with a value of 1.

DEVICE	LOCATION	DESCRIPTION
ide0	01-00-00	ATA/IDE Controller Device
ide1	01-00-01	ATA/IDE Controller Device

2nd digit is the adapter slot number | 6th digit indicates the controller number.

Initial setup

Use the *About Your Machine* document to determine the IDE I/O controllers on the system if the device is being installed during initial setup.

Note: Incorrect results are produced if controllers have been added after the system was shipped from the factory.

Determine whether the system unit has an IDE controller built into the planar board. A built-in IDE I/O controller is standard on some system units. Your system unit has a built-in IDE controller if *About Your Machine* document shows an internal media IDE device with a blank slot number.

Selecting an IDE controller and an IDE address on the controller

You can select an IDE controller and an IDE address on the controller.

This topic is part of a series of tasks that allow you to install an IDE device. These tasks are:

1. The prerequisites for this series of tasks is Chapter 15, “Installing an IDE device,” on page 275
2. Task 1 is “Determining the number and location of IDE controllers” on page 275
3. Task 2 is this topic (“Selecting an IDE controller and an IDE address on the controller”)
4. Task 3 is “Setting up the hardware for IDE device installation” on page 278
5. Task 4 is “Adding an IDE device to the customized configuration database” on page 278
6. Task 5 is “Customizing the attributes for an IDE device” on page 279. Task 5 is optional.

After identifying the IDE controllers attached to the system unit, select the IDE I/O controller to which you want to connect a device. This IDE I/O controller must have at least one IDE setting that is not already assigned to another device.

Determine whether IDE device setting must be jumpered as master or slave. If no device is currently attached to the controller, the IDE device jumper must be set to master (some devices require no device ID setting in this situation). If an IDE device is already attached, the type of device must be determined.

Disks are ATA devices. CD-ROM and tape are ATAPI devices. If ATA and ATAPI devices are both attached to the same IDE controller, the ATA device must be set to master ID and the ATAPI device must be set to slave ID.

Determine what IDE devices are attached to a controller by viewing information about the devices already connected to the IDE controllers.

You can use two methods to select an IDE I/O controller and an IDE address on the controller that is not already assigned to another device:

- Using a software configuration command if the operating system is already installed on the system unit.
- Using the *About Your Machine* document for initial setup and installation of a new system unit.

Using a software configuration command

This method applies to a system that already has the operating system installed.

1. Type the following command to list all the currently defined IDE devices:

```
lsdev -C -s ide -H
```

2. Examine the list of devices already assigned to each IDE controller. Each row in this display shows the logical name, status, location, and description of an IDE device. The location for each device begins with the location of the controller to which the device is connected. In the sample below, the IDE I/O controller with address 01-00-00 has two IDE devices attached. The IDE I/O controller with location 01-00-01 has one IDE device attached.

name	status	location	description
hdisk0	Available	01-00-00-00	720 MB IDE Disk Drive
hdisk1	Available	01-00-00-01	540 MB IDE Disk Drive
cd0	Available	01-00-01-00	IDE CD-ROM Drive

|
IDE controller address (6th digit)

3. Select a controller that does not have two IDE devices already connected.
4. If one device is already attached to the controller, determine the type of the device. Also determine the type of device to be installed. Disk devices are classified as ATA devices. CD-ROM and tape devices are classified as ATAPI devices.
5. Determine the IDE jumper setting for the new device depending upon the combination of devices to be connected to the IDE controller. If the new device is the only device connected to the controller, the device jumper setting must be set to the master position (some devices require no setting in this case). If both devices are the same type, the new device jumper setting can be set to the slave position. If there is a mix of devices (ATA and ATAPI), the ATA device jumper must be set to the master position and the ATAPI device jumper must be set to the slave position. If there is a mix of devices and the new device is an ATA device (disk), the device jumper for the currently existing ATAPI device must be changed to the slave position and the new ATA device jumper must be set to master. If there is a mix of devices and the new device is an ATAPI device (CD-ROM or tape), the device jumper for the new ATAPI device must be set to slave and if the ATA device does not currently have a jumper setting, it must be set to master.

Initial setup

Use the *About Your Machine* document to determine the devices assigned to the IDE I/O controllers on the system if the device is being installed during initial setup.

Note: Incorrect results are produced if controllers have been added after the system was shipped from the factory.

1. To determine the IDE devices assigned to addresses on the IDE controllers, see "Internal Media Devices" in *About Your Machine*.
2. Select a controller that does not have two IDE devices already connected.

3. If one device is already attached to the controller, determine the type of the device. Also determine the type of device to be installed. Disk devices are classified as ATA devices. CD-ROM and tape devices are classified as ATAPI devices.
4. Determine the IDE jumper setting for the new device depending upon the combination of devices to be connected to the IDE controller. If the new device will be the only device connected to the controller, the device jumper setting must be set to the master position (some devices require no setting in this case). If both devices are the same type, the new device jumper setting can be set to the slave position. If there is a mix of devices (ATA and ATAPI), the ATA device jumper must be set to the master position and the ATAPI device jumper must be set to the slave position. If there is a mix of devices and the new device is an ATA device (disk), the device jumper for the currently existing ATAPI device must be changed to the slave position and the new ATA device jumper must be set to master. If there is a mix of devices and the new device is an ATAPI device (CD-ROM or tape), the device jumper for the new ATAPI device must be set to slave and if the ATA device does not currently have a jumper setting, it must be set to master.

Setting up the hardware for IDE device installation

You must set up the hardware in order to install an IDE device.

- Do not begin this task until you have selected and recorded the following:
 - Position of the IDE I/O controller where the device will be connected (either built-in or identified by an adapter slot number).
 - IDE address for the device.
- Determine the physical position on the system unit to connect the selected IDE controller. For example, locate the position of the built-in IDE controller. Refer to the operator's guide for help.

This topic is part of a series of tasks that allow you to install an IDE device. These tasks are:

1. The prerequisites for this series of tasks is Chapter 15, "Installing an IDE device," on page 275
2. Task 1 is "Determining the number and location of IDE controllers" on page 275
3. Task 2 is "Selecting an IDE controller and an IDE address on the controller" on page 276
4. Task 3 is this topic ("Setting up the hardware for IDE device installation")
5. Task 4 is "Adding an IDE device to the customized configuration database"
6. Task 5 is "Customizing the attributes for an IDE device" on page 279. Task 5 is optional.

Procedure

1. Shut down the system unit using the **shutdown** command after stopping all applications that are currently running. Type `shutdown -F` to stop the system immediately without notifying other users.
2. Wait for the message `Ha1t Completed` or a similar message to be displayed.
3. Turn off the system unit and all attached devices.
4. Unplug the system unit and all attached devices.
5. Make the physical connections following the procedure described in the setup and operator guide.

Note: Do not power on the system unit; proceed to "Adding an IDE device to the customized configuration database."

Adding an IDE device to the customized configuration database

This task makes the device known to the system. During system unit startup, the operating system reads the current configuration and detects new devices. A record of each new device is added to the customized configuration database and are given default attributes.

If the device is being installed on a new system unit, the operating system must be installed. Instructions for installing the operating system are included in the installation guide for the operating system.

This topic is part of a series of tasks that allow you to install an IDE device. These tasks are:

1. The prerequisites for this series of tasks is Chapter 15, “Installing an IDE device,” on page 275
2. Task 1 is “Determining the number and location of IDE controllers” on page 275
3. Task 2 is “Selecting an IDE controller and an IDE address on the controller” on page 276
4. Task 3 is “Setting up the hardware for IDE device installation” on page 278
5. Task 4 is this topic (“Adding an IDE device to the customized configuration database” on page 278)
6. Task 5 is “Customizing the attributes for an IDE device.” Task 5 is optional.

Follow this procedure to add a device to the customized configuration database:

1. Plug in the system unit and all attached devices.
2. Turn on all the devices, but leave the system unit turned off.
3. Turn on the system unit when all the attached devices have completed power-on self-tests (POSTs).

Note: The startup process automatically detects and records the device in the customized configuration database.

4. Confirm that the device was added to the customized configuration database using the Web-based System Manager (type `wsm`), or the SMIT fast path, **smit lsidea**. A list of all defined devices is displayed. Look at the location field for the IDE adapter and IDE address values of the device you just installed.

Customizing the attributes for an IDE device

Default attributes are assigned to a supported device when it is added to the customized configuration database. These attributes are appropriate for typical use of the device.

This topic is part of a series of tasks that allow you to install an IDE device. These tasks are:

1. The prerequisites for this series of tasks is Chapter 15, “Installing an IDE device,” on page 275
2. Task 1 is “Determining the number and location of IDE controllers” on page 275
3. Task 2 is “Selecting an IDE controller and an IDE address on the controller” on page 276
4. Task 3 is “Setting up the hardware for IDE device installation” on page 278
5. Task 4 is “Adding an IDE device to the customized configuration database” on page 278
6. Task 5 is this topic (“Customizing the attributes for an IDE device”). Task 5 is optional.

Change the device attributes when the device you are installing is not supported or when you need to customize some part of the device’s operation. For example, you might need to change your tape drive to write tapes in a lower-density format.

To customize the attributes for a device use the SMIT fast path, **smit dev**.

Chapter 16. Configuring a read/write optical drive

There are two methods for configuring a read/write optical drive.

The read/write optical drive must be connected to the system and powered on.

Method 1

Method one is the faster of the two methods. It only configures the read/write optical drive specified. To use this method, you must provide the following information:

Subclass	Defines how the drive is attached.
Type	Specifies the type of read/write optical drive.
Parent Name	Specifies the system attachment the drive is connected to.
Where Connected	Specifies the logical address of the drive.

Enter the following command to configure the read/write optical drive:

```
mkdev -c rwoptical -s Subclass -t Type -p ParentName -w WhereConnected
```

The following is an example of a read/write optical drive that has a SCSI ID of 6, a logical unit number of zero, and is connected to the third (scsi3) SCSI bus:

```
mkdev -c rwoptical -s scsi -t osomd -p scsi3 -w 6,0 -a pv=yes
```

Method 2

Method two uses the Configuration Manager, searching the current configuration, detecting any new devices, and automatically configuring the devices. This method is used when little information is known about the read/write optical drive.

1. Use the configuration manager to configure all newly detected devices on the system (including the read/write optical drive) by typing:

```
cfgmgr
```

2. Type the following command to list the names, location codes, and types of all currently configured read/write optical drives:

```
lsdev -C -c rwoptical
```

3. Determine the name of the newly configured read/write optical drive using the location code that matches the location of the drive being added.

Chapter 17. Configuration of a large number of 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. Device drivers are located in the `/usr/lib/drivers` directory.

The number of devices that AIX can support can vary from system to system, depending on several important factors. The following factors have an impact on the file systems that support the devices:

- Configuring a large number of devices requires storage of more information in the ODM device-configuration database. It can also require more device special files. As a result, more space and more i-nodes are required of the file system.
- Some devices require more space than others in the ODM device-configuration database. The number of special files or i-nodes used also varies from device to device. As a result, the amount of space and i-nodes required of the file system depends on the types of devices on the system.
- Multipath I/O (MPIO) devices require more space than non-MPIO devices because information is stored in the ODM for the device itself as well as for each path to the device. As a rough guideline, assume that each path takes up the space of one-fifth of a device. For example, an MPIO device with five paths will have the space equivalent to two non-MPIO devices.
- AIX includes both logical devices and physical devices in the ODM device-configuration database. Logical devices include volume groups, logical volumes, network interfaces, and so on. In some cases, the relationship between logical and physical devices can greatly affect the total number of devices supported. For example, if you define a volume group with two logical volumes for each physical disk that is attached to a system, this will result in four AIX devices for each disk. On the other hand, if you define a volume group with six logical volumes for each physical disk, there will be eight AIX devices for each disk. Therefore, only half as many disks could be attached.
- Changing device attributes from their default settings results in a larger ODM device-configuration database and could lead to fewer devices that can be supported.
- More devices require more real memory.

Two file systems are used by AIX to support devices:

- The RAM file system is used during boot in an environment that has no paging space and no disk file systems mounted. The size of the RAM file system is 25% of the system memory size up to a maximum of 128 MB. One i-node is allocated for every KB in the RAM file system. The minimum system memory requirement for AIX 5.2 is 128 MB, which translates into a minimum RAM file system size of 32 MB with 32768 i-nodes. If the system memory size is 512 MB or larger, then the RAM file system will be at its maximum size of 128 MB with 131072 i-nodes. If either the amount of RAM file system space or number of i-nodes needed to support the attached devices exceeds what has been allocated to the RAM disk, the system might not boot. If this is the case, you must remove some of the devices.
- The space and i-nodes of the root file system (rootvg) on the disk can be increased as long as there are unallocated partitions in the rootvg. With AIX 5.2 and the minimum RAM file system size, it is likely that up to 5000 AIX devices can be configured. With the maximum RAM file system size, it is likely that up to 25,000 AIX devices could be configured. These numbers include both physical and logical devices. Depending on the various factors mentioned in this section, your system might be able to configure more or fewer devices than this number.

Note:

1. With a large number of devices in the system, the longer configuration time contributes to a longer boot time.
2. AIX 5L Version 5.2 with the 5200-03 Recommended Maintenance package and newer support IDE DVD-RAM devices.

Chapter 18. Device nodes

Devices are organized into clusters known as *nodes*. Each node is a logical subsystem of devices, where lower-level devices have a dependency on upper-level devices in child-parent relationships.

For example, the system node is the highest of all nodes and consists of all the physical devices in the system. The system device is the top of the node and below that are the bus and adapters that have a dependency on the higher-level system device. At the bottom of the hierarchy are devices to which no other devices are connected. These devices have dependencies on all devices above them in the hierarchy.

At startup time, parent-child dependencies are used to configure all devices that make up a node. Configuration occurs from the top node down and any device having a dependency on a higher-level device is not configured until the higher-level device is configured.

Multiple-path I/O (MPIO) is a feature available with AIX 5.2 and later. If a device has an MPIO-capable device driver, it can have more than one parent within this hierarchy, which allows multiple, simultaneous communication paths between the device and a given machine or logical partition within a machine.

Device classes

Managing devices requires the operating system to comprehend what device connections are allowed. The operating system classifies devices hierarchically into three groups.

These groups are:

- Functional classes
- Functional subclasses
- Device types

Functional classes consist of devices that perform the same function. Printers, for example, comprise a functional class. Functional classes are grouped into subclasses according to certain device similarities. For example, printers have a serial or parallel interface. Serial printers are one subclass and parallel printers are another. Device types are classified according to their model and manufacturer.

Device classes define valid parent-child connections for the operating system. The hierarchy defines the possible subclasses that can be connected for each of the possible child connection locations. For example, the term RS-232 8-port adapter specifies that only devices belonging to the RS-232 subclass can be connected to any of the eight ports of the adapter.

Device classes and their hierarchical dependencies are maintained in an Object Data Manager (ODM) Device Configuration database.

Device configuration database and device management

Device information is contained in a predefined database or a customized database that makes up the device configuration database.

The predefined database contains configuration data for all possible devices supported by the system. The hierarchical device class information is contained in this database. The customized database contains configuration data for all currently defined and configured devices in the system. A record is kept of each device currently connected to your system.

The Configuration Manager is a program that automatically configures devices on your system during system startup and run time. The Configuration Manger uses the information from the predefined and customized databases during this process, and updates the customized database afterwards.

Beginning with AIX 5.2, the Multiple-path I/O (MPIO) feature uses a unique device identifier (UDID) to identify each MPIO-capable device, regardless of the path on which it was discovered. The UDID is saved in the device configuration database. When a device is discovered, the UDIDs in the database are checked to determine whether the device is new or whether the discovery is another path to an existing device. When multiple paths to a device are detected, the device driver or the Path Control Manager kernel extension decides which path to use for a particular request. For information on managing MPIO-capable devices, see “MPIO-capable device management” on page 304.

You can use the Web-based System Manager Devices application, SMIT, or operating system commands to perform device management tasks such as deleting, adding, or configuring a device.

Device states

Devices that are connected to the system can be in one of four states.

Devices that are connected to the system can be in one of the following 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.

Chapter 19. Adding a removable media drive

You can add a removable media drive.

The following procedure uses SMIT to add a CD-ROM drive to your system. Other types of removable media drives are added using different fast paths but all follow the same general procedure. You can also add a removable media drive using Web-based System Manager, the Configuration Manager, or the **mkdev** command.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. To add a CD-ROM drive to your system, install the hardware according to the documentation that came with your system.
2. With root authority, type the following SMIT fast path:
`smit makcdr`
3. In the next screen, select the drive type from the available list of supported drives.
4. In the next screen, select the parent adapter from the available list.
5. In the next screen, at minimum, select the connection address from the available list. You can also use this screen to select other options. When you are finished, press Enter, and then SMIT adds the new CD-ROM drive.

At this point, the new CD-ROM drive is recognized by your system. To add a read/write optical drive, use the **smit makomd** fast path. To add a tape drive, use the **smit maktpe** fast path.

See the **mkdev** command description in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for more information.

Chapter 20. Device location codes

The *location code* is a path from the CPU drawer or system unit through the adapter, signal cables, and the asynchronous distribution box (if there is one) to the device or workstation. This code is another way of identifying physical devices.

The location code consists of up to four fields of information depending on the type of device. These fields represent drawer, slot, connector, and port. Each of these fields consists of two characters.

The location code of a drawer consists of only the drawer field and is simply a two-character code. The location code of an adapter consists of the drawer and slot fields and has the format *AA-BB*, where *AA* corresponds to the drawer location and *BB* indicates the bus and slot that contains the adapter. Other devices have location codes of formats *AA-BB-CC* or *AA-BB-CC-DD*, where *AA-BB* is the location code of the adapter to which the device is connected, *CC* corresponds to the connector 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.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

Adapter location codes

The location code for an adapter consists of two pairs of digits with the format *AA-BB*, where *AA* identifies the location code of the drawer containing the adapter and *BB* identifies the I/O bus and slot containing the card.

A value of 00 for the **AA** field means that the adapter is located in the CPU drawer or system unit, depending on the type of system. Any other value for the **AA** field indicates that the card is located in an I/O expansion drawer. In this case, the *AA* value identifies the I/O bus and slot number in the CPU drawer that contains the asynchronous expansion adapter. The first digit identifies the I/O bus with 0 corresponding to the standard I/O bus and 1 corresponding to the optional I/O bus. The second digit identifies the slot number on the indicated I/O bus.

The first digit of the **BB** field identifies the I/O board containing the adapter card. If the card is in the CPU drawer or system unit, this digit is 0 for the standard I/O bus and 1 for the optional I/O bus. If the card is in an I/O expansion drawer, this digit is 0. The second digit identifies the slot number on the indicated I/O bus (or slot number in the I/O expansion drawer) that contains the card.

A location code of 00-00 is used to identify the standard I/O board.

Examples:

- 00-05 Identifies an adapter card in slot 5 of the standard I/O board and is located in either the CPU drawer or the system unit, depending on the type of system.
- 00-12 Identifies an adapter in slot 2 of the optional I/O bus and is located in the CPU drawer.
- 18-05 Identifies an adapter card located in slot 5 of an I/O expansion drawer. The drawer is connected to the asynchronous expansion adapter located in slot 8 of the optional I/O bus in the CPU drawer.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

Printer and plotter location codes

Location codes of 00-00-S1-00 or 00-00-S2-00 indicate the printer, plotter, or tty device is connected to the standard I/O board serial ports s1 or s2. A location code of 00-00-0P-00 indicates the parallel printer is connected to the standard I/O board parallel port.

Any other location code indicates that the printer, plotter, or tty device is connected to an adapter card other than the standard I/O board. For these printers, plotters, and tty devices, the location code format is *AA-BB-CC-DD*, where *AA-BB* indicates the location code of the controlling adapter.

- AA** A value of 00 for the **AA** field indicates the adapter card is located in the CPU drawer or system unit depending on the type of system. Any other value for the **AA** field indicates the card is located in an I/O expansion drawer; in which case, the first digit identifies the I/O bus and the second digit identifies the slot number on the bus, in the CPU drawer, that contains the asynchronous expansion adapter to which the I/O expansion drawer is connected.
- BB** The first digit of the **BB** field identifies the I/O bus containing the adapter card. If the card is in the CPU drawer or system unit, this digit is 0 for the standard I/O bus and 1 for the optional I/O bus. If the card is in an I/O expansion drawer, this digit is 0. The second digit identifies the slot number on the I/O bus (or slot number in the I/O expansion drawer) that contains the card.
- CC** The **CC** field identifies the connector on the adapter card where the asynchronous distribution box is connected. Possible values are 01, 02, 03, and 04.
- DD** The **DD** field identifies the port number on the asynchronous distribution box where the printer, plotter, or tty device is attached.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

tty location codes

Location codes of 00-00-S1-00 or 00-00-S2-00 indicate the tty device is connected to the standard I/O serial ports s1 or s2.

Any other location code indicates the tty device is connected to an adapter card other than the standard I/O board. For these devices, the location code format is *AA-BB-CC-DD*, where *AA-BB* indicates the location code of the controlling adapter card.

- AA** A value of 00 for the **AA** field indicates the adapter card is located in the CPU drawer or system unit depending on the type of system. Any other value for the **AA** field indicates the card is located in an I/O expansion drawer. In this case, the first digit identifies the I/O bus, and the second digit identifies the slot number on the bus in the CPU drawer that contains the asynchronous expansion adapter where the I/O expansion drawer is connected.
- BB** The first digit of the **BB** field identifies the I/O bus containing the adapter card. If the card is in the CPU drawer or system unit, this digit will be 0 for the standard I/O bus and 1 for the optional I/O bus. If the card is in an I/O expansion drawer this digit is 0. The second digit identifies the slot number on the I/O bus (or slot number in the I/O expansion drawer) that contains the card.
- CC** The **CC** field identifies the connector on the adapter card where the asynchronous distribution box is connected. Possible values are 01, 02, 03, and 04.
- DD** The **DD** field identifies the port number on the asynchronous distribution box where the tty device is attached.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

SCSI device location codes

The following are location codes for SCSI devices.

These location codes apply to all SCSI devices including:

- CD-ROMs
- Disks
- Initiator devices
- Read/write optical drives
- Tapes
- Target mode

The location code format is AA-BB-CC-S,L. The **AA-BB** fields identify the location code of the SCSI adapter controlling the SCSI device.

- AA** A value of 00 for the **AA** field indicates the controlling adapter card is located in the CPU drawer or system unit, depending on the type of system.
- BB** The **BB** field identifies the I/O bus and slot containing the card. The first digit identifies the I/O bus. It is 0 for the standard I/O bus and 1 for the optional I/O bus. The second digit is the slot on the indicated I/O bus containing the card. A value of 00 for the **BB** field indicates the standard SCSI controller.
- CC** The **CC** field identifies the SCSI bus of the card that the device is attached to. For a card that provides only a single SCSI bus, this field is set to 00. Otherwise, a value of 00 indicates a device attached to the internal SCSI bus of the card, and a value of 01 indicates a device attached to the external SCSI bus of the card.
- S,L** The **S,L** field identifies the SCSI ID and logical unit number (LUN) of the SCSI device. The S value indicates the SCSI ID and the L value indicates the LUN.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

Direct-bus-attached 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.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

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.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

Diskette drive location codes

Diskette drives are assigned location codes.

Diskette drives have location codes of either 00-00-0D-01 or 00-00-0D-02, indicating that they are attached to the standard I/O planar diskette ports 0 or 1.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

Dials/LPFKeys location codes

For a Dials/LPFKeys device attached to a graphics input adapter, the location code format is *AA-BB-CC*.

The individual fields are interpreted as follows:

- AA** A value of 00 for the **AA** field indicates the controlling adapter card is located in the CPU drawer or system unit, depending on the type of system.
- BB** The **BB** field identifies the I/O bus and slot containing the card. The first digit identifies the I/O bus. It is 0 for the standard I/O bus and 1 for the optional I/O bus. The second digit is the slot on the indicated I/O bus that contains the card.
- CC** The **CC** field indicates the card connector where the device is attached. The value is either 01 or 02, depending on whether the attached device is port 1 or port 2 on the card.

Note: Serially-attached Dials/LPFKeys devices do not indicate location codes. This is because these devices are considered to be attached to a tty device. The tty device is specified by the user during Dials/LPFKeys definition.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

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.

For more information about location codes, see *Understanding Location Codes* at <http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphau/loccodes.htm>.

Chapter 21. Setting up iSCSI

Setting up iSCSI involves configuring the adapter and adding or updating targets.

Configuring the iSCSI adapter in AIX

iSCSI adapter configuration is a very simple and straightforward task.

1. Enter **smit iscsi** at the AIX command prompt. The iSCSI screen displays.
2. Select **iSCSI Adapter** from the iSCSI screen. The iSCSI Adapter screen displays.
3. Select **Change / Show Characteristics of an iSCSI Adapter** from the iSCSI Adapter screen. The Change / Show Characteristics of an iSCSI Adapter screen displays.
4. Select the iSCSI adapter that you want to configure from the list. A configuration screen displays, similar to the following example.

```
[Entry Fields]
iSCSI Adapter          ics0
Description            iSCSI Adapter
Status                 Available
Location               10-60
Max. number of commands to queue to adapter  [200]      +          +#
Maximum Transfer Size  [0x100000] +
Discovery Filename     [/etc/iscsi/targetshw]
Discovery Policy       [file]
Automatic Discovery Secrets Filename  [/etc/iscsi/autosecret]
Adapter IP Address     [10.1.4.187]
Adapter Subnet Mask    [255.255.255.0]
Adapter Gateway Address [10.1.4.1]
Apply change to DATABASE only  no      +
```

Note: If you have a question about the purpose of a particular field, place the cursor in the field and press **F1** for help.

To use flat file discovery, type `file` in the **Discovery Policy** field. To use ODM discovery, type `odm` in the **Discovery Policy** field. For DHCP-discovered iSCSI targets, type `slp` in the **Discovery Policy** field.

Updating the flat file of an iSCSI target

The flat file is the static configuration file used to configure iSCSI targets. Its default filename is **/etc/iscsi/targetshw**.

You must explicitly specify all relevant iSCSI target discovery properties in the flat file. See targets File for more information.

Adding a statically-discovered iSCSI target into ODM

When auto discovery is not used, the iSCSI adapter obtains the iSCSI target descriptions from either a flat file or ODM.

You can use AIX commands or SMIT to manipulate iSCSI target information in ODM. The **chiscsi**, **lsiscsi**, **mkiscsi**, and **rmiscsi** commands change, display, add, and remove iSCSI target information from ODM.

To add one statically-discovered iSCSI target into ODM using SMIT, do the following:

1. Enter **smit iscsi** at the AIX command prompt. The iSCSI screen displays.
2. Select **iSCSI Target Device Parameters in ODM** from the iSCSI screen. The iSCSI Target Device Parameters in ODM screen displays.
3. Select **Add an iSCSI Target Device in ODM** from the iSCSI screen. The Add an iSCSI Target Device in ODM screen displays.
4. Select **Add a Statically Discovered iSCSI Target Device in ODM** from the iSCSI screen. The Add a Statically Discovered iSCSI Target Device in ODM screen displays.
5. Select the iSCSI adapter that you want to configure from the list. The Add a Statically Discovered iSCSI Target Device screen for the iSCSI adapter that you selected displays.
6. Enter the appropriate information in the fields. Below is an example.

```
[Entry Fields]
iSCSI Adapter                ics0
iSCSI Target Name            [iqn.mds9216.emc.sym]    +
iSCSI Group                  static    +
IP Address of iSCSI Target   [10.1.4.25]    +
Port Number of iSCSI Target  [3260]    +
Password                     [my password]    +
```

Note: If you have a question about the purpose of a particular field, place the cursor in the field and press **F1** for help.

Adding statically-discovered iSCSI targets from a flat file into ODM

You can use SMIT to import a flat file's information into ODM.

1. Enter **smit iscsi** at the AIX command prompt. The iSCSI screen displays.
2. Select **iSCSI Target Device Parameters in ODM** from the iSCSI screen. The iSCSI Target Device Parameters in ODM screen displays.
3. Select **Add an iSCSI Target Device in ODM** from the iSCSI screen. The Add an iSCSI Target Device in ODM screen displays.
4. Select **Add iSCSI Target Device Data in ODM from a File** from the iSCSI screen. The Add iSCSI Target Device Data in ODM from a File screen displays.
5. Select the iSCSI adapter that you want to configure from the list. The Add iSCSI Target Device Data in ODM from a File screen for the iSCSI adapter that you selected displays.
6. Enter the appropriate information in the fields. Below is an example.

```
[Entry Fields]
iSCSI Protocol Device        iscsi3
iSCSI Group                  [static]    +
Filename of iSCSI Targets    [/etc/iscsi/targetshw]    +
```

Note: If you have a question about the purpose of a particular field, place the cursor in the field and press **F1** for help.

Chapter 22. PCI hot plug management

You can insert a new PCI hot plug adapter into an available PCI slot while the operating system is running.

This can be another adapter of the same type that is currently installed or of a different type of PCI adapter. New resources are made available to the operating system and applications without having to restart the operating system. Some reasons for adding a hot plug adapter might include:

- Adding additional function or capacity to your existing hardware and firmware.
- Migrating PCI adapters from a system that no longer requires the function provided by those adapters.
- Installing a new system for which adapter cards become available after the initial configuration of optional hardware subsystems, including PCI adapters, and the installation and start of the operating system.

Note: If you add an adapter using a PCI hot plug replace or add operation, or using Dynamic Logical Partitioning, it and its child devices may not be available for specification as a boot device using the **bootlist** command. You may have to restart the machine to make all potential boot devices known to the operating system. An adapter already listed in the boot list, which is then replaced by the exact type of adapter, is still a valid boot device.

You can also remove a defective or failing PCI hot plug adapter or exchange it with another of the same type without shutting down or powering off the system. When you exchange the adapter, the existing device driver supports the adapter because it is of the same type. Device configuration and configuration information about devices below the adapter are retained for the replacement device. Some reasons for replacing an adapter might include:

- Temporarily replacing the card to aid in determining a problem or to isolate a failing FRU.
- Replacing a flawed, failing, or intermittently failing adapter with a functional card.
- Replacing a failing redundant adapter in an HACMP™ or multipath I/O configuration.

When you remove a hot plug adapter, the resources provided by the adapter become unavailable to the operating system and applications. Some reasons for removing an adapter might include:

- Removing existing I/O subsystems.
- Removing an adapter that is no longer required or is failing and a replacement card is not available.
- Migrating an adapter to another system when the function is no longer required on the system from which it is being removed.

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.

The system administrator must perform the following tasks before and after removing an adapter:

- Terminate and restore applications, daemons, or processes using the device.
- Unmount and remount file systems.
- 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.

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. For more information about the **rmdev** command, see **rmdev**.

In some cases, the you can 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: Although PCI hot plug management provides the capability of adding, removing, and replacing PCI adapters without powering off the system or restarting the operating system, not all devices in hot plug slots can be managed in this fashion. For example, the hard disk that makes up the rootvg volume group or the I/O controller to which it is attached cannot be removed or replaced without powering off the system because it is necessary for running the operating system. If the rootvg volume group is mirrored, you can use the **chpv** command to take the disks offline. If the rootvg volume group resides on one or more disks that are Multi-Path I/O (MPIO) enabled and connected to multiple I/O controllers, one of these I/O controllers can be removed (or replaced) without rebooting the system. In this situation, all paths from the I/O controller to be removed (or replaced) should be unconfigured using the **rmdev -R** command on the adapter. This will unconfigure the paths and the adapter. You can then proceed with Hot Plug Management. Before you attempt to remove or insert PCI hot plug adapters, refer to the *PCI Adapter Placement Reference*, (shipped with system units that support hot plug), to determine whether your adapter can be hot-swapped. Refer to your system unit documentation for instructions for installing or removing adapters.

For instructions on how to manage a PCI hot plug adapter, see the following:

Displaying PCI hot-plug slot information

Before you add, remove, or replace a hot-plug adapter, you can display information about the PCI hot-plug slots in a machine.

You can display the following information:

- A list of all the PCI hot-plug slots in the machine
- Whether a slot is available or empty
- Slots that are currently in use
- The characteristics of a specific slot such as slot name, description, connector type, and the attached device name

You can complete these tasks with Web-based System Manager. You can also use SMIT or system commands. To perform these tasks, you must log in as root user.

For additional information, see Chapter 22, “PCI hot plug management,” on page 297.

SMIT fastpath procedure

1. Type `smit devdrpci` at the system prompt, then press Enter.
2. Use the SMIT dialogs to complete the task.

To obtain additional information for completing the task, you can select the F1 Help key in the SMIT dialogs.

Commands procedure

You can use the following commands to display information about hot-plug slots and connected devices:

- The **Isslot** command displays a list of all the PCI hot-plug slots and their characteristics. For information about using this command, see **Isslot** in the *AIX 5L Version 5.3 Commands Reference, Volume 3*.
- The **Isdev** command displays the current state of all the devices installed in your system. For information about using this command, see **Isdev** in the *AIX 5L Version 5.3 Commands Reference, Volume 3*.

Unconfiguring PCI communications adapters

The following is an overview of the process for unconfiguring PCI communications adapters. This includes Ethernet, Token-ring, FDDI, and ATM adapters.

For procedural information, see “Communications adapter removal” on page 309.

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. For information about the **netstat** command, see **netstat**.

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.

To unconfigure the children of PCI bus pci1 and all other devices under them while retaining their device definitions in the Customized Devices object class, type:

```
rmdev -p pci1
```

The system displays a message similar to the following:

```
rmt0 Defined
hdisk1 Defined
scsil Defined
ent0 Defined
```

Removing or replacing a PCI hot plug adapter

You can remove or replace a PCI hot plug adapter from the system unit without shutting down the operating system or turning off the system power. Removing an adapter makes the resources provided by that adapter unavailable to the operating system and applications.

Before you can remove an adapter, you must unconfigure it. See “Communications adapter removal” on page 309, “Unconfiguring storage adapters” on page 316, or “Unconfiguring async adapters” on page 317 for instructions for unconfiguring adapters.

The following are the procedures for removing a PCI hot plug adapter. You can complete these tasks with Web-based System Manager. You can also use SMIT or system commands. To perform these tasks, you must log in as root user.

Replacing an adapter with another adapter of the same type retains the replaced adapter's configuration information and compares the information to the card that replaces it. The existing device driver of the replaced adapter must be able to support the replacement adapter.

For additional information, see Chapter 22, "PCI hot plug management," on page 297.

SMIT fastpath procedure

1. Type `smit devdrpci` at the system prompt, then press Enter.
2. Use the SMIT dialogs to complete the task.

To obtain additional information for completing the task, you can select the F1 Help key in the SMIT dialogs.

Commands procedure

You can use the following commands to display information about hot plug slots and connected devices and to remove a PCI hot plug adapter:

- The **lsslot** command displays a list of all the PCI hot plug slots and their characteristics. For information about using this command, see **lsslot** in the *AIX 5L Version 5.3 Commands Reference, Volume 3*.
- The **lsdev** command displays the current state of all the devices installed in your system. For information about using this command, see **lsdev** in the *AIX 5L Version 5.3 Commands Reference, Volume 3*.
- The **drslot** command prepares a hot plug slot for removal of a hot plug adapter. For information about using this command, see **drslot** in the *AIX 5L Version 5.3 Commands Reference, Volume 2*.

For information about the physical handling of a PCI hot plug adapter, refer to your system unit documentation.

Adding a PCI hot plug adapter

You can add a PCI hot plug adapter into an available slot in the system unit and make new resources available to the operating system and applications without having to reboot the operating system. The adapter can be another adapter type that is currently installed or it can be a different adapter type.

The following are the procedures for adding a new PCI hot plug adapter.

Attention: Before you attempt to add 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.

Adding a new PCI hot plug adapter involves the following tasks:

- Finding and identifying an available slot in the machine
- Preparing the slot for configuring the adapter
- Installing the device driver, if necessary
- Configuring the new adapter

You can complete these tasks with Web-based System Manager. You can also use SMIT or system commands. To perform these tasks, you must log in as root user.

For additional information, see Chapter 22, "PCI hot plug management," on page 297.

Note: When you add a hot plug adapter to the system, that adapter and its child devices might not be available for specification as a boot device using the **bootlist** command. You might be required to reboot your system to make all potential boot devices known to the operating system.

SMIT fastpath procedure

1. Type `smit devdrpci` at the system prompt, then press Enter.
2. Use the SMIT dialogs to complete the task.

To obtain additional information for completing the task, you can select the F1 Help key in the SMIT dialogs.

Commands procedure

You can use the following commands to display information about PCI hot plug slots and connected devices and to add a PCI hot plug adapter:

- The **lsslot** command displays a list of all the hot plug slots and their characteristics. For information about using this command, see **lsslot** in the *AIX 5L Version 5.3 Commands Reference, Volume 3*.
- The **lsdev** command displays the current state of all the devices installed in your system. For information about using this command, see **lsdev** in the *AIX 5L Version 5.3 Commands Reference, Volume 3*.
- The **drslot** command prepares a hot plug slot for adding or removing a hot plug adapter. For information about using this command, see **drslot** in the *AIX 5L Version 5.3 Commands Reference, Volume 2*.

For information about installing or removing adapters, refer to your system unit documentation.

Chapter 23. Multiple Path I/O

With Multiple Path I/O (MPIO), a device can be uniquely detected through one or more physical connections, or *paths*.

A path-control module (PCM) provides the path management functions.

An MPIO-capable device driver can control more than one type of target device. A PCM can support one or more specific devices. Therefore, one device driver can be interfaced to multiple PCMs that control the I/O across the paths to each of the target devices.

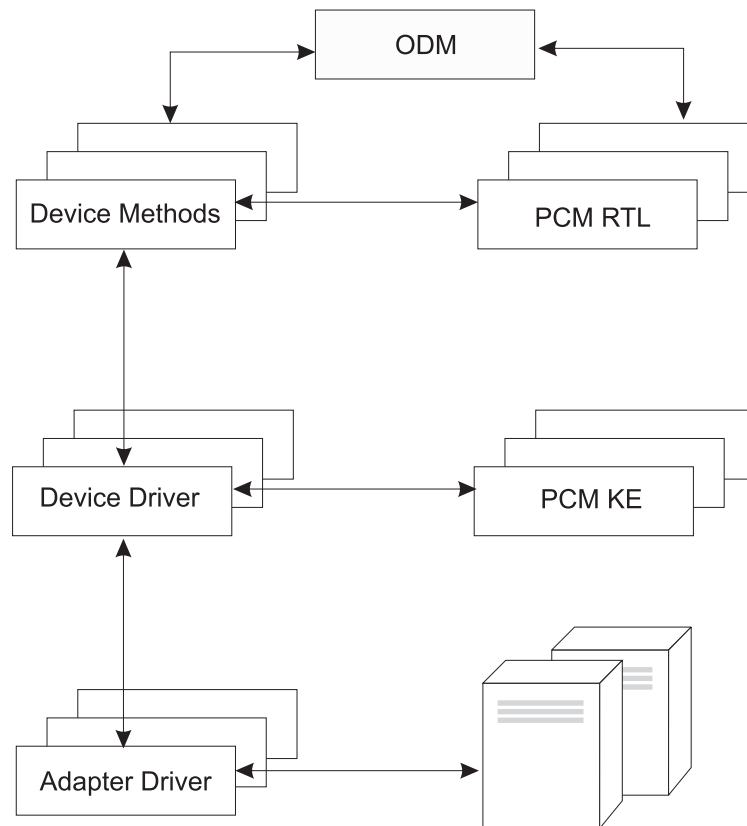


Figure 12. MPIO Component Interaction. This illustration shows the interaction between the different components that make up the MPIO solution. In this figure, the MPIO device driver controls multiple types of target devices, each requiring a different PCM. (KE=Kernel Extension, RTL=Run-time Loadable).

Before a device can take advantage of MPIO, the device's driver, methods, and predefined attributes in the Object Data Manager (ODM) must be modified to support detection, configuration, and management of multiple paths. The parallel SCSI and Fibre Channel disk device drivers and their device methods support MPIO disk devices. Beginning with AIX 5L Version 5.2 with the 5200-04 Recommended Maintenance package, iSCSI disk devices are supported as MPIO devices. Beginning with AIX 5.3, the Fibre Channel tape device driver and its device methods have been modified to support MPIO tape devices. Also, the predefined attributes for some devices in the ODM have been modified for MPIO.

The AIX PCM consists of: the PCM RTL configuration module and the PCM KE kernel extension. The PCM RTL is a run-time loadable module that enables the device methods to detect additional PCM KE device-specific or path ODM attributes that the PCM KE requires. The PCM RTL is loaded by a device method. One or more routines within the PCM RTL are then accessed to perform specific operations that initialize or modify PM KE variables.

The PCM KE supplies path-control management capabilities to any device driver that supports the MPIO interface. The PCM KE depends on device configuration to detect paths and communicate that information to the device driver. Each MPIO-capable device driver adds the paths to a device from its immediate parent or parents. The maintenance and scheduling of I/O across different paths is provided by the PCM KE and is not apparent to the MPIO-capable device driver.

The PCM KE can provide more than one routing algorithm, which can be selected by the user. The PCM KE also helps collect information that can be used to determine and select the best path for any I/O request. The PCM KE can select the best path based on a variety of criteria, including load balancing, connection speed, connection failure, and so on.

The AIX PCM has a health-check capability that can be used to do the following:

- Check the paths and determine which paths are currently usable for sending I/O
- Enable a path that was previously marked failed because of a temporary path fault (for example, when a cable to a device was removed and then reconnected)
- Check currently unused paths that would be used if a failover occurred (for example, when the algorithm attribute value is `failover`, the health check can test the alternate paths)

Not all disk devices and tape devices can be detected and configured using the AIX default PCMs. The AIX default PCMs consist of two path control modules, one to manage disk devices and another to manage tape devices. If your device is not detected, check with the device vendor to determine if a PCM is available for your device.

MPIO-capable device management

The Multiple Path I/O (MPIO) feature can be used to define alternate paths to a device for failover purposes.

Failover is a path-management algorithm that improves the reliability and availability of a device because the system automatically detects when one I/O path fails and re-routes I/O through an alternate path. Beginning with AIX 5.2, all SCSI SCSD disk drives are automatically configured as MPIO devices and a select number of Fibre Channel disk drives can be configured as MPIO Other disk. Other devices can be supported, providing the device driver is compatible with the MPIO implementation in AIX. For more information about MPIO concepts, see Chapter 23, “Multiple Path I/O,” on page 303.

MPIO is installed and configured as part of BOS installation. No further configuration is required, but you can add, remove, reconfigure, enable, and disable devices (or device paths) using SMIT, Web-based System Manager, or the command-line interface. The following commands help manage MPIO paths:

mkpath

Adds a path to a target device.

rmpath

Removes a path to a target device.

chpath

Changes an attribute or the operational status of a path to a target device.

lspath Displays information about paths to a target device.

Cabling a SCSI device as an MPIO device

A SCSI device can be supported by a maximum of two adapters when configured as a MPIO-capable device.

To cable a parallel SCSI device as an MPIO device, use the following simple configuration as an example. The following is the minimum configuration that must be done; your device might require additional configuration.

1. With the power off, install two SCSI adapters.
2. Cable the device to both SCSI adapters.
3. Power on the system.
4. Change the settings on one of the adapters to a unique SCSI ID. By default, SCSI adapters have a SCSI ID of 7. Because each ID must be unique, change one adapter to another number, for example, 6.
5. Run the **cfgmgr** command.
6. To verify the configuration, type the following on the command line:

```
lspath -l hdiskX
```

where *X* is the logical number of the newly configured device. The command output should display two paths and their status.

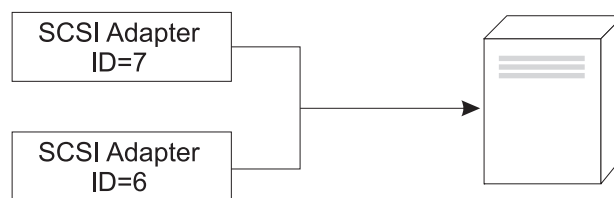


Figure 13. Cable Configuration for MPIO SCSI Device

This illustration shows cabling two SCSI adapters to the same device.

Cabling a Fibre Channel device as an MPIO device

A Fibre Channel device can be cabled to multiple adapters. There is no limit within the software.

To cable a Fibre Channel device as an MPIO device, use the following simple configuration as an example. The following is the minimum configuration that must be done; your device might require additional configuration.

1. With the power off, install two Fibre Channel adapters.
2. Cable the adapters to a switch or hub.
3. Cable the device to the switch or hub.
4. Power on the system.
5. To verify the configuration, type the following on the command line:

```
lspath -l hdiskX
```

where *X* is the logical number of the newly configured device. The command output should display one path for each adapter you installed and the status of each.

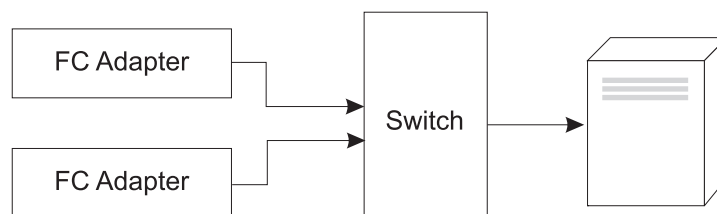


Figure 14. Cable Configuration for MPIO Fibre Channel Device. This illustration shows a simple configuration of two Fibre Channel adapters to a switch, which is cabled to a device.

MPIO device configuration

Configuring an MPIO-capable device uses the same commands as a non-MPIO device.

Beginning with AIX 5.2 with 5200-01, the **cfgmgr**, **mkdev**, **chdev**, **rmdev** and **lsdev** commands support managing MPIO device instances and display their attributes. An MPIO device instance also has path instances associated with the device instance. The **mkpath**, **chpath**, **rmpath**, and **lspath** commands manage path instances and display their attributes.

A path instance can be added or removed from an MPIO device without unconfiguring the device. For more information about these commands, see “MPIO-capable device management” on page 304.

An MPIO device can have additional attributes, but the required attributes that all MPIO devices must support are **reserve_policy** and **algorithm**. The **reserve_policy** attribute determines the type of reserve methodology the device driver will implement when the device is opened, and it can be used to limit device access from other adapters, whether on the same system or another system. The **algorithm** attribute defines the methodology that the PCM uses to manage I/O across the paths configured for a device. For more information about the **reserve_policy** attribute, see “MPIO device attributes” on page 307. For more information about the **algorithm** attribute, see “Path control module attributes” on page 308.

Supported multi-path devices

The AIX default PCMs support a set of disk devices and tape devices defined in the **devices.common.IBM.mpio.rte** fileset.

Devices not supported by the AIX disk PCMs or tape PCMs require the device vendor to provide attributes predefined in the ODM, a PCM, and any other supporting code necessary to recognize the device as MPIO-capable.

To determine which disk devices are supported by the AIX disk PCM, run the following script :

```
odmget -qDvDr=aixdiskpcmke PdDv | grep uniquetype | while read line
do
    utype=`echo $line | cut -d'"' -f2`
    dvc=`odmget -q"uniquetype=$utype AND attribute=dvc_support" PdAt`
    echo $dvc | grep values | cut -d'"' -f2
done
```

To determine which tape devices are supported by the AIX disk PCM, run the following script :

```
odmget -qDvDr=aixtapepcmke PdDv | grep uniquetype | while read line
do
    utype=`echo $line | cut -d'"' -f2`
    dvc=`odmget -q"uniquetype=$utype AND attribute=dvc_support" PdAt`
    echo $dvc | grep values | cut -d'"' -f2
done
```

The script output displays a list of unique device types supported by the AIX default PCMs. The three device types supported by the AIX Disk PCM are *self-configuring parallel SCSI disk* (**disk/scsi/scsd**) and *MPIO other FC disk* (**disk/fcp/mpioosdisk**), and *MPIO other iSCSI* (**disk/iscsi/mpioosdisk**). The device type supported by the AIX tape PCM is *MPIO other FC tape* (**tape/fcp/mpioost**).

MPIO other FC disk and *MPIO other FC tape* devices are a subset of other Fibre Channel disks and other Fibre Channel tapes, respectively. A device is supported as an *MPIO other FC* device only if there are no vendor-provided ODM predefined attributes and the device has been certified to operate with one of the AIX default PCMs. Certification does not guarantee that all device capabilities are supported when the device is configured as an *MPIO other FC* device.

An *MPIO other iSCSI* disk is a subset of other iSCSI disks. A device is supported as an *MPIO other iSCSI* disk only if there is no vendor-provided ODM predefined attributes and the device has been certified to operate with the AIX PCM. Certification does not guarantee that all device capabilities are supported when the device is configured as an *MPIO other iSCSI* disk.

To determine which devices are supported as *MPIO other FC* disk, run the following script:

```
odmget -q uniqueness=disk/fcp/mpioosdisk PdAt | grep deflt | cut -d'"' -f2
```

To determine which devices are supported as *MPIO other FC* tape, run the following script:

```
odmget -q "uniquetype=tape/fcp/mpioosdisk AND attribute=mpio_model_map PdAt | grep deflt | cut -d'"' -f2
```

To determine which devices are supported as *MPIO other iSCSI* disks, run the following script:

```
odmget -q uniqueness=disk/iscsi/mpioosdisk PdAt | grep deflt | cut -d'"' -f2
```

The script output displays a list of inquiry data that contains the device vendor and model.

To display all MPIO-capable devices that are installed on the system, run the following script:

```
odmget -q attribute=unique_id PdAt | grep uniqueness | cut -d'"' -f2
```

The script output displays a list of unique MPIO-capable device types, supported by the AIX default PCMs and vendor provided PCMs.

MPIO device attributes

The following attributes are supported only by multi-path devices. The attributes can be displayed or changed using the Web-based System Manager, SMIT, or commands (in particular, the **lsattr** and the **chdev** commands).

For more information about these commands, see **lsattr** and **chdev**.

The required device attributes that all MPIO devices must support is **reserve_policy**. Typically, a multi-path device also has the **PR_key** device attribute. A multi-path device can have additional device-specific attributes. Other device-specific attributes are as follows:

reserve_policy

Defines whether a reservation methodology is employed when the device is opened. The values are as follows:

no_reserve

Does not apply a reservation methodology for the device. The device might be accessed by other initiators, and these initiators might be on other host systems.

single_path_reserve

Applies a SCSI2 reserve methodology for the device, which means the device can be accessed only by the initiator that issued the reserve. This policy prevents other initiators in the same host or on other hosts from accessing the device. This policy uses the SCSI2 reserve to lock the device to a single initiator (path), and commands routed through any other path result in a reservation conflict.

Path selection algorithms that alternate commands among multiple paths can result in thrashing when the **single_path_reserve** value is selected. As an example, assume a device-specific PCM has a required attribute that is set to a value that distributes I/O across multiple paths. When **single_path_reserve** is in effect, the disk driver must issue a bus device reset (BDR) and then issue a reserve using a new path for sending the next command to break the previous reservation. Each time a different path is selected,

thrashing occurs and performance degrades because of the overhead of sending a BDR and issuing a reserve to the target device. (The AIX PCM does not allow you to select an algorithm that could cause thrashing.)

PR_exclusive

Applies a SCSI3 persistent-reserve, exclusive-host methodology when the device is opened. The **PR_key** attribute value must be unique for each host system. The **PR_key** attribute is used to prevent access to the device by initiators from other host systems.

PR_shared

Applies a SCSI3 persistent-reserve, shared-host methodology when the device is opened. The **PR_key** value must be a unique value for each host system. Initiators from other host systems are required to register before they can access the device.

PR_key

Required only if the device supports any of the persistent reserve policies (**PR_exclusive** or **PR_shared**).

Path control module attributes

In addition to the default AIX default path control modules (PCMs), a device-specific PCM might be supplied by a device vendor. The set of user changeable attributes is defined by the device vendor. A device-specific PCM can have device and path attributes.

The following are the device attributes for the AIX default PCMs:

algorithm

Determines the methodology by which the I/O is distributed across the paths for a device. The algorithm attribute has the following values:

failover

Sends all I/O down a single path. If the path is determined to be faulty, an alternate path is selected for sending all I/O. This algorithm keeps track of all the enabled paths in an ordered list. If the path being used to send I/O becomes marked failed or disabled, the next enabled path in the list is selected. The sequence within the list is determined by the path priority **path priority** attribute. This algorithm is available in both AIX disk PCM and AIX tape PCM.

round_robin

Distributes the I/O across all enabled paths. The path priority is determined by the **path priority** path priority attribute value. If a path becomes marked failed or disabled, it is no longer used for sending I/O. The priority of the remaining paths is then recalculated to determine the percentage of I/O that should be sent down each path. If all paths have the same value, then the I/O is then equally distributed across all enabled paths. This algorithm is available only in the AIX disk PCM; AIX tape PCM does not support **round_robin**.

hcheck_mode

Determines which paths should be checked when the health check capability is used. The attribute supports the following modes:

enabled

Sends the **healthcheck** command down paths with a state of enabled.

failed Sends the **healthcheck** command down paths with a state of failed.

nonactive

(Default) Sends the **healthcheck** command down paths that have no active I/O, including paths with a state of failed. If the algorithm selected is failover, then the **healthcheck** command is also sent on each of the paths that have a state of enabled but have no

active I/O. If the algorithm selected is `round_robin`, then the **healthcheck** command is only sent on paths with a state of failed, because the `round_robin` algorithm keeps all enabled paths active with I/O.

hcheck_interval

Defines how often the health check is performed on the paths for a device. The attribute supports a range from 0 to 3600 seconds. When a value of 0 is selected, health checking is disabled.

dist_tw_width

Defines the duration of a "time window". This is the time frame during which the distributed error detection algorithm will cumulate I/Os returning with an error. The **dist_tw_width** attribute unit of measure is milliseconds. Lowering this attribute's value decreases the time duration of each sample taken and decreases the algorithm's sensitivity to small bursts of I/O errors. Increasing this attribute's value will increase the algorithm's sensitivity to small bursts of errors and the probability of failing a path.

dist_err_percent

Defines the percentage of "time windows" having an error allowed on a path before the path is failed due to poor performance. The **dist_err_percent** has a range from 0-100. The distributed error detection algorithm will be disabled when the attribute is set to zero (0). The default setting is zero. The distributed error detection algorithm samples the fabric connecting the device to the adapter for errors. The algorithm calculates a percentage of samples with errors and will fail a path if the calculated value is larger than the **dist_err_percent** attribute value.

The following is the path attribute for the AIX PCM:

path priority

Modifies the behavior of the algorithm methodology on the list of paths.

When the algorithm attribute value is `failover`, the paths are kept in a list. The sequence in this list determines which path is selected first and, if a path fails, which path is selected next. The sequence is determined by the value of the path priority attribute. A priority of 1 is the highest priority. Multiple paths can have the same priority value, but if all paths have the same value, selection is based on when each path was configured.

When the algorithm attribute value is **round_robin**, the sequence is determined by percent of I/O. The path priority value determines the percentage of the I/O that should be processed down each path. I/O is distributed across the enabled paths. A path is selected until it meets its required percentage. The algorithm then marks that path failed or disabled to keep the distribution of I/O requests based on the path priority value.

Communications adapter removal

Before you can remove or replace a hot-plug adapter, you must unconfigure that adapter.

To unconfigure communications adapters, use the following procedures:

- "Unconfiguring Ethernet, Token-ring, FDDI, and ATM adapters" on page 310
- "Unconfiguring WAN adapters" on page 311
- "Unconfiguring IBM 4-Port 10/100 Base-TX Ethernet PCI adapters" on page 312
- "Unconfiguring ATM adapters" on page 313

Unconfiguring a communications adapter involves the following tasks:

- Closing all applications that are using the adapter you are removing or replacing
- Ensuring that all devices connected to the adapter are identified and stopped
- Listing all slots that are currently in use or a slot that is occupied by a specific adapter
- Identifying the adapter's slot location
- Displaying and removing interface information from the network interface list

- Making the adapter unavailable

To perform these tasks, you must log in as **root**.

For additional information about unconfiguring communications adapters, see Chapter 22, “PCI hot plug management,” on page 297.

Unconfiguring Ethernet, Token-ring, FDDI, and ATM adapters

To unconfigure an Ethernet, Token-ring, FDDI, or ATM Adapter, perform the following steps:

1. Type `lsslot -c pci` to list all the hot-plug slots in the system unit and display their characteristics.
2. Type the appropriate SMIT command, shown in the following examples, to list installed adapters and show the current state of all the devices in the system unit:

smit lsdenet	To list Ethernet adapters
smit lsdtok	To list token-ring adapters
smit ls_atm	To list ATM adapters

The following naming convention is used for the different type of adapters:

Name	Adapter Type
atm0, atm1, ...	ATM adapter
ent0, ent1, ...	Ethernet adapter
tok0, tok1, ...	Token Ring adapter

3. Close all applications that are using the adapter you are unconfiguring. To continue with this procedure, network dump locations must be disabled on the system. To look for and disable network dump locations, do the following:
 - a. Type the following from a command line:

```
smit dump
```
 - b. Select **Show Current Dump Devices**.
 - c. Check whether any configured dump device shows a network location. If not, exit SMIT and you are ready for step 4. To change a dump device to a local location, select **Cancel** or press F3 and continue with the following step.
 - d. If the primary dump device shows a network location, change to a local location by selecting **Change the Primary Dump Device** and then enter the local location in the **Primary dump device** field.
 - e. If the secondary dump device shows a network location, change to a local location by selecting **Change the Secondary Dump Device** and then enter the local location in the **Secondary dump device** field.
 - f. When finished, click **OK** or press Enter.
4. Type `netstat -i` to display a list of all configured interfaces and determine whether your adapter is configured for TCP/IP. Output similar to the following displays:

```
Name Mtu Network Address Ipkts Ierrs Opkts Oerrs Coll
lo0 16896 link#1 076 0 118 0 0
lo0 16896 127 127.0.0.1 076 0 118 0 0
lo0 16896 ::1 076 0 118 0 0
tr0 1492 link#2 8.0.5a.b8.b.ec 151 0 405 11 0
tr0 1492 19.13.97 19.13.97.106 151 0 405 11 0
at0 9180 link#3 0.4.ac.ad.e0.ad 0 0 0 0 0
at0 9180 6.6.6 6.6.6.5 0 0 0 0 0
en0 1500 link#5 0.11.0.66.11.1 212 0 1 0 0
en0 1500 8.8.8 8.8.8.106 212 0 1 0 0
```

Token-ring adapters can have only one interface. Ethernet adapters can have two interfaces. ATM adapters can have multiple interfaces. For additional information, see “Communications adapter removal” on page 309.

5. Type the appropriate **ifconfig** command, shown in the following examples, to remove the interface from the network interface list.

<code>ifconfig en0 detach</code>	To remove the standard Ethernet interface
<code>ifconfig et0 detach</code>	To remove the IEEE 802.3 Ethernet interface
<code>ifconfig tr0 detach</code>	To remove a token-ring interface
<code>ifconfig at0 detach</code>	To remove an ATM interface

For an explanation of the association between these adapters and their interfaces, see “Communications adapter removal” on page 309.

6. Type the appropriate **rmdev** command, shown in the following examples, to unconfigure the adapter and *keep* its device definition in the Customized Devices Object Class:

<code>rmdev -l ent0</code>	To unconfigure an Ethernet adapter
<code>rmdev -l tok1</code>	To unconfigure a token-ring adapter
<code>rmdev -l atm1</code>	To unconfigure an ATM adapter
<code>rmdev -p pci1</code>	To unconfigure the children of a PCI bus and all other devices under them while retaining their device definitions in the Customized Devices object class.

Note: To unconfigure the adapter and *remove* the device definition in the Customized Devices object class, you can use the **rmdev** command with the **-d** flag.

Attention: *Do not* use the **-d** flag with the **rmdev** command for a hot-plug operation unless your intent is to remove the adapter and not replace it.

Unconfiguring WAN adapters

You can unconfigure a WAN adapter.

To unconfigure a WAN Adapter:

1. Type `lsslot -c pci` to list all the hot-plug slots in the system unit and display their characteristics.
2. Type the appropriate SMIT command, shown in the following examples, to list installed adapters and show the current state of all the devices in the system unit:

<code>smit 331121b9_ls</code>	To list 2-Port Multiprotocol WAN adapters
<code>smit riciophx_ls</code>	To list ARTIC WAN adapters

The following naming convention is used for the different type of adapters:

Name	Adapter Type
dpmpa	2-Port Multiprotocol Adapter
riciop	ARTIC960 Adapter

3. Type `lsdev -C -c port` to list X.25 ports on your host. A message similar to the following displays:


```

sx25a0 Available 00-05-01-00 X.25 Port
x25s0 Available 00-05-01-00-00 V.3 X.25 Emulator
      
```
4. Close all applications that are using the adapter you are unconfiguring. To continue with this procedure, network dump locations must be disabled on the system. To look for and disable network dump locations, do the following:
 - a. Type the following from a command line:

smit dump

- b. Select **Show Current Dump Devices**.
 - c. Check whether any configured dump device shows a network location. If not, exit SMIT and you are ready for step 5 below. To change a dump device to a local location, select **Cancel** or press F3 and continue with the following step.
 - d. If the primary dump device shows a network location, change to a local location by selecting **Change the Primary Dump Device** and then enter the local location in the **Primary dump device** field.
 - e. If the secondary dump device shows a network location, change to a local location by selecting **Change the Secondary Dump Device** and then enter the local location in the **Secondary dump device** field.
 - f. When finished, click **OK** or press Enter.
5. Remove an X.25 driver and port, following the steps in Configuration Commands in *AIXlink/X.25 Version 2.1 for AIX: Guide and Reference*.
 6. Use the commands in the following table to unconfigure and remove the device drivers and emulator ports for these adapters:

2-Port Multiprotocol adapter	
smit rmhdlcdpmpdd	To unconfigure the device
smit rmsdlcscied	To unconfigure the SDLC COMIO emulator

ARTIC960Hx PCI adapter	
smit rmtsdd	To unconfigure the device driver
smit rmtsports	To remove an MPQP COMIO emulation port

Unconfiguring IBM 4-Port 10/100 Base-TX Ethernet PCI adapters

The 4-Port 10/100 Base-TX Ethernet PCI adapter has four ethernet ports and each port must be unconfigured before you can remove the adapter.

1. Type `lsslot -c pci` to list all the hot-plug slots in the system unit and display their characteristics.
2. Type `smit lsdenet` to list all the devices in the PCI subclass. A message similar to the following displays:

```
ent1 Available 1N-00 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 1)
ent2 Available 1N-08 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 2)
ent3 Available 1N-10 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 3)
ent4 Available 1N-18 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 4)
```
3. Close all applications that are using the adapter you are unconfiguring. To continue with this procedure, network dump locations must be disabled on the system. To look for and disable network dump locations, do the following:
 - a. Type the following from a command line:
smit dump
 - b. Select **Show Current Dump Devices**.
 - c. Check whether any configured dump device shows a network location. If not, exit SMIT and you are ready for step 4. To change a dump device to a local location, select **Cancel** or press F3 and continue with the following step.
 - d. If the primary dump device shows a network location, change to a local location by selecting **Change the Primary Dump Device** and then enter the local location in the **Primary dump device** field.

- e. If the secondary dump device shows a network location, change to a local location by selecting **Change the Secondary Dump Device** and then enter the local location in the **Secondary dump device** field.
 - f. When finished, click **OK** or press Enter.
4. Type `netstat -i` to display a list of all configured interfaces and determine whether your adapter is configured for TCP/IP. Output similar to the following displays:

```

Name Mtu Network Address Ipkts Ierrs Opkts Oerrs Coll
lo0 16896 link#1 076 0 118 0 0
lo0 16896 127 127.0.0.1 076 0 118 0 0
lo0 16896 ::1 076 0 118 0 0
tr0 1492 link#2 8.0.5a.b8.b.ec 151 0 405 11 0
tr0 1492 19.13.97 19.13.97.106 151 0 405 11 0
at0 9180 link#3 0.4.ac.ad.e0.ad 0 0 0 0 0
at0 9180 6.6.6 6.6.6.5 0 0 0 0 0
en0 1500 link#5 0.11.0.66.11.1 212 0 1 0 0
en0 1500 8.8.8 8.8.8.106 212 0 1 0 0

```

Ethernet adapters can have two interfaces, for example, `et0` and `en0`. For additional information, see “Communications adapter removal” on page 309.

5. Use the **ifconfig** command to remove each interface from the network interface list. For example, type `ifconfig en0 detach` to remove the standard Ethernet interface, and type `ifconfig et0` to remove the IEEE 802.3 interface. For an explanation of the association between these adapters and their interfaces, see “Communications adapter removal” on page 309.
6. Use the **rmdev** command to unconfigure the adapter and retain its device definition in the Customized Devices Object Class. For example, `rmdev -l ent0`.

Note: To unconfigure the adapter and *remove* the device definition in the Customized Devices object class, you can use the **rmdev** command with the **-d** flag.

Attention: *Do not* use the **-d** flag with the **rmdev** command for a hot-plug operation unless your intent is to remove the adapter and not replace it.

Unconfiguring ATM adapters

You must unconfigure each LAN-emulated device before you can remove the adapter.

Classic IP and LAN emulation protocols can run over ATM adapters. LAN emulation protocol enables the implementation of emulated LANs over an ATM network. Emulated LANs can be Ethernet/IEEE 802.3, Token-ring/IEEE 802.5, and MPOA (MultiProtocol Over ATM).

For instructions for removing a classical interface, see “Unconfiguring Ethernet, Token-ring, FDDI, and ATM adapters” on page 310. To remove a LAN interface, do the following:

1. Type `lsslot -c pci` to list all the hot-plug slots in the system unit and display their characteristics.
2. Type `smit ls_atm` to list all the ATM adapters. A message similar to the following displays:

```

:
:
atm0 Available 04-04 IBM PCI 155 Mbps ATM Adapter (14107c00)
atm1 Available 04-06 IBM PCI 155 Mbps ATM Adapter (14104e00)

```

3. Type `smit lista1_atmle` to list all the LAN-emulated clients on the adapters. A message similar to the following displays:

```

ent1 Available ATM LAN Emulation Client (Ethernet)
ent2 Available ATM LAN Emulation Client (Ethernet)
ent3 Available ATM LAN Emulation Client (Ethernet)
tok1 Available ATM LAN Emulation Client (Token Ring)
tok2 Available ATM LAN Emulation Client (Token Ring)

```

All ATM adapters can have multiple emulated clients running on them.

4. Type `smit listall_mpoa` to list all the LAN-emulated clients on the adapters. A message similar to the following displays:

```
mpc0 Available      ATM LAN Emulation MPOA Client
```

`atm0` and `atm1` are the physical ATM adapters. `mpc0` is an MPOA-emulated client. `ent1`, `ent2`, `ent3`, `tok1`, and `tok2` are LAN-emulated clients.

5. Type `entstat` to determine on which adapter the client is running. A message similar to the following displays:

```
-----  
ETHERNET STATISTICS (ent1) :  
Device Type: ATM LAN EmulationATM Hardware Address: 00:04:ac:ad:e0:ad  
.  
.  
.  
ATM LAN Emulation Specific Statistics:  
-----  
Emulated LAN Name: ETHelan3  
Local ATM Device Name: atm0  
Local LAN MAC Address:  
.  
.
```

6. Close all applications that are using the adapter you are unconfiguring. To continue with this procedure, network dump locations must be disabled on the system. To look for and disable network dump locations, do the following:

- a. Type the following from a command line:

```
smit dump
```

- b. Select **Show Current Dump Devices**.

- c. Check whether any configured dump device shows a network location. If not, exit SMIT and you are ready for step 7. To change a dump device to a local location, select **Cancel** or press F3 and continue with the following step.

- d. If the primary dump device shows a network location, change to a local location by selecting **Change the Primary Dump Device** and then enter the local location in the **Primary dump device** field.

- e. If the secondary dump device shows a network location, change to a local location by selecting **Change the Secondary Dump Device** and then enter the local location in the **Secondary dump device** field.

- f. When finished, click **OK** or press Enter.

7. Use the `rmdev -l device` command to unconfigure the interfaces in the following order:

- Emulated interface = `en1`, `et1`, `en2`, `et2`, `tr1`, `tr2` ...
- Emulated interface = `ent1`, `ent2`, `tok1`, `tok2` ...
- Multiprotocol Over ATM (MPOA) = `mpc0`
- ATM adapter = `atm0`

8. To unconfigure the SCSI adapter `scsi1` and all of its children while retaining their device definitions in the Customized Devices object class, type:

```
rmdev -R scsi1
```

The system displays a message similar to the following:

```
rmt0 Defined  
hdisk1 Defined  
scsi1 Defined
```

9. To unconfigure just the children of the SCSI adapter `scsi1`, but not the adapter itself, while retaining their device definitions in the Customized Devices object class, type:

```
rmdev -p scsi1
```


The system displays a message similar to the following:

```
rmt0 Defined
hdisk1 Defined
```

10. To unconfigure the children of PCI bus `pci1` and all other devices under them while retaining their device definitions in the Customized Devices object class, type:

```
rmdev -p pci1
```

The system displays a message similar to the following:

```
rmt0 Defined
hdisk1 Defined
scsi1 Defined
ent0 Defined
```

Adapter removal problem resolution

You may receive error messages if the device is open when you use the **rmdev** command to unconfigure an adapter.

If the following type of message displays when you use the **rmdev** command to unconfigure an adapter, this indicates that the device is open, possibly because applications are still trying to access the adapter you are trying to remove or replace.

```
#rmdev -l ent0
Method error (/usr/lib/methods/ucfgent):
0514-062
Cannot perform the requested function because the
specified device is busy.
```

To resolve the problem, you must identify any applications that are still using the adapter and close them. These applications can include the following:

- TCP/IP
- SNA
- OSI
- IPX/SPX
- Novell NetWare
- Streams
- The generic data link control (GDLC)
 - IEEE Ethernet DLC
 - Token-ring DLC
 - FDDI DLC

Systems Network Architecture applications

Some SNA applications that may be using your adapter include:

- DB2®
- TXSeries® (CICS® & Encina®)
- DirectTalk®
- MQSeries®
- HCON
- ADSM

Streams applications

Some of the streams-based applications that may be using your adapter include:

- IPX/SPX
- Novell NetWare V4 and Novell NetWare Services 4.1
- Connections and NetBios for this operating system

Applications running on WAN adapters

Applications that may be using your WAN adapter include:

- SDLC
- Bisync
- X.25
- ISDN
- QLLC for X.25

TCP/IP applications

All TCP/IP applications using the interface layer can be detached with the **ifconfig** command. This causes the applications using TCP/IP to time out and warn users that the interface is down. After you add or replace the adapter and run the **ifconfig** command to attach the interface, the applications resume.

Unconfiguring storage adapters

Before you can remove or replace a storage adapter, you must unconfigure that adapter.

To perform these tasks, you must log in as root user.

The following are the steps for unconfiguring SCSI, SSA, and Fibre Channel storage adapters.

Unconfiguring a storage adapter involves the following tasks:

- Closing all applications that are using the adapter you are removing, replacing, or moving
- Unmounting file systems
- Ensuring that all devices connected to the adapter are identified and stopped
- Listing all slots that are currently in use or a slot that is occupied by a specific adapter
- Identifying the adapter's slot location
- Making parent and child devices unavailable
- Making the adapter unavailable

For additional information, see Chapter 22, "PCI hot plug management," on page 297.

Unconfiguring SCSI, SSA, and Fibre Channel adapters

Storage adapters are generally parent devices to media devices, such as disk or tape drives. Removing the parent requires that all attached child devices either be removed or placed in the define state.

To unconfigure SCSI, SSA, and Fibre Channel adapters:

1. Close all applications that are using the adapter you are unconfiguring.
2. Type `lsslot-c pci` to list all the hot plug slots in the system unit and display their characteristics.
3. Type `lsdev -C` to list the current state of all the devices in the system unit.
4. Type `umount` to unmount previously mounted file systems, directories, or files using this adapter. For additional information, see Mount a JFS or JFS2.
5. Type `rmdev -l adapter -R` to make the adapter unavailable.

Attention: Do *not* use the **-d** flag with the **rmdev** command for hot plug operations because this will cause your configuration to be removed.

Unconfiguring async adapters

You can unconfigure an async adapter.

To perform these tasks, you must log in as root user.

The following are the steps for unconfiguring async adapters.

Before you can remove or replace an async adapter, you must unconfigure that adapter. Unconfiguring an async adapter involves the following tasks:

- Closing all applications that are using the adapter you are removing, replacing, or moving
- Ensuring that all devices connected to the adapter are identified and stopped
- Listing all slots that are currently in use or a slot that is occupied by a specific adapter
- Identifying the adapter's slot location
- Making parent and child devices unavailable
- Making the adapter unavailable

For additional information, see Chapter 22, "PCI hot plug management," on page 297.

Procedure

Before you can replace or remove an async adapter, you must unconfigure the adapter and all the devices controlled by that adapter. To unconfigure the devices, you must terminate all the processes using those devices. Use the following steps:

1. Close all applications that are using the adapter you are unconfiguring.
2. Type `lsslot-c pci` to list all the hot plug slots in the system unit and display their characteristics.
3. Type `lsdev -C -c tty` to list all available tty devices and the current state of all the devices in the system unit.
4. Type `lsdev -C -c printer` to list all printer and plotter devices connected to the adapter. For additional information, see Printing administration in the *Printers and printing*.
5. Use the **rmdev** command to make the adapter unavailable.

Attention: Do *not* use the **-d** flag with the **rmdev** command for hot plug operations because this will cause your configuration to be removed.

Determining the cause of device problems

You can determine the cause of device problems.

Device software check

Correct a device software problem by:

- Checking the Error Log
- Listing All Devices
- Checking the State of a Device
- Checking the Attributes of a Device
- Changing the Attributes of a Device
- Using a Device with Another Application
- Defining a New Device

Error log check

Check the error log to see whether any errors are recorded for either the device, its adapter, or the application using the device. Go to Error Logging Facility for information about performing this check. Return to this step after completing the procedures.

Did you correct the problem with the device?

If you were not able to correct the correct the problem using the previous method, go to the next step (**Device listing**) to list all of the devices.

Device listing

Use the **lsdev -C** command to list all defined or available devices. This command shows the characteristics of all the devices in your system.

If the device is in the list of devices, go to the next step (**Device state check**) to check the state of the device.

If the device is not in the list of devices, define a new device (see **New device definition**, below).

Device state check

Find the device in the list generated from the **lsdev -C** command. Check whether the device is in the Available state.

If the device is in the Available state, go to the next step (**Device attribute check**) to check the device attributes.

If the device is not in the Available state, define a new device (see **New device definition**, below).

Device attribute check

Use the **lsattr -E -l DeviceName** command to list the attributes of your device.

The **lsattr** command shows attribute characteristics and possible values of attributes for devices in the system. Refer to the documentation for the specific device for the correct settings.

If the device attributes are set correctly, see **Device use with another application**, below.

If the device attributes are not set correctly, go to the next step, **Device attribute change**.

Device attribute change

Use the **chdev -l Name -a Attribute=Value** command to change device attributes. Before you run this command, refer to *AIX 5L Version 5.3 Commands Reference, Volume 1*.

The **chdev** command changes the characteristics of the device you specify with the **-l Name** flag.

If changing the attributes did not correct the problem with the device, go to the next step, **Device use with another application**.

Device use with another application

Try using the device with another application. If the device works correctly with another application, there might be a problem with the first application.

If the device worked correctly with another application, you might have a problem with the first application. Report the problem to your software service representative.

If the device did not work correctly with another application, go to the next step, **New device definition**.

New device definition

Note: You must either have root user authority or be a member of the security group to use the **mkdev** command.

Use the **mkdev** command to add a device to the system.

The **mkdev** command can either define and make available a new device or make available a device that is already defined. You can uniquely identify the predefined device by using any combination of the **-c**, **-s**, and **-t** flags. Before you run this command, refer to the *AIX 5L Version 5.3 Commands Reference, Volume 3*.

If defining the device did not correct the problem, You can either stop and report the problem to your service representative or use a diagnostics program to test your device.

Checking the device connections

To check your device connections, perform the following steps:

1. Check that power is available at the electrical outlet.
2. Check that the device power cable is correctly attached to the device and to the electrical outlet.
3. Check that the device signal cable is attached correctly to the device and to the correct connection on the system unit.
4. For SCSI devices, check that the SCSI terminator is correctly attached and the SCSI address setting is correct.
5. For communications devices, check that the device is correctly attached to the communications line.
6. Check that the device is turned on.

Refer to the documentation for the specific device for cabling and configuring procedures and for further troubleshooting information.

If your check of the device connections have not corrected the problem. Go to the next step, "Checking the ready state of a device."

Checking the ready state of a device

You can check to see if a device is in a ready state.

To determine whether the device is in a ready state, do the following:

1. Check that the device's Ready indicator is on.
2. Check that removable media, such as tape, diskette, and optical devices, are inserted correctly.
3. Check the ribbon, the paper supply, and the toner supply for printers and plotters.
4. Check that the write medium is write-enabled if you are trying to write to the device.

Did your checks correct the problem with the device?

If your check of the device's ready state did not correct the problem, go to the next step, "Device diagnostics" on page 320.

Device diagnostics

To determine if a device is defective, run your hardware diagnostics.

If running hardware diagnostics fails to find a problem with your device, check the device software (see “Determining the cause of device problems” on page 317). If your device passes the diagnostic tests, you might have a problem with the way your device works with your system software. If it is possible that the preceding problem exists, report the problem to your software service organization.

Chapter 24. Tape drives

The system management functions described here relate to tape drives.

Many of these functions alter or get information from the device configuration database, which contains information about the devices on your system. The device configuration database consists of the predefined configuration database, which contains information about all possible types of devices supported on the system, and the customized configuration database, which contains information about the particular devices currently on the system. For the operating system to make use of a tape drive, or any other device, the device must be defined in the customized configuration database and must have a device type defined in the predefined configuration database.

Tape drive attributes

You can adjust these tape drive attributes to meet the needs of your system.

The attributes can be displayed or changed using the Web-based System Manager Devices application, SMIT, or commands (in particular, the **lsattr** and the **chdev** commands).

Each type of tape drive only uses a subset of all the attributes.

General information about each attribute

Block size

The block size attribute indicates the block size to use when reading or writing the tape. Data is written to tape in blocks of data, with inter-record gaps between blocks. Larger records are useful when writing to unformatted tape, because the number of inter-record gaps is reduced across the tape, allowing more data to be written. A value of **0** indicates variable length blocks. The allowable values and default values vary depending on the tape drive.

Device Buffers

Setting the Device Buffers attribute (using the **chdev** command) to `mode=yes` indicates an application is notified of write completion after the data has been transferred to the data buffer of the tape drive, but not necessarily after the data is actually written to the tape. If you specify the `mode=no`, an application is notified of write completion only after the data is actually written to the tape. Streaming mode cannot be maintained for reading or writing if this attribute is set to the `mode=no` value. The default value is `mode=yes`.

With the `mode=no` value, the tape drive is slower but has more complete data in the event of a power outage or system failure and allows better handling of end-of-media conditions.

Extended File Marks

Setting the Extended File Marks attribute (for **chdev** command, the **extfm** attribute) to the `no` value writes a regular file mark to tape whenever a file mark is written. Setting this attribute to the `yes` value writes an extended file mark. For tape drives, this attribute can be set on. The default value is `no`. For example, extended filemarks on 8 mm tape drives use 2.2 MB of tape and can take up to 8.5 seconds to write. Regular file marks use 184 KB and take approximately 1.5 seconds to write.

To reduce errors when you use an 8 mm tape in append mode, use extended file marks for better positioning after reverse operations at file marks.

Retension

Setting the Retension attribute (for the **chdev** command, the **ret** attribute) to `ret=yes` instructs the tape drive to re-tension a tape automatically whenever a tape is inserted or the drive is reset. *Retensioning* a tape means to wind to the end of the tape and then rewind to the beginning of the tape to even the tension throughout the tape. Retensioning the tape can reduce errors, but this

action can take several minutes. If you specify the `ret=no` value, the tape drive does not automatically retension the tape. The default value is `yes`.

Density Setting #1 and Density Setting #2

Density Setting #1 (for the `chdev` command, the `density_set_1` attribute) sets the density value that the tape drive writes when using special files `/dev/rmt*`, `/dev/rmt*.1`, `/dev/rmt*.2`, and `/dev/rmt*.3`. Density Setting #2 (for `chdev`, the `density_set_2` attribute) sets the density value that the tape drive writes when using special files `/dev/rmt*.4`, `/dev/rmt*.5`, `/dev/rmt*.6`, and `/dev/rmt*.7`. See “Special files for tape drives” on page 331 for more information.

The density settings are represented as decimal numbers in the range **0** to **255**. A zero (**0**) setting selects the default density for the tape drive, which is usually the high density setting of the drive. Specific permitted values and their meanings vary with different types of tape drives. These attributes do not affect the ability of the tape drive to read tapes written in all densities supported by the tape drive. It is customary to set Density Setting #1 to the highest density possible on the tape drive and Density Setting #2 to the second highest density possible on the tape drive.

Reserve support

For tape drives that use the Reserve attribute (for the `chdev` command, the `res_support` attribute), specifying the value `res_support=yes` causes the tape drive to be reserved on the SCSI bus while it is open. If more than one SCSI adapter shares the tape device, this ensures access by a single adapter while the device is open. Some SCSI tape drives do not support the reserve or release commands. Some SCSI tape drives have a predefined value for this attribute so that reserve or release commands are always supported.

Variable Length Block Size

The Variable Length Block Size attribute (for the `chdev` command, the `var_block_size` attribute) specifies the block size required by the tape drive when writing variable length records. Some SCSI tape drives require that a nonzero block size be specified in their Mode Select data even when writing variable length records. The **Block Size** attribute is set to **0** to indicate variable length records. See the specific tape drive SCSI specification to determine whether or not this is required.

Data Compression

Setting the Data Compression attribute (for the `chdev` command, the `compress` attribute) to `compress=yes` causes the tape drive to be in compress mode, if the drive is capable of compressing data. If so, then the drive writes data to the tape in compressed format so that more data fits on a single tape. Setting this attribute to `no` forces the tape drive to write in native mode (noncompressed). Read operations are not affected by the setting of this attribute. The default setting is `yes`.

Autoloader

Setting the Autoloader attribute (for the `chdev` command, the `autoload` attribute) to `autoload=yes` causes Autoloader to be active, if the drive is so equipped. If so, and another tape is available in the loader, any read or write operation that advances the tape to the end is automatically continued on the next tape. Tape drive commands that are restricted to a single tape cartridge are unaffected. The default setting is `yes`.

Retry Delay

The Retry Delay attribute sets the number of seconds that the system waits after a command has failed before reissuing the command. The system may reissue a failed command up to four times. This attribute applies only to type OST tape drives. The default setting is 45.

Read/Write Timeout

The Read/Write Timeout or Maximum Delay for a **READ/WRITE** attribute sets the maximum number of seconds that the system allows for a read or write command to complete. This attribute applies only to type OST tape drives. The default setting is 144.

Return Error on Tape Change

The Return Error on Tape Change or Reset attribute, when set, causes an error to be returned on

open when the tape drive has been reset or the tape has been changed. A previous operation to the tape drive must have taken place that left the tape positioned beyond beginning of tape upon closing. The error returned is a -1 and errno is set to EIO. Once presented to the application, the error condition is cleared. Also, reconfiguring the tape drive itself will clear the error condition.

Attributes for 2.0 GB 4 mm tape drives (type 4mm2gb)

The following are the attributes for 2.0 GB 4 mm tape drives (type 4mm2gb).

Block size

The default value is 1024.

Device buffers

The general information for this attribute applies to this tape drive type.

Attributes with fixed values

If a tape drive is configured as a 2.0 GB 4 mm tape drive, the attributes for Retension, Reserve Support, Variable Length Block Size, Density Setting #1, and Density Setting #2 have predefined values that cannot be changed. The density settings are predefined because the tape drive always writes in 2.0 GB mode.

Attributes for 4.0 GB 4 mm tape drives (type 4mm4gb)

The following are the attributes for 4.0 GB 4 mm tape drives (type 4mm4gb).

Block size

The default value is 1024.

Device buffers

The general information for this attribute applies to this tape drive type.

Density setting #1 and Density setting #2

The user cannot change the density setting of this drive; the device reconfigures itself automatically depending on the Digital Data Storage (DDS) media type installed, as follows:

Media Type	Device Configuration
DDS	Read-only.
DDS IIII	Read/write in 2.0 GB mode only.
DDS2	Read in either density; write in 4.0 GB mode only.
non-DDS	Not supported; cartridge will eject.

Data compression

The general information for this attribute applies to this tape drive type.

Attributes with fixed values

If a tape drive is configured as a 4.0 GB 4 mm tape drive, the Retension, Reserve Support, Variable Length Block Size, Density Setting #1, and Density Setting #2 attributes have predefined values that cannot be changed.

Attributes for 2.3 GB 8 mm tape drives (type 8mm)

The following are attributes for 2.3 GB 8 mm tape drives (type 8mm).

Block size

The default value is 1024. A smaller value reduces the amount of data stored on a tape.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

The general information for this attribute applies to this tape drive type.

Attributes with fixed values

If a tape drive is configured as a 2.3 GB 8mm tape drive, the Retension, Reserve Support, Variable Length Block Size, Data Compression, Density Setting #1, and Density Setting #2 attributes have predefined values which cannot be changed. The density settings are predefined because the tape drive always writes in 2.3 GB mode.

Attributes for 5.0GB 8mm tape drives (type 8mm5gb)

The following are the attributes for 5.0GB 8mm tape drives (type 8mm5gb).

Block size

The default value is 1024. If a tape is being written in 2.3 GB mode, a smaller value reduces the amount of data stored on a tape.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The following settings apply:

Setting	Meaning
140	5 GB mode (compression capable)
21	5 GB mode noncompressed tape
20	2.3 GB mode
0	Default (5.0 GB mode)

The default values are 140 for Density Setting #1, and 20 for Density Setting #2. A value of 21 for Density Setting #1 or #2 permits the user to read or write a noncompressed tape in 5 GB mode.

Data compression

The general information for this attribute applies to this tape drive type.

Attributes with fixed values

If a tape drive is configured as a 5.0 GB 8 mm tape drive, the Retension, Reserve Support, and Variable Length Block Size attributes have predefined values which cannot be changed.

Attributes for 20000 MB 8 mm tape drives (self-configuring)

The following are attributes for 20000 MB 8 mm tape drives (self-configuring).

Block size

The default value is 1024.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The drive can read and write data cartridges in 20.0 GB format. During a Read command, the drive automatically determines which format is written on tape. During a Write, the Density Setting determines which data format is written to tape.

The following settings apply:

Setting	Meaning
39	20 GB mode (compression capable)
0	Default (20.0 GB mode)

The default value is 39 for Density Setting #1 and Density Setting #2.

Data compression

The general information for this attribute applies to this tape drive type.

Attributes with fixed values

If a tape drive is configured as a 20.0 GB 8 mm tape drive, the Retention, Reserve Support, and Variable Length Block Size attributes have predefined values which cannot be changed.

Attributes for 35 GB tape drives (type 35gb)

The following are attributes for 35 GB tape drives (type 35gb).

Block size

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

Supported Command	Default Block Size (Bytes)	RECOMMENDATION
BACKUP	32 K or 51.2 K (default)	Uses either 32 K or 51.2 K depending on if "Backup" is by name or not. No change is required.
TAR	10 K	There is an error in the manual that states a 512 K byte block size. Set the Blocking Parameter to -N64 .
MKSYSB	See BACKUP	MKSYSB uses the BACKUP command. No change is required.
DD	n/a	Set the Blocking Parameter to bs=32K .
CPIO	n/a	Set the Blocking Parameter to -C64 .

Note: You should be aware of the capacity and throughput when you select a blocksize. Small blocksizes have a significant impact on performance and a minimal impact on capacity. The capacities of the 2.6 GB format (density) and 6.0 GB format (density) are significantly impacted when you use smaller than the recommended blocksizes. As an example: using a blocksize of 1024 bytes to backup 32 GB of data takes approximately 22 hours. Backing up the same 32 GB of data using a blocksize of 32 K Bytes takes approximately 2 hours.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The following chart shows the Supported Data Cartridge type and Density Settings (in decimal and hex) for the IBM 7205-311 Tape Drive. When you perform a Restore (Read) Operation, the tape drive automatically sets the density to match the written density. When you perform a Backup Operation (Write), you must set the Density Setting to match the Data Cartridge that you are using.

Supported Data Cartridges	Native Capacity	Compressed Data Capacity	Web-based System Manager or SMIT Density Setting	HEX Density Setting
DLTtape III	2.6 GB	2.6 GB (No Compression)	23	17h
	6.0 GB	6.0 GB (No Compression)	24	18h
	10.0 GB	20.0 GB (Default for drive)	25	19h
DLTtapellxt	15.0 GB	30.6 GB (Default for drive)	25	19h

Supported Data Cartridges	Native Capacity	Compressed Data Capacity	Web-based System Manager or SMIT Density Setting	HEX Density Setting
DLTtapeIV	20.0 GB	40.0 GB	26	1Ah
	35.0 GB	70.0 GB (Default for drive)	27	1Bh

Note: If you request an unsupported Native Capacity for the Data Cartridge, the drive defaults to the highest supported capacity for the Data Cartridge that is loaded into the drive.

Data compression

The actual compression depends on the type of data being that is being written (see above table). A Compression Ratio of 2:1 is assumed for this Compressed Data Capacity.

Attributes with fixed values

The general information for this attribute applies to this tape drive type.

Attributes for 150 MB 1/4-inch tape drives (type 150mb)

The following are attributes for 150 MB 1/4-inch tape drives (type 150mb).

Block size

The default block size is 512. The only other valid block size is 0 for variable length blocks.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you wish to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The following settings apply:

Setting	Meaning
16	QIC-150
15	QIC-120
0	Default (QIC-150), or whatever was the last density setting by a using system.

The default values are 16 for Density Setting #1, and 15 for Density Setting #2.

Attributes with fixed values

If a tape drive is configured as a 150 MB 1/4-inch tape drive, attributes for Extended File Marks, Reserve Support, Variable Length Block Size, and Data Compression have predefined values which cannot be changed.

Attributes for 525 MB 1/4-inch tape drives (type 525mb)

The following are attributes for 525 MB 1/4-inch tape drives (type 525mb).

Block size

The default block size is 512. The other valid block sizes are 0 for variable length blocks, and 1024.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you want to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension

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.

Density setting #1 and density setting #2

The following settings apply:

Setting	Meaning
17	QIC-525*
16	QIC-150
15	QIC-120
0	Default (QIC-525), or whatever was the last density setting by a using system.

* QIC-525 is the only mode that supports the 1024 block size.

The default values are 17 for Density Setting #1, and 16 for Density Setting #2.

Attributes with fixed values

If a tape drive is configured as a 525 MB 1/4-inch tape drive, the Extended File Marks, Reserve Support, Variable Length Block Size, and Data Compression attributes have predefined values which cannot be changed.

Attributes for 1200 MB 1/4-inch tape drives (type 1200mb-c)

The following are attributes for 1200 MB 1/4-inch tape drives (type 1200mb-c).

Block size

The default block size is 512. The other valid block sizes are 0 for variable length blocks, and 1024.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you wish to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The following settings apply:

Setting	Meaning
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120

Setting	Meaning
0	Default (QIC-1000), or whatever was the last density setting by a using system.

* QIC-525 and QIC-1000 are the only modes that support the 1024 block size.

The default values are 21 for Density Setting #1, and 17 for Density Setting #2.

Attributes with fixed values

If a tape drive is configured as a 1200 MB 1/4-inch tape drive, the Extended File Marks, Reserve Support, Variable Length Block Size, and Data Compression attributes have predefined values which cannot be changed.

Attributes for 12000 MB 4 mm tape drives (self-configuring)

The following are attributes for 12000 MB 4 mm tape drives (self-configuring).

Block size

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

Supported Command	Default Block Size (Bytes)	RECOMMENDATION
BACKUP	32 K or 51.2 K (default)	Will use either 32 K or 51.2 K depending on if "Backup" is by name or not. No change is required.
TAR	10 K	There is an error in the manual that states a 512 K byte block size. Set the Blocking Parameter to -N64 .
MKSYSB	See BACKUP	MKSYSB uses the BACKUP command. No change is required.
DD	n/a	Set the Blocking parameter to bs=32K.
CPIO	n/a	Set the Blocking parameter to -C64.

Note: You should be aware of the capacity and throughput when you select a blocksize. Small blocksizes have a significant impact on performance and a minimal impact on capacity.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The following chart shows the supported data cartridge type and density settings (in decimal and hex) for the IBM 12000 MB 4 mm tape drive. When you perform a restore (read) operation, the tape drive automatically sets the density to match the written density. When you perform a backup operation (write), you must set the density setting to match the data cartridge you are using.

Supported Data Cartridges	Native Capacity	Compressed Data Capacity	Web-based System	
			Manager or SMIT Density Setting	HEX Density Setting
DDS III	2.0 GB	4.0 GB	19	13h
DDS2	4.0 GB	8.0 GB	36	24h
DDS3	12.0 GB	24.0 GB	37	25h

Note: If you request an unsupported native capacity for the data cartridge, the drive defaults to the highest supported capacity for the data cartridge that is loaded into the drive.

Data compression

The actual compression depends on the type of data being that is being written (see above table). A compression ratio of 2:1 is assumed for this compressed data capacity.

Attributes with fixed values

The general information for this attribute applies to this tape drive type.

Attributes for 13000 MB 1/4-inch tape drives (self-configuring)

The following are the attributes for 13000 MB 1/4-inch tape drives (self-configuring).

Block size

The default block size is 512. The other valid block sizes are 0 for variable length blocks, and 1024.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

Writing to a 1/4-inch tape can only occur at the beginning of tape (BOT) or after blank tape is detected. If data exists on the tape, you cannot overwrite the data except at BOT. If you wish to add data to a tape that has been written and then rewound, you must space forward until the next file mark is detected, which causes the system to return an error. Only then can you start writing again.

Retension

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The following settings apply:

Setting	Meaning
33	QIC-5010-DC*
34	QIC-2GB*
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120
0	Default (QIC-5010-DC)*

* QIC-525, QIC-1000, QIC-5010-DC, and QIC-2GB are the only modes that support the 1024 block size.

The default values are 33 for Density Setting #1, and 34 for Density Setting #2.

Attributes with fixed values

If a tape drive is configured as a 13000 MB 1/4-inch tape drive, the **Extended File Marks**, **Reserve Support**, and **Variable Length Block Size** attributes have predefined values which cannot be changed.

Attributes for 1/2-inch 9-track tape drives (type 9trk)

The following are the 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 that cannot be changed.

Attributes for 3490e 1/2-inch cartridge (type 3490e)

The following are attributes for the 3490e 1/2-inch cartridge (type 3490e).

Block size

The default block size is 1024. This drive features a high data transfer rate, and block size can be critical to efficient operation. Larger block sizes can greatly improve operational speeds, and in general, the largest possible block size should be used.

Note: Increasing the block value can cause incompatibilities with other programs on your system. If this occurs, you receive the following error message while running those programs:

A system call received a parameter that is not valid.

Device buffers

The general information for this attribute applies to this tape drive type.

Compression

The general information for this attribute applies to this tape drive type.

Autoloader

This drive features a tape sequencer, an autoloader that sequentially loads and ejects a series of tape cartridges from the cartridge loader. For this function to operate correctly, the front panel switch should be in the AUTO position and the Autoloader attribute must be set to yes.

Attributes for other SCSI tapes (type ost)

The following are attributes for other SCSI tapes (type ost).

Block size

The system default is 512, but this should be adjusted to the default block size for your tape drive. Typical values are 512 and 1024. 8 mm and 4 mm tape drives usually use 1024 and waste space on the tape if the block size attribute is left at 512. A value of 0 indicates variable block size on some drives.

Device buffers

The general information for this attribute applies to this tape drive type.

Extended file marks

The general information for this attribute applies to this tape drive type.

Density setting #1 and density setting #2

The default value is 0 for both of these settings. Other values and their meanings vary for different tape drives.

Reserve support

The default value is no. This can be set to yes, if the drive supports reserve/release commands. If you are unsure, no is a safer value.

Variable length block size

The default variable length block size value is 0. Nonzero values are used primarily on quarter inch cartridge (QIC) drives. See the SCSI specification for the particular tape drive for advice.

Retry delay

This attribute applies exclusively to type ost tape drives.

Read/write timeout

This attribute applies exclusively to type ost tape drives.

Attributes with fixed values

If a tape drive is configured as an Other SCSI tape drive, the attributes for Extended File Marks, Retension, and Data Compression have predefined values which cannot be changed.

MPIO tape attributes

MPIO-supported tape devices will have additional attributes listed under MPIO device attributes. See Chapter 23, "Multiple Path I/O," on page 303.

Special files for tape drives

There are several special files associated with each tape drive known to the operating system.

Writing to and reading from files on tapes is done by using **rmt** special files. These special files are **/dev/rmt***, **/dev/rmt*.1**, **/dev/rmt*.2**, through **/dev/rmt*.7**. The **rmt*** is the logical name of a tape drive, such as **rmt0**, **rmt1**, and so on.

By selecting one of the special files associated with a tape drive, you make choices about how the I/O operations related to the tape drive will be performed.

Density

You can select whether to write with the tape drive Density Setting #1 or with the tape drive Density Setting #2. The values for these density settings are part of the attributes of the tape drive. Because it is customary to set Density Setting #1 to the highest possible density for the tape drive and Density Setting #2 to the next highest possible density for the tape drive, special files that use Density Setting #1 are sometimes referred to as high density and special files that use Density Setting #2 sometimes are referred to as low density, but this view is not always correct. When reading from a tape, the density setting is ignored.

Rewind-on-Close

You can select whether the tape is rewound when the special file referring to the tape drive is closed. If rewind-on-close is selected, the tape is positioned at the beginning of the tape when the file is closed.

Retension-on-Open

You can select whether the tape is retensioned when the file is opened. Retensioning means winding to the end of the tape and then rewinding to the beginning of the tape to reduce errors. If retension-on-open is selected, the tape is positioned at the beginning of the tape as part of the open process.

The following table shows the names of the **rmt** special files and their characteristics.

Special File	Rewind on Close	Retension on Open	Density Setting
/dev/rmt*	Yes	No	#1
/dev/rmt*.1	No	No	#1
/dev/rmt*.2	Yes	Yes	#1
/dev/rmt*.3	No	Yes	#1
/dev/rmt*.4	Yes	No	#2
/dev/rmt*.5	No	No	#2
/dev/rmt*.6	Yes	Yes	#2

Special File	Rewind on Close	Retention on Open	Density Setting
<code>/dev/rmt*.7</code>	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 Retention-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 Retention-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 Retention-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.

Chapter 25. Login names, system IDs, and passwords

The operating system must know who you are in order to provide you with the correct environment.

To identify yourself to the operating system, log in by typing your *login name* (also known as your user ID or user name) and a *password*. Passwords are a form of security. People who know your login name cannot log in to your system unless they know your password.

If your system is set up as a multiuser system, each authorized user will have an account, password, and login name on the system. The operating system keeps track of the resources used by each user. This is known as *system accounting*. Each user will be given a private area in the storage space of the system, called the *file system*. When you log in, the file system appears to contain only your files, although there are thousands of other files on the system.

It is possible to have more than one valid login name on a system. If you want to change from one login name to another, you do not have to log out of the system. Rather, you can use the different login names simultaneously in different shells or consecutively in the same shell without logging out. In addition, if your system is part of a network with connections to other systems, you can log in to any of the other systems where you have a login name. This is referred to as a *remote login*.

When you have finished working on the operating system, log out to ensure that your files and data are secure.

Logging in to the operating system

To use the operating system, your system must be running, and you must be logged in. When you log in to the operating system, you identify yourself to the system and allow the system to set up your environment.

Your system might be set up so that you can only log in during certain hours of the day and on certain days of the week. If you attempt to log in at a time other than the time allowed, access will be denied. Your system administrator can verify your login times.

You log in at the login prompt. When you log in to the operating system, you are automatically placed into your home directory (also called your *login directory*).

After your system is turned on, log in to the system to start a session.

1. Type your login name following the **login:** prompt:

```
login: LoginName
```

For example, if your login name is denise:

```
login: denise
```

2. If the **password:** prompt is displayed, type your password. (The screen does not display your password as you type it.)

```
password: [your password]
```

If the password prompt is not displayed, you have no password defined; you can begin working in the operating system.

If your machine is not turned on, do the following before you log in:

1. Set the power switches of each attached device to On.
2. Start the system unit by setting the power switch to On (I).

3. Look at the three-digit display. When the self-tests complete without error, the three-digit display is blank.

If an error requiring attention occurs, a three-digit code remains, and the system unit stops. See your system administrator for information about error codes and recovery.

When the self-tests complete successfully, a login prompt similar to the following is displayed on your screen:

```
login:
```

After you have logged in, depending on how your operating system is set up, your system will start up in either a command line interface (shell) or a graphical interface (for example, AIXwindows or Common Desktop Environment (CDE)).

If you have questions concerning the configuration of your password or user name, please consult your system administrator.

Logging in more than one time (login command)

If you are working on more than one project and want to maintain separate accounts, you can have more than one concurrent login. You do this by using the same login name or by using different login names to log in to your system.

Note: Each system has a maximum number of login names that can be active at any given time. This number is determined by your license agreement and varies between installations.

For example, if you are already logged on as `denise1` and your other login name is `denise2`, at the prompt, type the following:

```
login denise2
```

If the **password:** prompt is displayed, type your password. (The screen does not display your password as you type it.) You now have two logins running on your system.

See the **login** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Becoming another user on a system (su command)

You can change the user ID associated with a session (if you know that user's login name) by using the **su** (switch user) command.

For example, if you want to switch to become user `joyce`, at the prompt, type the following:

```
su joyce
```

If the **password:** prompt is displayed, type the password for user `joyce`. Your user ID is now `joyce`. If you do not know the password, the request is denied.

To verify that your user ID is `joyce`, use the **id** command. For more information on the **id** command, see "Command for displaying user IDs (id command)" on page 337.

See the **su** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Suppressing login messages

After a successful login, the **login** command displays the message of the day, the date and time of the last successful and unsuccessful login attempts for this user, and the total number of unsuccessful login attempts for this user since the last change of authentication information (usually a password). You can suppress these messages by including a **.hushlogin** file in your home directory.

At the prompt in your home directory, type the following:

```
touch .hushlogin
```

The **touch** command creates the empty file named **.hushlogin** if it does not already exist. The next time you log in, all login messages will be suppressed. You can instruct the system to retain only the message of the day, while suppressing other login messages.

See the **touch** command for the complete syntax.

Logging out of the operating system (exit and logout commands)

To log out of the operating system, do one of the following at the system prompt.

Press the end-of-file control-key sequence (Ctrl-D keys).

OR

Type `exit`.

OR

Type `logout`.

After you log out, the system displays the **login:** prompt.

Shutdown command

If you have root user authority, you can use the **shutdown** command to stop the system. If you are not authorized to use the **shutdown** command, simply log out of the operating system and leave it running.

Attention: Do not turn off the system without first shutting down. Turning off the system ends all processes running on the system. If other users are working on the system, or if jobs are running in the background, data might be lost. Perform proper shutdown procedures before you stop the system.

At the prompt, type the following:

```
shutdown
```

When the **shutdown** command completes and the operating system stops running, you receive the following message:

```
....Shutdown completed....
```

See the **shutdown** command for the complete syntax.

Displaying your login name (whoami and logname commands)

When you have more than one concurrent login, it is easy to lose track of the login names or, in particular, the login name that you are currently using. You can use the **whoami** and **logname** commands to display this information.

Using the whoami command

To determine which login name is being used, at the prompt, type the following:

```
whoami
```

The system displays information similar to the following:

```
denise
```

In this example, the login name being used is denise.

See the **whoami** command in the *AIX 5L Version 5.3 Commands Reference, Volume 6* for the complete syntax.

Using the who am i command

A variation of the **who** command, the **who am i** command, allows you to display the login name, terminal name, and time of the login. At the prompt, type the following:

```
who am i
```

The system displays information similar to the following:

```
denise pts/0 Jun 21 07:53
```

In this example, the login name is denise, the name of the terminal is pts/0, and this user logged in at 7:53 a.m. on June 21.

See the **who** command in the *AIX 5L Version 5.3 Commands Reference, Volume 6* for the complete syntax.

Using the logname command

Another variation of the **who** command, the **logname** command displays the same information as the **who** command.

At the prompt, type the following:

```
logname
```

The system displays information similar to the following:

```
denise
```

In this example, the login name is denise.

Displaying the operating system name (uname command)

To display the name of the operating system, use the **uname** command.

For example, at the prompt, type the following:

```
uname
```

The system displays information similar to the following:

```
AIX
```

In this example, the operating system name is AIX.

See the **uname** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Displaying your system name (uname command)

To display the name of your system if you are on a network, use the **uname** command with the **-n** flag. Your system name identifies your system to the network; it is not the same as your login ID.

For example, at the prompt, type the following:

```
uname -n
```

The system displays information similar to the following:

```
barnard
```

In this example, the system name is barnard.

See the **uname** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* book for the complete syntax.

Displaying all users who are logged in

To display information about all users currently logged into the local system, use the **who** command.

The following information is displayed: login name, system name, and date and time of login.

Note: This command only identifies logged-in users on the local node.

To display information about who is using the local system node, type the following:

```
who
```

The system displays information similar to the following:

```
joe 1ft/0 Jun 8 08:34
denise pts/1 Jun 8 07:07
```

In this example, the user joe, on terminal 1ft/0, logged in at 8:34 a.m. on June 8.

See the **who** command.

Command for displaying user IDs (id command)

To display the system identifications (IDs) for a specified user, use the **id** command . The system IDs are numbers that identify users and user groups to the system.

The **id** command displays the following information, when applicable:

- User name and real user ID
- Name of the user's group and real group ID
- Name of the user's supplementary groups and supplementary group IDs, if any

For example, at the prompt, type the following:

```
id
```

The system displays information similar to the following:

```
uid=1544(sah) gid=300(build) euid=0(root) egid=9(printq) groups=0(system),10(audit)
```

In this example, the user has user name sah with an ID number of 1544; a primary group name of build with an ID number of 300; an effective user name of root with an ID number of 0; an effective group name of printq with an ID number of 9; and two supplementary group names of system and audit, with ID numbers 0 and 10, respectively.

For example, at the prompt, type the following:

```
id denise
```

The system displays information similar to the following:

```
uid=2988(denise) gid=1(staff)
```

In this example, the user denise has an ID number of 2988 and has only a primary group name of staff with an ID number of 1.

See the **id** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Passwords

A unique password provides some system security for your files.

Your system associates a password with each account. Security is an important part of computer systems because it keeps unauthorized people from gaining access to the system and from tampering with other users' files. Security can also allow some users exclusive privileges to which commands they can use and which files they can access. For protection, some system administrators permit the users access only to certain commands or files.

Password guidelines

You should have a unique password. *Passwords should not be shared.* Protect passwords as you would any other company asset. When creating passwords, make sure they are difficult to guess, but not so difficult that you have to write them down to remember them.

Using obscure passwords keeps your user ID secure. Passwords based on personal information, such as your name or birthday, are poor passwords. Even common words can be easily guessed.

Good passwords have at least six characters and include nonalphabetic characters. Strange word combinations and words purposely misspelled are also good choices.

Note: If your password is so hard to remember that you have to write it down, it is not a good password.

Use the following guidelines when selecting a password:

- Do not use your user ID as a password. Do not use it reversed, doubled, or otherwise modified.
- Do not reuse passwords. The system might be set up to deny the reuse of passwords.
- Do not use any person's name as your password.
- Do not use words that can be found in the online spelling-check dictionary as your password.
- Do not use passwords shorter than six characters.
- Do not use obscene words; they are some of the first ones checked when guessing passwords.
- Do use passwords that are easy to remember, so you won't have to write them down.
- Do use passwords that use both letters and numbers and that have both lowercase and uppercase letters.
- Do use two words, separated by a number, as a password.
- Do use pronounceable passwords. They are easier to remember.

- Do not write passwords down. However, if you must write them down, place them in a physically secure place, such as a locked cabinet.

Changing passwords (passwd command)

To change your password, use the **passwd** command.

1. At the prompt, type the following:

```
passwd
```

If you do not already have a password, skip step 2.

2. The following prompt displays:

```
Changing password for UserID  
UserID's Old password:
```

This request keeps an unauthorized user from changing your password while you are away from your system. Type your current password, and press Enter.

3. The following prompt is displayed:

```
UserID's New password:
```

Type the new password you want, and press Enter.

4. The following prompt is displayed, asking you to re-type your new password.

```
Enter the new password again:
```

This request protects you from setting your password to a mistyped string that you cannot re-create.

See the **passwd** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Password nullification (passwd command)

If you do not want to type a password each time you log in, set your password to null (blank).

To set your password to null, type the following:

```
passwd
```

When you are prompted for the new password, press Enter or Ctrl-D.

The **passwd** command does not prompt again for a password entry. A message verifying the null password displays.

See the **passwd** command for more information and the complete syntax.

Command summary for login names, system IDs, and passwords

Commands are available for working with login names, system IDs, and passwords.

Login and logout commands

login	Initiates your session
logout	Stops all your processes
shutdown	Ends system operation
su	Changes the user ID associated with a session
touch	Updates the access and modification times of a file, or creates an empty file

User and system identification commands

id	Displays the system identifications of a specified user
logname	Displays login name.
uname	Displays the name of the current operating system
who	Identifies the users currently logged in
whoami	Displays your login name

Password command

passwd	Changes a user's password
---------------	---------------------------

Chapter 26. User environment

Each login name has its own system environment.

The system environment is an area where information that is common to all processes running in a session is stored. You can use several commands to display information about your system.

User environment files and customization procedures

These files and procedures help the user customize the system environment.

System startup files

/etc/profile	System file that contains commands that the system executes when you log in.
/etc/environment	System file that contains variables specifying the basic environment for all processes.
\$HOME/.profile	File in your home directory that contains commands that override the system /etc/profile when you log in. For more information, see .profile file.
\$HOME/.env	File in your home directory that overrides the system /etc/environment and contains variables specifying the basic environment for all processes. For more information, see .env file.

AIXwindows startup files

\$HOME/.xinitrc	File in your home directory that controls the windows and applications that start up when you start AIXwindows. For more information, see .xinitrc file.
\$HOME/.Xdefaults	File in your home directory that controls the visual or behavioral aspect of AIXwindows resources. For more information, see “ .Xdefaults file” on page 428.
\$HOME/.mwmrc	File in your home directory that defines key bindings, mouse button bindings, and menu definitions for your window manager. For more information, see “ .mwmrc file” on page 429.

Customization procedures

PS1	Normal system prompt
PS2	More input system prompt
PS3	Root system prompt
chfont	Changes the font used by a display at system restart
stty	Sets, resets, and reports workstation operating parameters

System devices list (lscfg command)

To display the name, location, and description of each device found in the current configuration, use the **lscfg** command. The list is sorted by device location.

For example, to list the devices configured in your system, at the prompt, type the following:

```
lscfg
```

The system displays output similar to the following:

```
INSTALLED RESOURCE LIST
```

The following resources are installed on your machine.

+/- = Added/Deleted from Diagnostic Test List.

* = NOT Supported by Diagnostics.

```
Model Architecture: chrp
Model Implementation: Multiple Processor, PCI bus
```

```
+ sysplanar0 00-00 CPU Planar
+ fpa0 00-00 Floating Point Processor
+ mem0 00-0A Memory Card
+ mem1 00-0B Memory Card
+ ioplanar0 00-00 I/O Planar
+ rs2320 00-01 RS232 Card
+ tty0 00-01-0-01 RS232 Card Port
- tty1 00-01-0-02 RS232 Card Port
..
..
..
```

The device list is not sorted by device location alone. It is sorted by the parent/child hierarchy. If the parent has multiple children, the children are sorted by device location. If the children have the same device location, they are displayed in the order in which they were obtained by the software. To display information about a specific device, you can use the **-l** flag. For example, to list the information on device **sysplanar0**, at the prompt, type the following:

```
lscfg -l sysplanar0
```

The system displays output similar to the following:

DEVICE	LOCATION	DESCRIPTION
sysplanar0	00-00	CPU Planar

You can also use the **lscfg** command to display vital product data (VPD), such as part numbers, serial numbers, and engineering change levels. For some devices, the VPD is collected automatically and added to the system configuration. For other devices, the VPD is typed manually. An ME preceding the data indicates that the data was typed manually.

For example, to list VPD for devices configured in your system, at the prompt, type the following:

```
lscfg -v
```

The system displays output similar to the following:

```
INSTALLED RESOURCE LIST WITH VPD
```

The following resources are installed in your machine.

```
Model Architecture: chrp
Model Implementation: Multiple Processor, PCI bus
sysplanar0 00-00 CPU Planar
  Part Number.....342522
  EC Level.....254921
  Serial Number.....353535
fpa0 00-00 Floating Point Processor
mem0 00-0A Memory Card
  EC Level.....990221
.
.
.
```

See the **lscfg** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Displaying console names

To write the name of the current console device to standard output (usually your screen), use the **lscons** command.

For example, at the prompt, type the following:

```
lscons
```

The system displays output similar to the following:

```
/dev/lft0
```

See the **lscons** command for the complete syntax.

Displaying the terminal name (tty command)

To display the name of your terminal, use the **tty** command.

For example, at the prompt, type the following:

```
tty
```

The system displays information similar to the following:

```
/dev/tty06
```

In this example, `tty06` is the name of the terminal, and `/dev/tty06` is the device file that contains the interface to this terminal.

See the **tty** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Listing available displays (lsdisp command)

To list the displays currently available on your system, providing a display identification name, slot number, display name, and description of each of the displays, use the **lsdisp** command.

For example, to list all available displays, type the following:

```
lsdisp
```

The following is an example of the output. The list displays in ascending order according to slot number.

Name	Slot	Name	Description
ppr0	00-01	POWER_G4	Midrange Graphics Adapter
gda0	00-03	colorgda	Color Graphics Display Adapter
ppr1	00-04	POWER_Gt3	Midrange Entry Graphics Adapter

See the **lsdisp** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Listing available fonts (lsfont command)

To display a list of the fonts available to your display, use the **lsfont** command.

For example, to list all fonts available to the display, type the following:

```
lsfont
```

The following is an example of the output, showing the font identifier, file name, glyph size, and font encoding:

FONT ID	FILE NAME	GLYPH SIZE	FONT ENCODING
0	Erg22.iso1.snf	12x30	IS08859-1
1	Erg11.iso1.snf	8x15	IS08859-1

See the **lsfont** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Listing the current software keyboard map (lskbd command)

To display the absolute path name of the current software keyboard map loaded into the system, use the **lskbd** command.

For example, to list your current keyboard map, type the following:

```
lskbd
```

The following is an example of the listing displayed by the **lskbd** command:

```
The current software keyboard map = /usr/lib/nls/loc/C.lftkeymap
```

Listing available software products (lslpp command)

To display information about software products available on your system, use the **lslpp** command.

For example, to list all the software products in your system, at the system prompt, type the following:

```
lslpp -l -a
```

The following is an example of the output:

```
Fileset          Level State  Description
-----
Path: /usr/lib/objrepos
  X11_3d.gl.dev.obj      APPLIED  AIXwindows/3D GL
                        Development Utilities
Fonts
  X11fnt.oldX.fnt       APPLIED  AIXwindows Miscellaneous
                        X Fonts
X11mEn_US.msg         APPLIED  AIXwindows NL Message
                        files
.
.
.
```

If the listing is very long, the top portion may scroll off the screen. To display the listing one page (screen) at a time, use the **lslpp** command piped to the **pg** command. At the prompt, type the following:

```
lslpp | pg
```

See the **lslpp** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

List of control key assignments for your terminal (stty command)

To display your terminal settings, use the **stty** command. Note especially which keys your terminal uses for control keys.

For example, at the prompt, type the following:

```
stty -a
```

The system displays information similar to the following:

```
.  
.   
.   
intr = ^C; quit = ^\; erase = ^H; kill = ^U; eof = ^D;  
eol = ^@ start = ^Q; stop = ^S; susp = ^Z; dsusp = ^Y;  
reprint = ^R discard = ^O; werase = ^W; lnext = ^V  
.   
.   
.
```

In this example, lines such as `intr = ^C; quit = ^\; erase = ^H;` display your control key settings. The ^H key is the Backspace key, and it is set to perform the erase function.

If the listing is very long, the top portion may scroll off the screen. To display the listing one page (screen) at a time, use the **stty** command piped to the **pg** command. At the prompt, type the following:

```
stty -a | pg
```

See the **stty** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Listing environment variables (env command)

To display your current environment variables, use the **env** command. An environment variable that is accessible to all your processes is called a *global variable*.

For example, to list all environment variables and their associated values, type the following:

```
env
```

The following is an example of the output:

```
TMPDIR=/usr/tmp  
myid=denise  
LANG=En_US  
UNAME=barnard  
PAGER=/bin/pg  
VISUAL=vi  
PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/u/bin1  
MAILPATH=/usr/mail/denise?denise has mail !!!  
MAILRECORD=/u/denise/.Outmail  
EXINIT=set beautify noflash nomesg report=1 showmode showmatch  
EDITOR=vi  
PSCH=>  
HISTFILE=/u/denise/.history  
LOGNAME=denise  
MAIL=/usr/mail/denise  
PS1=denise@barnard:${PWD}>  
PS3=#  
PS2=>  
epath=/usr/bin  
USER=denise  
SHELL=/bin/ksh  
HISTSIZE=500  
HOME=/u/denise  
FCEDIT=vi  
TERM=1ft  
MAILMSG=**YOU HAVE NEW MAIL. USE THE mail COMMAND TO SEE YOUR PWD=/u/denise  
ENV=/u/denise/.env
```

If the listing is very long, the top portion scrolls off the screen. To display the listing one page (screen) at a time, use the **env** command piped to the **pg** command. At the prompt, type the following:

```
env | pg
```

See the **env** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Displaying an environment variable value (printenv command)

To display the values of environment variables, use the **printenv** command.

If you specify the **Name** parameter, the system only prints the value associated with the variable you requested. If you do not specify the **Name** parameter, the **printenv** command displays all current environment variables, showing one **Name =Value** sequence per line.

For example, to find the current setting of the *MAILMSG* environment variable, type the following:

```
printenv MAILMSG
```

The command returns the value of the *MAILMSG* environment variable. For example:

```
YOU HAVE NEW MAIL
```

See the **printenv** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Bidirectional languages (aixterm command)

The **aixterm** command supports Arabic and Hebrew, which are bidirectional languages.

Bidirectional languages have the ability to be read and written in two directions: from left to right and from right to left. You can work with Arabic and Hebrew applications by opening a window specifying an Arabic or Hebrew locale.

See the **aixterm** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Command summary for user environment and system information

The following are commands for user environment and system information.

aixterm	Enables you work with bidirectional languages
env	Displays the current environment or sets the environment for the execution of a command
lscfg	Displays diagnostic information about a device
lscons	Displays the name of the current console
lsdisp	Lists the displays currently available on the system
lsfont	Lists the fonts available for use by the display
lskbd	Lists the keyboard maps currently loaded in the system
lslpp	Lists software products
printenv	Displays the values of environment variables
stty	Displays system settings
tty	Displays the full path name of your terminal

Chapter 27. Common Desktop Environment

With the Common Desktop Environment (CDE), you can access networked devices and tools without having to be aware of their location. You can exchange data across applications by simply dragging and dropping objects.

Many tasks that previously required complex command line syntax can be done more easily and similarly from platform to platform. For example, you can centrally configure and distribute applications to users. You can also centrally manage the security, availability, and interoperability of applications for the users you support.

Note: The Common Desktop Environment (CDE) 1.0 Help volumes, Web-based documentation, and hardcopy manuals may refer to the desktop as the Common Desktop Environment, the AIXwindows desktop, CDE 1.0, or the desktop.

Starting and stopping the Common Desktop Environment

You can set up the system so that Common Desktop Environment starts automatically when you start the system, or you can start Common Desktop Environment manually.

You must log in as root to perform each of these tasks.

Enabling and disabling desktop autostart

You might find it more convenient to set up your system to start Common Desktop Environment automatically when the system is turned on.

You can do this through the Web-based System Manager (type `wsm`, and select **System**), through the System Management Interface Tool (SMIT), or from the command line.

Prerequisites

You must have root user authority to enable or disable desktop autostart.

Consult the following table to determine how to enable or disable desktop autostart.

Starting and Stopping the Common Desktop Environment Automatically

Task	SMIT Fast Path	Command or File
Enabling the desktop autostart ¹	<code>smit dtconfig</code>	<code>/usr/dt/bin/dtconfig -e</code>
Disabling the desktop autostart ¹	<code>smit dtconfig</code>	<code>/usr/dt/bin/dtconfig -d</code>

Note: Restart the system after completing this task.

Starting the Common Desktop Environment manually

Use this procedure to start the Common Desktop Environment manually.

1. Log in to your system as root.
2. At the command line, type the following:

```
/usr/dt/bin/dtlogin -daemon
```

A **Desktop Login** screen is displayed. When you log in, a desktop session starts.

Stopping the Common Desktop Environment manually

When you manually stop the Login Manager, all X Servers and desktop sessions that the Login Manager started are stopped.

1. Open a terminal emulator window, and log in as root.
2. Obtain the process ID of the Login Manager by typing the following:

```
cat /var/dt/Xpid
```

3. Stop the Login Manager by typing:

```
kill -term process_id
```

Desktop profile modification

When you log in to the desktop, the shell environment file (**.profile** or **.login**) is not automatically read. The desktop runs the X Server before you log in, so the function provided by the **.profile** file or the **.login** file must be provided by the desktop's Login Manager.

User-specific environment variables are set in **/Home Directory/.dtprofile**. A template for this file is located in **/usr/dt/config/sys.dtpfile**. Place variables and shell commands in **.dtprofile** that apply only to the desktop. Add lines to the end of the **.dtprofile** to incorporate the shell environment file.

System-wide environment variables can be set in the Login Manager configuration files. For details on configuring environment variables, see the *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*.

Adding and removing displays and terminals for the Common Desktop Environment

You can add and remove displays and terminals for the Common Desktop Environment.

The Login Manager can be started from a system with a single local bitmap or graphics console. Many other situations are also possible, however (see the following figure). You can start the Common Desktop Environment from:

- Local consoles
- Remote consoles
- Bitmap and character-display X terminal systems running on a host system on the network

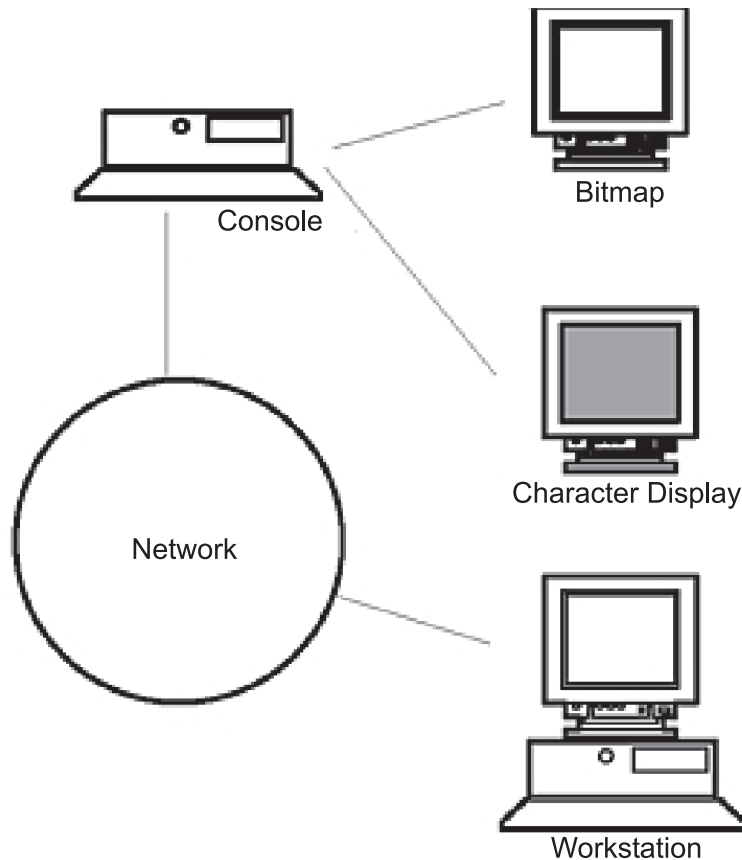


Figure 15. CDE Interface Points. This illustration shows the connection points between a console, a network, a bitmap display, a character display, and a workstation.

An X terminal system consists of a display device, keyboard, and mouse that runs only the X Server. Clients, including the Common Desktop Environment, are run on one or more host systems on the networks. Output from the clients is directed to the X terminal display.

The following Login Manager configuration tasks support many possible configurations:

- Removing a local display
- Adding an ASCII or character-display terminal

To use a workstation as an X terminal, type the following at a command line:

```
/usr/bin/X11/X -query hostname
```

The X Server of the workstation acting as an X terminal must:

- Support XDMCP and the **-query** command-line option.
- Provide xhost permission (in **/etc/X*.hosts**) to the terminal host.

To remove a local display, remove its entry in the **Xservers** file in the **/usr/dt/config** directory.

A *character-display terminal*, or *ASCII terminal*, is a configuration in which the terminal is not a bitmap device.

To add an ASCII or character-display console if no bitmap display is present, perform the following steps:

1. If the **/etc/dt/config/Xservers** file does not exist, copy the **/usr/dt/config/Xservers** file to the **/etc/dt/config** directory.

2. If you have to copy the **Xservers** file to **/etc/dt/config**, change or add the `Dtlogin.servers:` line in **/etc/dt/config/Xconfig** to:

```
Dtlogin*servers: /etc/dt/config/Xservers
```

3. Comment out the line in **/etc/dt/config/Xservers** that starts the X Server.

```
# * Local local@console /path/X :0
```

This disables the **Login Option Menu**.

4. Read the Login Manager configuration files.

To add a character-display console if a bitmap display exists, perform the following steps:

1. If the **/etc/dt/config/Xservers** file does not exist, copy the **/usr/dt/config/Xservers** file to the **/etc/dt/config** directory.

2. If you have to copy the **Xservers** file to **/etc/dt/config**, change or add the `Dtlogin.servers:` line in **/etc/dt/config/Xconfig** to:

```
Dtlogin*servers: /etc/dt/config/Xservers
```

3. Edit the line in **/etc/dt/config/Xservers** that starts the X Server to:

```
* Local local@none /path/X :0
```

4. Read the Login Manager configuration files.

Display device customization for the Common Desktop Environment

You can configure the Common Desktop Environment Login Manager to run on systems with two or more display devices.

When a system includes multiple displays, the following configuration requirements must be met:

- A server must be started on each display.
- No Windows Mode must be configured for each display.

It might be necessary or desirable to use different `dtlogin` resources for each display.

It might also be necessary or desirable to use different system-wide environment variables for each display device.

Starting the server on each display device

Start a server on each display device using this procedure.

1. If the **/etc/dt/config/Xservers** file does not exist, copy the **/usr/dt/config/Xservers** file to the **/etc/dt/config** directory.

2. If you have to copy the **Xservers** file to **/etc/dt/config**, change the `Dtlogin.servers:` line in **/etc/dt/config/Xconfig** to:

```
Dtlogin*servers: /etc/dt/config/Xservers
```

3. Edit **/etc/dt/config/Xservers** to start an X Server on each display device.

The general syntax for starting the server is:

```
DisplayName DisplayClass DisplayType [ @ite ] Command
```

Only displays with an associated Internal Terminal Emulator (ITE) can operate in *No Windows Mode*. No Windows Mode temporarily disables the desktop for the display and runs a *getty* process if one is not already started. A *getty* process is a UNIX program that sets terminal type and is used in the login process.

This allows you to log in and perform tasks not possible under the Common Desktop Environment. When you log out, the desktop is restarted for the display device. If a `getty` process is not already running on a display device, Login Manager starts one when No Windows Mode is initiated.

In the default configuration, when `ite` is omitted, `display:0` is associated with the ITE (`/dev/console`).

Setting up a different display as ITE

Use this procedure to set up a different display as ITE.

To specify a different display as ITE:

- On the ITE display, set ITE to the character device.
- On all other displays, set ITE to none.

The following examples show entries in the **Xserver** file which start a server on three local displays on `sysaaa:0`. Display `:0` will be the console (ITE).

```
sysaaa:0 Local local /usr/bin/X11/X :0
sysaaa:1 Local local /usr/bin/X11/X :1
sysaaa:2 Local local /usr/bin/X11/X :2
```

On host `sysbbb`, the bitmap display `:0` is not the ITE; the ITE is associated with device `/dev/tty1`. The following entries in the **Xserver** file start servers on the two bitmap displays with No Windows Mode enabled on `:1`.

```
sysaaa:0 Local local@none /usr/bin/X11/X :0
sysaaa:1 Local local@tty1 /usr/bin/X11/X :1
```

Setting up the display name in Xconfig

You cannot use regular `hostname:0` syntax for the display name in `/etc/dt/config/Xconfig`.

To specify the display name in Xconfig:

- Use an underscore in place of the colon.
- In a fully qualified host name, use underscores in place of the periods.

The following example shows the setup of the display name in Xconfig:

```
Dtlogin.claaa_0.resource: value
Dtlogin.sysaaa_prsm_ld_edu_0.resource: value
```

Using different Login Manager resources for each display

To use different Login Manager resources for each display, perform the following steps:

1. If the `/etc/dt/config/Xconfig` file does not exist, copy the `/usr/dt/config/Xconfig` file to the `/etc/dt/config` directory.
2. Edit the `/etc/dt/config/Xconfig` file to specify a different resource file for each display. For example:

```
Dtlogin.DisplayName.resources: path/file
```

where *path* is the path name of the Xresource files to be used, and *file* is the file name of the Xresource files to be used.

3. Create each of the resource files specified in the **Xconfig** file. A language-specific Xresources file is installed in `/usr/dt/config/<LANG>`.
4. In each file, place the `dtlogin` resources for that display.

The following example shows lines in the **Xconfig** file which specify different resource files for three displays:

```
Dtlogin.sysaaa_0.resources: /etc/dt/config/Xresources0
Dtlogin.sysaaa_1.resources: /etc/dt/config/Xresources1
Dtlogin.sysaaa_2.resources: /etc/dt/config/Xresources2
```

Running different scripts for each display

Use this procedure to run a particular script for a specific display.

1. If the **/etc/dt/config/Xconfig** file does not exist, copy the **/usr/dt/config/Xconfig** file to the **/etc/dt/config** directory.
2. Use the startup, reset, and setup resources in **/etc/dt/config/Xconfig** to specify different scripts for each display (these files are run instead of the **Xstartup**, **Xreset**, and **Xsetup** files):

```
Dtlogin*DisplayName*startup: /path/file
Dtlogin*DisplayName*reset: /path/file
Dtlogin*DisplayName*setup: /path/file
```

where *path* is the path name of the file to be used, and *file* is the file name of the file to be used. The startup script is run as root after the user has logged in, before the Common Desktop Environment session is started.

The script **/usr/dt/config/Xreset** can be used to reverse the setting made in the **Xstartup** file. The **Xreset** file runs when the user logs out.

The following example shows lines in the **Xconfig** file which specify different scripts for two displays:

```
Dtlogin.sysaaa_0*startup: /etc/dt/config/Xstartup0
Dtlogin.sysaaa_1*startup: /etc/dt/config/Xstartup1
Dtlogin.sysaaa_0*setup: /etc/dt/config/Xsetup0
Dtlogin.sysaaa_1*setup: /etc/dt/config/Xsetup1
Dtlogin.sysaaa_0*reset: /etc/dt/config/Xreset0
Dtlogin.sysaaa_1*reset: /etc/dt/config/Xreset1
```

Setting different system-wide environment variables for each display

Use this procedure to customize system-wide environment variables to each display.

To set different system-wide environment variables for each display:

1. If the **/etc/dt/config/Xconfig** file does not exist, copy the **/usr/dt/config/Xconfig** file to the **/etc/dt/config** directory.
2. Set the environment resource in **/etc/dt/config/Xconfig** separately for each display:

```
Dtlogin*DisplayName*environment: value
```

The following rules apply to environment variables for each display:

- Separate variable assignments with a space or tab.
- Do not use the environment resource to set **TZ** and **LANG**.
- There is no shell processing within the **Xconfig** file.

The following example shows lines in the **Xconfig** file which set variables for two displays:

```
Dtlogin*syshere_0*environment:EDITOR=vi SB_DISPLAY_ADDR=0xB00000
Dtlogin*syshere_1*environment: EDITOR=emacs \
    SB_DISPLAY_ADDR=0xB00000
```

Chapter 28. Commands and processes

A *command* is a request to perform an operation or run a program. A *process* is a program or command that is actually running on the computer.

You use commands to tell the operating system what task you want it to perform. When commands are entered, they are deciphered by a command interpreter (also known as a *shell*), and that task is processed.

The operating system can run many different processes at the same time.

The operating system allows you to manipulate the input and output (I/O) of data to and from your system by using specific I/O commands and symbols. You can control input by specifying the location from which to gather data. For example, you can specify to read input entered on the keyboard (standard input) or to read input from a file. You can control output by specifying where to display or store data. For example, you can specify to write output data to the screen (standard output) or to write it to a file.

Commands

Some commands can be entered simply by typing one word. It is also possible to combine commands so that the output from one command becomes the input for another command.

Combining commands so that the output from one command becomes the input for another command is known as *pipng*. For more information on piping, see “Shell features” on page 435.

Flags further define the actions of commands. A *flag* is a modifier used with the command name on the command line, usually preceded by a dash.

Commands can also be grouped together and stored in a file. These files are known as *shell procedures* or *shell scripts*. Instead of executing the commands individually, you execute the file that contains the commands. For more information on scripts and procedures, see “Creating and running a shell script” on page 439.

To enter a command, type the command name at the prompt, and press Enter.

```
$ CommandName
```

Command syntax and command names

Although some commands can be entered by simply typing one word, other commands use flags and parameters. Each command has a syntax that designates both the required and optional flags and parameters.

The general format for a command is as follows:

```
CommandName flag(s) parameter(s)
```

The following are some general rules about commands:

- Spaces between commands, flags, and parameters are significant.
- Two commands can be entered on the same line by separating the commands with a semicolon (;). For example:

```
$ CommandOne;CommandTwo
```

The shell runs the commands sequentially.

- Commands are case-sensitive. The shell distinguishes between uppercase and lowercase letters. To the shell, `print` is not the same as `PRINT` or `Print`.

- A very long command can be entered on more than one line by using the backslash (\) character. A backslash signifies line continuation to the shell. The following example is one command that spans two lines:

```
$ ls Mail info temp \  
(press Enter)
```

```
> diary  
(the > prompt appears)
```

The > character is your secondary prompt (\$ is the nonroot user's default primary prompt), indicating that the current line is the continuation of the previous line. Note that `cs`h (the C shell) gives no secondary prompt, and the break must be at a word boundary, and its primary prompt is %.

The first word of every command is the command name. Some commands have only a command name.

Command flags

A number of flags might follow the command name. Flags modify the operation of a command and are sometimes called *options*.

A flag is set off by spaces or tabs and usually starts with a dash (-). Exceptions are **ps**, **tar**, and **ar**, which do not require a dash in front of some of the flags. For example, in the following command:

```
ls -a -F
```

ls is the command name, and **-a -F** are the flags.

When a command uses flags, they come directly after the command name. Single-character flags in a command can be combined with one dash. For example, the previous command can also be written as follows:

```
ls -aF
```

There are some circumstances when a parameter actually begins with a dash (-). In this case, use the delimiter dash dash (--) before the parameter. The -- tells the command that whatever follows is not a flag but a parameter.

For example, if you want to create a directory named **-tmp** and you type the following command:

```
mkdir -tmp
```

The system displays an error message similar to the following:

```
mkdir: Not a recognized flag: t  
Usage: mkdir [-p] [-m mode] Directory ...
```

The correct way of typing the command is as follows:

```
mkdir -- -tmp
```

Your new directory, **-tmp**, is now created.

Command parameters

After the command name, there might be a number of flags, followed by parameters. Parameters are sometimes called *arguments* or *operands*. Parameters specify information that the command needs in order to run.

If you do not specify a parameter, the command might assume a default value. For example, in the following command:

```
ls -a temp
```

ls is the command name, **-a** is the flag, and *temp* is the parameter. This command displays all (**-a**) the files in the directory **temp**.

In the following example:

```
ls -a
```

the default value is the current directory because no parameter is given.

In the following example:

```
ls temp mail
```

no flags are given, and *temp* and *mail* are parameters. In this case, **temp** and **mail** are two different directory names. The **ls** command displays all but the hidden files in each of these directories.

Whenever a parameter or option-argument is, or contains, a numeric value, the number is interpreted as a decimal integer, unless otherwise specified. Numerals in the range 0 to INT_MAX, as defined in the **/usr/include/sys/limits.h** file, are syntactically recognized as numeric values.

If a command you want to use accepts negative numbers as parameters or option-arguments, you can use numerals in the range INT_MIN to INT_MAX, both as defined in the **/usr/include/sys/limits.h** file. This does not necessarily mean that all numbers within that range are semantically correct. Some commands have a built-in specification permitting a smaller range of numbers, for example, some of the print commands. If an error is generated, the error message lets you know the value is out of the supported range, not that the command is syntactically incorrect.

Usage statements

Usage statements are a way to represent command syntax and consist of symbols such as brackets ([]), braces ({ }), and vertical bars (|).

The following is a sample of a usage statement for the **unget** command:

```
unget [ -rSID ] [ -s ] [ -n ] File ...
```

The following conventions are used in the command usage statements:

- Items that must be entered literally on the command line are in **bold**. These items include the command name, flags, and literal characters.
- Items representing variables that must be replaced by a name are in *italics*. These items include parameters that follow flags and parameters that the command reads, such as *Files* and *Directories*.
- Parameters enclosed in brackets are optional.
- Parameters enclosed in braces are required.
- Parameters not enclosed in either brackets or braces are required.
- A vertical bar signifies that you choose only one parameter. For example, [a | b] indicates that you *can* choose a, b, or nothing. Similarly, { a | b } indicates that you *must* choose either a or b.
- Ellipses (...) signify the parameter can be repeated on the command line.
- The dash (-) represents standard input.

Web-based System Manager startup

The Web-based System Manager is a graphical user interface for managing the system, either from a locally attached display or remotely from another system or personal computer equipped with a Web browser.

You can start the Web-based System Manager in the following ways:

Do this	From here
Enter the wsm command	A command-line terminal in the Common Desktop Environment (CDE)
Enter the wsm command	A command-line terminal in the AIXwindows environment
Go to the System_Admin folder and click the Management Console icon	From the CDE Application Manager
Start an HTML 3.2-compatible Web browser	On a personal computer that is configured as described in the <i>AIX 5L Version 5.3 Web-based System Manager Administration Guide</i>

smit command

The **smit** command is a tool you can use to run other commands.

Command names entered as a parameter to the **smit** command might take you to a submenu or panel for that command. For example, the `smit lsuser` command takes you directly to **List All Users**, which lists the attributes of users on your system.

See the **smit** command for the complete syntax.

Locating another command or program (whereis command)

The **whereis** command locates the source, binary, and manuals sections for specified files. The command attempts to find the desired program from a list of standard locations.

See the following examples:

- To find files in the current directory that have no documentation, type the following:

```
whereis -m -u *
```

- To find all of the files that contain the name Mail, type the following:

```
whereis Mail
```

The system displays information similar to the following:

```
Mail: /usr/bin/Mail /usr/lib/Mail.rc
```

See the **whereis** command in the *AIX 5L Version 5.3 Commands Reference, Volume 6* for the complete syntax.

Displaying information about a command (man command)

The **man** command displays information on commands, subroutines, and files.

The general format for the **man** command is as follows:

```
man CommandName
```

To obtain information about the **pg** command, type the following:

```
man pg
```

The system displays information similar to the following:

```
pg Command
```

```
Purpose
```

```
Formats files to the display.
```

Syntax

```
pg [ - Number ] [ -c ] [ -e ] [ -f ] [ -n ] [ -p String ]  
[ -s ] [ +LineNumber | +/Pattern/ ] [ File ... ]
```

Description

The `pg` command reads a file name from the `File` parameter and writes the file to standard output one screen at a time. If you specify a `-` (dash) as the `File` parameter, or run the `pg` command without options, the `pg` command reads standard input. Each screen is followed by a prompt. If you press the Enter key, another page is displayed. Subcommands used with the `pg` command let you review or search in the file.

See the **man** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Displaying the function of a command (whatis command)

The **whatis** command looks up a given command, system call, library function, or special file name, as specified by the **Command** parameter, from a database you create using the **catman -w** command.

For information about the **catman -w** command, see **catman -w**. The **whatis** command displays the header line from the manual section. You can then issue the **man** command to obtain additional information. For more information about the **man** command, see **man**.

The **whatis** command is equivalent to using the **man -f** command.

To find out what the **ls** command does, type the following:

```
whatis ls
```

The system displays information similar to the following:

```
ls(1) -Displays the contents of a directory.
```

See the **whatis** command in the *AIX 5L Version 5.3 Commands Reference, Volume 6* for the complete syntax.

Listing previously entered commands (history command)

Use the **history** command to list commands that you have previously entered.

The **history** command is a Korn shell built-in command that lists the last 16 commands entered. The Korn shell saves commands that you entered to a command history file, usually named **\$HOME/.sh_history**. Using this command saves time when you need to repeat a previous command.

By default, the Korn shell saves the text of the last 128 commands for nonroot users and 512 commands for the root user. The history file size (specified by the *HISTSIZE* environment variable) is not limited, although a very large history file size can cause the Korn shell to start slowly.

Note: The Bourne shell does not support command history.

For detailed information about shells, see Chapter 34, “Shells,” on page 435.

To list the previous commands you entered, at the prompt, type the following:

```
history
```

The **history** command entered by itself lists the previous 16 commands entered. The system displays information similar to the following:

```
928 ls
929 mail
930 printenv MAILMSG
931 whereis Mail
932 whatis ls
933 cd /usr/include/sys
934 ls
935 man pg
936 cd
937 ls | pg
938 lscons
939 tty
940 ls *.txt
941 printenv MAILMSG
942 pwd
943 history
```

The listing first displays the position of the command in the **\$HOME/.sh_history** file followed by the command.

To list the previous five commands, at the prompt, type the following:

```
history -5
```

A listing similar to the following is displayed:

```
939 tty
940 ls *.txt
941 printenv MAILMSG
942 pwd
943 history
944 history -5
```

The **history** command followed by a number lists all the previous commands entered, starting at that number.

To list the commands since 938, at the prompt, type the following:

```
history 938
```

A listing similar to the following is displayed:

```
938 lscons
939 tty
940 ls *.txt
941 printenv MAILMSG
942 pwd
943 history
944 history -5
945 history 938
```

Repeating commands using the r alias

Use the **r** Korn shell alias to repeat previous commands.

Type **r**, and press Enter, and you can specify the number or the first character or characters of the command.

If you want to list the displays currently available on the system, type **lsdisp** at the prompt. The system returns the information on the screen. If you want the same information returned to you again, at the prompt, type the following:

```
r
```

The system runs the most recently entered command again. In this example, the **lsdisp** command runs.

To repeat the **ls *.txt** command, at the prompt, type the following:

```
r ls
```

The **r** Korn shell alias locates the most recent command that begins with the character or characters specified.

String substitution using the **r** alias

You can use the **r** Korn shell alias to modify a command before it is run.

In this case, a substitution parameter of the form *Old=new* can be used to modify the command before it is run.

The following examples show how to use the **r** alias:

- If command line 940 is **ls *.txt**, and you want to run **ls *.exe**, at the prompt, type the following:

```
r txt=exe 940
```

This runs command 940, substituting **exe** for **txt**.

- If the command on line 940 is the most recent command that starts with a lowercase letter *l*, you can also type the following:

```
r txt=exe l
```

Note: Only the first occurrence of the *Old* string is replaced by the *New* string. Entering the **r** Korn shell alias without a specific command number or character performs the substitution on the immediately previous command entered.

Editing the command history

Use the **fc** Korn shell built-in command to list or edit portions of the command history file.

To select a portion of the file to edit or list, specify the number or the first character or characters of the command. You can specify a single command or range of commands.

If you do not specify an editor program as an argument to the **fc** Korn shell built-in command, the editor specified by the *FCEDIT* variable is used. If the *FCEDIT* variable is not defined, the **/usr/bin/ed** editor is used. The edited command or commands are printed and run when you exit the editor. Use the **printenv** command to display the value of the *FCEDIT* variable.

The following are examples of how to edit the command history:

- If you want to run the command:

```
cd /usr/tmp
```

which is very similar to command line 933, at the prompt, type the following:

```
fc 933
```

At this point, your default editor appears with the command line 933. Change **include/sys** to **tmp**, and when you exit your editor, the edited command is run.

- You can also specify the editor you want to use in the **fc** command. For example, if you want to edit a command using the **/usr/bin/vi** editor, at the prompt, type the following:

```
fc -e vi 933
```

At this point, the **vi** editor appears with the command line 933.

- You can also specify a range of commands to edit. For example, if you want to edit the commands 930 through 940, at the prompt, type the following:

```
fc 930 940
```

At this point, your default editor appears with the command lines 930 through 940. When you exit the editor, all the commands that appear in your editor are run sequentially.

Creating a command alias (alias shell command)

An *alias* lets you create a shortcut name for a command, file name, or any shell text. By using aliases, you save a lot of time when doing tasks you do frequently. You can create a command alias.

Use the **alias** Korn shell built-in command to define a word as an alias for some command. You can use aliases to redefine built-in commands but not to redefine reserved words.

The first character of an alias name can be any printable character except the metacharacters. Any remaining characters must be the same as for a valid file name.

The format for creating an alias is as follows:

```
alias Name=String
```

in which the *Name* parameter specifies the name of the alias, and the *String* parameter specifies a string of characters. If *String* contains blank spaces, enclose it in quotation marks.

The following are examples how to create an alias:

- To create an alias for the command **rm -i** (prompts you before deleting files), at the prompt, type the following:

```
alias rm="/usr/bin/rm -i"
```

In this example, whenever you enter the command **rm**, the actual command performed is **/usr/bin/rm -i**.

- To create an alias named **dir** for the command **ls -aF | pg** (which displays detailed information of all the files in the current directory, including the invisible files; marks executable files with an * and directories with a /; and scrolls per screen), at the prompt, type the following:

```
alias dir="/usr/bin/ls -aF | pg"
```

In this example, whenever you enter the command **dir**, the actual command performed is **/usr/bin/ls -aF | pg**.

- To display all the aliases you have, at the prompt, type the following:

```
alias
```

The system displays information similar to the following:

```
rm="/usr/bin/rm -i"
dir="/usr/bin/ls -aF | pg"
```

International character support in text formatting

You can use text formatting commands to work with text composed of the international extended character set used for European languages.

The international extended character set provides the characters and symbols used in many European languages, as well as an ASCII subset composed of English-language characters, digits, and punctuation.

All characters in the European extended character set have ASCII forms. These forms can be used to represent the extended characters in input, or the characters can be entered directly with a device such as a keyboard that supports the European extended characters.

The following text-formatting commands support all international languages that use single-byte characters. These commands are located in `/usr/bin`. (The commands identified with an asterisk (*) support text processing for multibyte languages. For more information on multibyte languages, see “Multibyte character support in text formatting.”)

<code>addbib*</code>	<code>hyphen</code>	<code>pic*</code>	<code>pstext</code>
<code>checkmm</code>	<code>ibm3812</code>	<code>ps4014</code>	<code>refer*</code>
<code>checknr*</code>	<code>ibm3816</code>	<code>ps630</code>	<code>roffbib*</code>
<code>col*</code>	<code>ibm5587G*</code>	<code>psbanne</code>	<code>soelim*</code>
<code>colcrt</code>	<code>ibm5585H-T*</code>	<code>psdit</code>	<code>sortbib*</code>
<code>deroff*</code>	<code>indxbib*</code>	<code>psplot</code>	<code>tbl*</code>
<code>enscript</code>	<code>lookbib*</code>	<code>psrev</code>	<code>troff*</code>
<code>eqn*</code>	<code>makedev*</code>	<code>psroff</code>	<code>vgrind</code>
<code>grap*</code>	<code>neqn*</code>	<code>psrv</code>	<code>xpreview*</code>
<code>hplj</code>	<code>nroff*</code>		

Text-formatting commands and macro packages not in the preceding list have not been enabled to process international characters.

Text formatting with extended single-byte characters

If your input device supports characters from the European-language extended character set, you can enter them directly.

Otherwise, use the following ASCII escape sequence form to represent these characters:

The form `\[N]`, where *N* is the 2- or 4-digit hexadecimal code for the character.

Note: The NCesc form `\<xx>` is no longer supported.

Text containing extended characters is output according to the formatting conventions of the language in use. Characters that are not defined for the interface to a specific output device produce no output or error indication.

Although the names of the requests, macro packages, and commands are based on English, most of them can accept input (such as file names and parameters) containing characters in the European extended character set.

For the **nroff** and **troff** commands and their preprocessors, the command input must be ASCII, or an unrecoverable syntax error will result. International characters, either single-byte or multibyte, can be entered when enclosed within quotation marks and within other text to be formatted. For example, using macros from the **pic** command:

```
define foobar % SomeText %
```

After the define directive, the first name, foobar, must be ASCII. However, the replacement text, *SomeText*, can contain non-ASCII characters.

Multibyte character support in text formatting

Certain text formatting commands can be used to process text for multibyte languages.

These commands are identified with an asterisk (*) in the list under “International character support in text formatting” on page 360. Text formatting commands not in the list have not been enabled to process international characters.

If supported by your input device, multibyte characters can be entered directly. Otherwise, you can enter any multibyte character in the ASCII form `\[N]`, where *N* is the 2-, 4-, 6-, 7-, or 8-digit hexadecimal encoding for the character.

Although the names of the requests, macros, and commands are based on English, most of them can accept input (such as file names and parameters) containing any type of multibyte character.

If you are already familiar with using text-formatting commands with single-byte text, the following list summarizes characteristics that are noteworthy or unique to the multibyte locales:

- Text is not hyphenated.
- Special format types are required for multibyte numerical output. Japanese format types are available.
- Text is output in horizontal lines, filled from left to right.
- Character spacing is constant, so characters automatically align in columns.
- Characters that are not defined for the interface to a specific output device produce no output or error indication.

Processes

A program or command that is actually running on the computer is referred to as a *process*.

Processes exist in parent-child hierarchies. A process started by a program or command is a *parent process*; a *child process* is the product of the parent process. A parent process can have several child processes, but a child process can have only one parent.

The system assigns a process identification number (PID number) to each process when it starts. If you start the same program several times, it will have a different PID number each time.

When a process is started on a system, the process uses a part of the available system resources. When more than one process is running, a scheduler that is built into the operating system gives each process a share of the computer's time, based on established priorities. These priorities can be changed by using the **nice** or **renice** commands.

Note: To change a process priority to a higher one, you must have root user authority. All users can lower priorities on a process they start by using the **nice** command or on a process they have already started, by using the **renice** command.

The following list describes the types of processes:

Foreground and background processes

Processes that require a user to start them or to interact with them are called *foreground processes*. Processes that are run independently of a user are referred to as *background processes*. Programs and commands run as foreground processes by default. To run a process in the background, place an ampersand (&) at the end of the command name that you use to start the process.

Daemon processes

Daemons are processes that run unattended. They are constantly in the background and are available at all times. Daemons are usually started when the system starts, and they run until the system stops. A daemon process typically performs system services and is available at all times to more than one task or user. Daemon processes are started by the root user or root shell and can be stopped only by the root user. For example, the **qdaemon** process provides access to system resources such as printers. Another common daemon is the **sendmail** daemon.

Zombie processes

A *zombie process* is a dead process that is no longer executing but is still recognized in the process table (in other words, it has a PID number). It has no other system space allocated to it. Zombie processes have been killed or have exited and continue to exist in the process table until the parent process dies or the system is shut down and restarted. Zombie processes display as <defunct> when listed by the **ps** command.

Process startup

You start a foreground process from a display station by either entering a program name or command name at the system prompt.

After a foreground process has started, the process interacts with you at your display station until it is complete. No other interaction (for example, entering another command) can take place at the display station until the process is finished or you halt it.

A single user can run more than one process at a time, up to a default maximum of 40 processes per user.

Starting a process in the foreground

To start a process in the foreground, enter the name of the command with the appropriate parameters and flags:

```
$ CommandName
```

Starting a process in the background

To run a process in the background, type the name of the command with the appropriate parameters and flags, followed by an ampersand (&):

```
$ CommandName&
```

When a process is running in the background, you can perform additional tasks by entering other commands at your display station.

Generally, background processes are most useful for commands that take a long time to run. However, because they increase the total amount of work the processor is doing, background processes can slow down the rest of the system.

Most processes direct their output to standard output, even when they run in the background. Unless redirected, standard output goes to the display device. Because the output from a background process can interfere with your other work on the system, it is usually good practice to redirect the output of a background process to a file or a printer. You can then look at the output whenever you are ready.

Note: Under certain circumstances, a process might generate its output in a different sequence when run in the background than when run in the foreground. Programmers might want to use the **fflush** subroutine to ensure that output occurs in the correct order regardless of whether the process runs in foreground or background.

While a background process is running, you can check its status with the **ps** command.

Command to check the process status (ps command)

Any time the system is running, processes are also running. You can use the **ps** command to find out which processes are running and display information about those processes.

The **ps** command has several flags that enable you to specify which processes to list and what information to display about each process.

To show all processes running on your system, at the prompt, type the following:

```
ps -ef
```

The system displays information similar to the following:

```
USER  PID  PPID  C   STIME  TTY  TIME CMD
root   1    0     0   Jun 28  -   3:23 /etc/init
root  1588 6963  0   Jun 28  -   0:02 /usr/etc/biod 6
```

```

root 2280      1  0   Jun 28      -  1:39 /etc/syncd 60
mary 2413 16998  2 07:57:30     -  0:05 aixterm
mary 11632 16998  0 07:57:31 lft/1 0:01 xbiff
mary 16260 2413   1 07:57:35 pts/1 0:00 /bin/ksh
mary 16469   1  0 07:57:12 lft/1 0:00 ksh /usr/lpp/X11/bin/xinit
mary 19402 16260 20 09:37:21 pts/1 0:00 ps -ef

```

The columns in the previous output are defined as follows:

```

USER      User login name
PID       Process ID
PPID      Parent process ID
C         CPU utilization of process
STIME     Start time of process
TTY       Controlling workstation for the process
TIME     Total execution time for the process
CMD       Command

```

In the previous example, the process ID for the **ps -ef** command is 19402. Its parent process ID is 16260, the **/bin/ksh** command.

If the listing is very long, the top portion scrolls off the screen. To display the listing one page (screen) at a time, pipe the **ps** command to the **pg** command. At the prompt, type the following:

```
ps -ef | pg
```

To display status information of all processes running on your system, at the prompt, type the following:

```
ps gv
```

This form of the command lists a number of statistics for each active process. Output from this command looks similar to the following:

PID	TTY	STAT	TIME	PGIN	SIZE	RSS	LIM	TSIZ	TRS	%CPU	%MEM	COMMAND
0	-	A	0:44	7	8	8	xx	0	0	0.0	0.0	swapper
1	-	A	1:29	518	244	140	xx	21	24	0.1	1.0	/etc/init
771	-	A	1:22	0	16	16	xx	0	0	0.0	0.0	kproc
1028	-	A	0:00	10	16	8	xx	0	0	0.0	0.0	kproc
1503	-	A	0:33	127	16	8	xx	0	0	0.0	0.0	kproc
1679	-	A	1:03	282	192	12 32768	130		0	0.7	0.0	pcidossvr
2089	-	A	0:22	918	72	28	xx	1	4	0.0	0.0	/etc/sync
2784	-	A	0:00	9	16	8	xx	0	0	0.0	0.0	kproc
2816	-	A	5:59	6436	2664	616	8	852	156	0.4	4.0	/usr/lpp/
3115	-	A	0:27	955	264	128	xx	39	36	0.0	1.0	/usr/lib/
3451	-	A	0:00	0	16	8	xx	0	0	0.0	0.0	kproc
3812	-	A	0:00	21	128	12 32768	34		0	0.0	0.0	usr/lib/lpd/
3970	-	A	0:00	0	16	8	xx	0	0	0.0	0.0	kproc
4267	-	A	0:01	169	132	72 32768	16		16	0.0	0.0	/etc/sysl
4514	lft/0	A	0:00	60	200	72	xx	39	60	0.0	0.0	/etc/gett
4776	pts/3	A	0:02	250	108	280	8	303	268	0.0	2.0	-ksh
5050	-	A	0:09	1200	424	132 32768	243		56	0.0	1.0	/usr/sbin
5322	-	A	0:27	1299	156	192	xx	24	24	0.0	1.0	/etc/cron
5590	-	A	0:00	2	100	12 32768	11		0	0.0	0.0	/etc/writ
5749	-	A	0:00	0	208	12	xx	13	0	0.0	0.0	/usr/lpp/
6111	-	T	0:00	66	108	12 32768	47		0	0.0	0.0	/usr/lpp/

See the **ps** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Setting the initial priority of a process (nice command)

You can set the initial priority of a process to a value lower than the base scheduling priority.

To set the initial priority of a process to a value lower than the base scheduling priority, use the **nice** command to start the process.

Note: To run a process at a higher priority than the base scheduling priority, you must have root user authority.

To set the initial priority of a process, type the following:

```
nice -n Number CommandString
```

where *Number* is in the range of 0 to 39, with 39 being the lowest priority. The *nice value* is the decimal value of the system-scheduling priority of a process. The higher the number, the lower the priority. If you use zero, the process will run at its base scheduling priority. *CommandString* is the command and flags and parameters you want to run.

See the **nice** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

You can also use the **smit nice** command to perform this task.

Changing the priority of a running process (renice command)

You can change the scheduling priority of a running process to a value lower or higher than the base scheduling priority by using the **renice** command from the command line. This command changes the nice value of a process.

Note: To run a process at a higher priority or to change the priority for a process that you did not start, you must have root user authority.

To change the priority of a running process, type the following:

```
renice Priority -p ProcessID
```

where *Priority* is a number in the range of -20 to 20. The higher the number, the lower the priority. If you use zero, the process will run at its base scheduling priority. *ProcessID* is the PID for which you want to change the priority.

You can also use the **smit renice** command to perform this task.

Foreground process cancellation

If you start a foreground process and then decide that you do not want it to finish, you can cancel it by pressing INTERRUPT. This is usually Ctrl-C or Ctrl-Backspace.

To find out what your INTERRUPT key is set to, see “List of control key assignments for your terminal (stty command)” on page 344.

Note: INTERRUPT (Ctrl-C) does not cancel background processes. To cancel a background process, you must use the **kill** command.

Most simple commands run so quickly that they finish before you have time to cancel them. The examples in this section, therefore, use a command that takes more than a few seconds to run: **find / -type f**. This command displays the path names for all files on your system. You do not need to study the **find** command in order to complete this section; it is used here simply to demonstrate how to work with processes.

In the following example, the **find** command starts a process. After the process runs for a few seconds, you can cancel it by pressing the INTERRUPT key:

```

$ find / -type f
/usr/sbin/acct/lastlogin
/usr/sbin/acct/prctmp
/usr/sbin/acct/prdaily
/usr/sbin/acct/runacct
/usr/sbin/acct/sdisk
/usr/sbin/acct/shutacct INTERRUPT (Ctrl-C)
$ _

```

The system returns the prompt to the screen. Now you can enter another command.

Keyboard command to stop a foreground process

It is possible for a process to be stopped but not have its process ID (PID) removed from the process table. You can stop a foreground process by pressing Ctrl-Z from the keyboard.

Note: Ctrl-Z works in the Korn shell (**ksh**) and C shell (**csch**), but not in the Bourne shell (**bsh**).

Restarting a stopped process

This procedure describes how to restart a process that has been stopped with a Ctrl-Z.

Note: Ctrl-Z works in the Korn shell (**ksh**) and C shell (**csch**), but not in the Bourne shell (**bsh**). To restart a stopped process, you must either be the user who started the process or have root user authority.

1. To show all the processes running or stopped but not those killed on your system, type the following:

```
ps -ef
```

You might want to pipe this command through a **grep** command to restrict the list to those processes most likely to be the one you want to restart. For example, if you want to restart a **vi** session, you could type the following:

```
ps -ef | grep vi
```

This command would display only those lines from the **ps** command output that contained the word **vi**. The output would look something like this:

```

UID  PID  PPID  C    STIME      TTY  TIME  COMMAND
root 1234 13682 0     00:59:53  -   0:01  vi test
root 14277 13682 1     01:00:34  -   0:00  grep vi

```

2. In the **ps** command output, find the process you want to restart and note its PID number. In the example, the PID is 1234.
3. To send the CONTINUE signal to the stopped process, type the following:

```
kill -19 1234
```

Substitute the PID of your process for the 1234. The -19 indicates the CONTINUE signal. This command restarts the process in the background. If the process can run in the background, you are finished with the procedure. If the process must run in the foreground (as a **vi** session would), you must proceed with the next step.

4. To bring the process in to the foreground, type the following:

```
fg 1234
```

Once again, substitute the PID of your process for the 1234. Your process should now be running in the foreground. (You are now in your **vi** edit session).

Scheduling a process for later operation

You can set up a process as a *batch process* to run in the background at a scheduled time.

The **at** and **smit** commands let you enter the names of commands to be run at a later time and allow you to specify when the commands should be run.

Note: The `/var/adm/cron/at.allow` and `/var/adm/cron/at.deny` files control whether you can use the **at** command. A person with root user authority can create, edit, or delete these files. Entries in these files are user login names with one name to a line. The following is an example of an **at.allow** file:

```
root
nick
dee
sarah
```

If the **at.allow** file exists, only users whose login names are listed in it can use the **at** command. A system administrator can explicitly stop a user from using the **at** command by listing the user's login name, in the **at.deny** file. If only the **at.deny** file exists, any user whose name does not appear in the file can use the **at** command.

You cannot use the **at** command if any one of the following is true:

- The **at.allow** file and the **at.deny** file do not exist (allows root user only).
- The **at.allow** file exists but the user's login name is not listed in it.
- The **at.deny** file exists and the user's login name is listed in it.

If the **at.allow** file does not exist and the **at.deny** file does not exist or is empty, only someone with root user authority can submit a job with the **at** command.

The **at** command syntax allows you to specify a date string, a time and day string, or an increment string for when you want the process to run. It also allows you to specify which shell or queue to use. The following examples show some typical uses of the command.

For example, if your login name is `joyce` and you have a script named `WorkReport` that you want to run at midnight, do the following:

1. Type the time you want the program to start running:

```
at midnight
```

2. Type the names of the programs to run, pressing Enter after each name. After typing the last name, press the end-of-file character (Ctrl-D) to signal the end of the list.

```
WorkReport^D
```

After you press Ctrl-D, the system displays information similar to the following:

```
job joyce.741502800.a at Fri Jul 6 00:00:00 CDT 2002.
```

The program `WorkReport` is given the job number `joyce.741502800.a` and will run at midnight, July 6.

3. To list the programs you have sent to be run later, type the following:

```
at -l
```

The system displays information similar to the following:

```
joyce.741502800.a      Fri Jul 6 00:00:00 CDT 2002
```

See the **at** command for the complete syntax.

Listing all scheduled processes (at or atq command)

Use the **-l** flag with the **at** command or with the **atq** command to list all scheduled processes.

Both commands give the same output; however, the **atq** command can order the processes in the same amount of time that the **at** command is issued and displays only the number of processes in the queue.

You can list all scheduled processes in the following ways:

- With the **at** command from the command line
- With the **atq** command

For user restrictions on using the **at** command, see the **Note** in *Scheduling a Process for Later Operation (at Command)*.

at command

To list the scheduled processes, type the following:

```
at -l
```

This command lists all the scheduled processes in your queue. If you are a root user, this command lists all the scheduled processes for all users. For complete details of the syntax, see the **at** command.

atq command

See the following examples on how to use the **atq** command:

- To list all scheduled processes in the queue, type the following:

```
atq
```

- If you are a root user, you can list the scheduled processes in a particular user's queue by typing:

```
atq UserName
```

- To list the number of scheduled processes in the queue, type the following:

```
atq -n
```

Removing a process from the schedule

You can remove a scheduled process with the **at** command using the **-r** flag.

For user restrictions on using the **at** command, see the note in *Scheduling a Process for Later Operation (at Command)*.

See the following example on how to use the **at** or **atq** command:

1. To remove a scheduled process, you must know its process number. You can obtain the process number by using the **at -l** command or the **atq** command. See "Listing all scheduled processes (at or atq command)" on page 367 for details.
2. When you know the number of the process you want to remove, type the following:

```
at -r ProcessNumber
```

You can also use the `smit rmat` command to perform this task.

Removing a background process (kill command)

If **INTERRUPT** does not halt your foreground process or if you decide, after starting a background process, that you do not want the process to finish, you can cancel the process with the **kill** command.

Before you can cancel a process using the **kill** command, you must know its PID number. The general format for the **kill** command is as follows:

```
kill ProcessID
```

Note:

- To remove a process, you must have root user authority or be the user who started the process. The default signal to a process from the **kill** command is -15 (SIGTERM).
- To remove a zombie process, you must remove its parent process.

1. Use the **ps** command to determine the process ID of the process you want to remove. You might want to pipe this command through a **grep** command to list only the process you want. For example, if you want the process ID of a vi session, you could type the following:

```
ps -l | grep vi
```

2. In the following example, you issue the **find** command to run in the background. You then decide to cancel the process. Issue the **ps** command to list the PID numbers.

```
$ find / -type f > dir.paths &
[1] 21593
$ ps
  PID  TTY  TIME  COMMAND
  1627 pts3  0:00  ps
  5461 pts3  0:00  ksh
 17565 pts3  0:00  -ksh
 21593 pts3  0:00  find / -type f
$ kill 21593
$ ps
  PID  TTY  TIME  COMMAND
  1627 pts3  0:00  ps
  5461 pts3  0:00  ksh
 17565 pts3  0:00  -ksh
[1] + Terminated 21593      find / -type f > dir.paths &
```

The command **kill 21593** ends the background **find** process, and the second **ps** command returns no status information about PID 21593. The system does not display the termination message until you enter your next command, unless that command is **cd**.

The **kill** command lets you cancel background processes. You might want to do this if you realize that you have mistakenly put a process in the background or that a process is taking too long to run.

See the **kill** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

The **kill** command can also be used in **smit** by typing:

```
smit kill
```

Command summary for commands and processes

The following are commands for commands and processes.

Table 10. Command summary for commands

alias	Shell command that prints a list of aliases to standard output
history	Shell command that displays the history event list
man	Displays information about commands, subroutines, and files online
wsm	Performs system management from a web browser
whatis	Describes the function a command performs
whereis	Locates the source, binary, or manual for installed programs

Table 11. Command summary for processes

at	Runs commands at a later time, lists all scheduled processes, or removes a process from the schedule
atq	Displays the queue of jobs waiting to be run
kill	Sends a signal to running processes
nice	Runs a command at a lower or higher priority
ps	Shows current status of processes
renice	Alters priority of running processes

Chapter 29. Input and output redirection

The AIX operating system allows you to manipulate the input and output (I/O) of data to and from your system by using specific I/O commands and symbols.

You can control input by specifying the location from which to gather data. For example, you can specify to read input while data is entered on the keyboard (standard input) or to read input from a file. You can control output by specifying where to display or store data. You can specify to write output data to the screen (standard output) or to write it to a file.

Because AIX is a multitasking operating system, it is designed to handle processes in combination with each other.

Standard input, standard output, and standard error files

When a command begins running, it usually expects that the following files are already open: standard input, standard output, and standard error (sometimes called *error output* or *diagnostic output*).

A number, called a *file descriptor*, is associated with each of these files, as follows:

File descriptor 0	Standard input
File descriptor 1	Standard output
File descriptor 2	Standard error (diagnostic) output

A child process normally inherits these files from its parent. All three files are initially assigned to the workstation (0 to the keyboard, 1 and 2 to the display). The shell permits them to be redirected elsewhere before control is passed to a command.

When you enter a command, if no file name is given, your keyboard is the *standard input*, sometimes denoted as *stdin*. When a command finishes, the results are displayed on your screen.

Your screen is the *standard output*, sometimes denoted as *stdout*. By default, commands take input from the standard input and send the results to standard output.

Error messages are directed to standard error, sometimes denoted as *stderr*. By default, this is your screen.

These default actions of input and output can be varied. You can use a file as input and write results of a command to a file. This is called *input/output redirection*.

The output from a command, which normally goes to the display device, can be redirected to a file instead. This is known as *output redirection*. This is useful when you have a lot of output that is difficult to read on the screen or when you want to put files together to create a larger file.

Though not used as much as output redirection, the input for a command, which normally comes from the keyboard, can also be redirected from a file. This is known as *input redirection*. Redirection of input lets you prepare a file in advance and then have the command read the file.

Standard output redirection

When the notation *>filename* is added to the end of a command, the output of the command is written to the specified file name. The *>* symbol is known as the *output redirection operator*.

Any command that outputs its results to the screen can have its output redirected to a file.

Redirecting output to a file

The output of a process can be redirected to a file by typing the command followed by the output redirection operator and file name.

For example, to redirect the results of the **who** command to a file named **users**, type the following:

```
who > users
```

Note: If the **users** file already exists, it is deleted and replaced, unless the **noclobber** option of the **set** built-in **ksh** (Korn shell) or **csk** (C shell) command is specified.

To see the contents of the **users** file, type the following:

```
cat users
```

A list similar to the following is displayed:

```
denise 1ft/0 May 13 08:05
marta  pts/1 May 13 08:10
endrica pts/2 May 13 09:33
```

Redirecting output to append to a file

When the notation `>> filename` is added to the end of a command, the output of the command is appended to the specified file name, rather than writing over any existing data. The `>>` symbol is known as the *append redirection operator*.

For example, to append **file2** to **file1**, type the following:

```
cat file2 >> file1
```

Note: If the **file1** file does not exist, it is created, unless the **noclobber** option of the **set** built-in **ksh** (Korn shell) or **csk** (C shell) command is specified.

Creating a text file with redirection from the keyboard

Used alone, the **cat** command uses whatever you type at the keyboard as input. You can redirect this input to a file.

Press Ctrl-D on a new line to signal the end of the text.

At the system prompt, type the following:

```
cat > filename
This is a test.
^D
```

Text file concatenation

You can combine multiple files into one file. Combining various files into one file is known as *concatenation*.

The following example creates **file4**, which consists of **file1**, **file2**, and **file3**, appended in the order below.

See the following examples:

- At the system prompt, type the following:

```
cat file1 file2 file3 > file4
```
- The following example shows a common error when concatenating files:

```
cat file1 file2 file3 > file1
```

Attention: In this example, you might expect the **cat** command to append the contents of **file1**, **file2**, and **file3** into **file1**. The **cat** command creates the output file first, so it actually erases the contents of **file1** and then appends **file2** and **file3** to it.

Standard input redirection

When the notation `< filename` is added to the end of a command, the input of the command is read from the specified file name. The `<` symbol is known as the *input redirection operator*.

Note: Only commands that normally take their input from the keyboard can have their input redirected.

For example, to send the file **letter1** as a message to user **denise** with the **mail** command, type the following:

```
mail denise < letter1
```

Discarding output with the /dev/null file

The **/dev/null** file is a special file. This file has a unique property: it is always empty. Any data sent to **/dev/null** is discarded. This is a useful feature when you run a program or command that generates output that you want to ignore.

For example, you have a program named **myprog** that accepts input from the screen and generates messages while it is running that you would rather not see on your screen. To read input from the file **mymyscript** and discard the standard output messages, type the following:

```
myprog < myscript >/dev/null
```

In this example, **myprog** uses the file **mymyscript** as input, and all standard output is discarded.

Standard error and other output redirection

In addition to the standard input and standard output, commands often produce other types of output, such as error or status messages known as diagnostic output. Like standard output, standard error output is written to the screen unless it is redirected.

To redirect standard error or other output, use a file descriptor. A *file descriptor* is a number associated with each of the I/O files that a command ordinarily uses. File descriptors can also be specified to redirect standard input and standard output, but are already the default values. The following numbers are associated with standard input, output, and error:

0	Standard input (keyboard)
1	Standard output (display)
2	Standard error (display)

To redirect standard error output, type the file descriptor number 2 in front of the output or append redirection symbols (`>` or `>>`) and a file name after the symbol. For example, the following command takes the standard error output from the **cc** command where it is used to compile the **testfile.c** file and appends it to the end of the **ERRORS** file:

```
cc testfile.c 2 >> ERRORS
```

Other types of output can also be redirected using the file descriptors from 0 through 9. For example, if the **cmd** command writes output to file descriptor 9, you can redirect that output to the **savedata** file with the following command:

```
cmd 9> savedata
```

If a command writes to more than one output, you can independently redirect each one. Suppose that a command directs its standard output to file descriptor 1, directs its standard error output to file descriptor 2, and builds a data file on file descriptor 9. The following command line redirects each of these outputs to a different file:

```
command > standard 2> error 9> data
```

Redirecting output to inline input (here) documents

You can redirect output to inline input (here) documents.

If a command is in the following form:

```
command << eofstring
```

and *eofstring* is any string that does not contain pattern-matching characters, then the shell takes the subsequent lines as the standard input of *command* until the shell reads a line consisting of only *eofstring* (possibly preceded by one or more tab characters). The lines between the first *eofstring* and the second are frequently referred to as an *inline input document*, or a *here document*. If a hyphen (-) immediately follows the << redirection characters, the shell strips leading tab characters from each line of the **here** document before it passes the line to *command*.

The shell creates a temporary file containing the **here** document and performs variable and command substitution on the contents before passing the file to the command. It performs pattern matching on file names that are part of command lines in command substitutions. To prohibit all substitutions, quote any character of the *eofstring*:

```
command << \eofstring
```

The **here** document is especially useful for a small amount of input data that is more conveniently placed in the shell procedure rather than kept in a separate file (such as editor scripts). For example, you could type the following:

```
cat <<- xyz
  This message will be shown on the
  display with leading tabs removed.
xyz
```

Redirecting output using pipes and filters

You can connect two or more commands so that the standard output of one command is used as the standard input of another command. A set of commands connected this way is known as a *pipeline*. The connection that joins the commands is known as a *pipe*. Pipes are useful because they let you tie many single-purpose commands into one powerful command.

You can direct the output from one command to become the input for another command using a pipeline. The commands are connected by a pipe (|) symbol.

When a command takes its input from another command, modifies it, and sends its results to standard output, it is known as a *filter*. Filters can be used alone, but they are especially useful in pipelines. The most common filters are as follows:

- sort
- more
- pg

See the following examples:

- The **ls** command writes the contents of the current directory to the screen in one scrolling data stream. When more than one screen of information is presented, some data is lost from view. To control the

output so the contents display screen by screen, you can use a pipeline to direct the output of the **ls** command to the **pg** command, which controls the format of output to the screen. For example, type the following:

```
ls | pg
```

In this example, the output of the **ls** command becomes the input for the **pg** command. Press Enter to continue to the next screen.

Pipelines operate in one direction only (left to right). Each command in a pipeline runs as a separate process, and all processes can run at the same time. A process pauses when it has no input to read or when the pipe to the next process is full.

- Another example of using pipes is with the **grep** command. The **grep** command searches a file for lines that contain strings of a certain pattern. To display all your files created or modified in July, type the following:

```
ls -l | grep Jul
```

In this example, the output of the **ls** command becomes the input for the **grep** command.

Displaying program output and copying to a file (tee command)

The **tee** command, used with a pipe, reads standard input, then writes the output of a program to standard output and simultaneously copies it into the specified file or files. Use the **tee** command to view your output immediately and at the same time, store it for future use.

For example, type the following:

```
ps -ef | tee program.ps
```

This displays the standard output of the **ps -ef** command on the display device, and at the same time, saves a copy of it in the **program.ps** file. If the **program.ps** file already exists, it is deleted and replaced unless the **noclobber** option of the **set** built-in command is specified.

For example, to view and save the output from a command to an existing file:

```
ls -l | tee -a program.ls
```

This displays the standard output of **ls -l** at the display device and at the same time appends a copy of it to the end of the **program.ls** file.

The system displays information similar to the following, and the **program.ls** file contains the same information:

```
-rw-rw-rw- 1 jones  staff  2301  Sep 19   08:53 161414
-rw-rw-rw- 1 jones  staff  6317  Aug 31   13:17 def.rpt
-rw-rw-rw- 1 jones  staff  5550  Sep 10   14:13 try.doc
```

See the **tee** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Clearing your screen (clear command)

Use the **clear** command to empty the screen of messages and keyboard input.

At the prompt, type the following:

```
clear
```

The system clears the screen and displays the prompt.

Sending a message to standard output

Use the **echo** command to display messages on the screen.

For example, to write a message to standard output, at the prompt, type the following:

```
echo Please insert diskette . . .
```

The following message is displayed:

```
Please insert diskette . . .
```

For example, to use the **echo** command with pattern-matching characters, at the prompt, type the following:

```
echo The back-up files are: *.bak
```

The system displays the message The back-up files are: followed by the file names in the current directory ending with **.bak**.

Appending a single line of text to a file (echo command)

Use the **echo** command, used with the append redirection operator, to add a single line of text to a file.

For example, at the prompt, type the following:

```
echo Remember to back up mail files by end of week.>>notes
```

This adds the message Remember to back up mail files by end of week. to the end of the file **notes**.

Copying your screen to a file (capture and script commands)

Use the **capture** command, which emulates a VT100 terminal, to copy everything printed on your terminal to a file that you specify. Use the **script** command to copy everything printed on your terminal to a file that you specify, without emulating a VT100 terminal.

Both commands are useful for printing records of terminal dialogs.

For example, to capture the screen of a terminal while emulating a VT100, at the prompt, type the following:

```
capture screen.01
```

The system displays information similar to the following:

```
Capture command is started. The file is screen.01.  
Use ^P to dump screen to file screen.01.  
You are now emulating a vt100 terminal.  
Press Any Key to continue.
```

After entering data and dumping the screen contents, stop the **capture** command by pressing Ctrl-D or typing **exit** and pressing Enter. The system displays information similar to the following:

```
Capture command is complete. The file is screen.01.  
You are NO LONGER emulating a vt100 terminal.
```

Use the **cat** command to display the contents of your file.

For example, to capture the screen of a terminal, at the prompt, type the following:

```
script
```

The system displays information similar to the following:

Script command is started. The file is typescript.

Everything displayed on the screen is now copied to the **typescript** file.

To stop the **script** command, press Ctrl-D or type `exit` and press Enter. The system displays information similar to the following:

```
Script command is complete. The file is typescript.
```

Use the **cat** command to display the contents of your file.

See the **capture** and **script** commands in the *AIX 5L Version 5.3 Commands Reference* for the complete syntax.

Command to display text in large letters on your screen (**banner** command)

The **banner** command displays ASCII characters to your screen in large letters.

Each line in the output can be up to 10 digits (or uppercase or lowercase characters) in length.

For example, at the prompt, type the following:

```
banner GOODBYE!
```

The system displays GOODBYE! in large letters on your screen.

Command summary for input and output redirection

The following are commands for input and output redirection.

>	“Standard output redirection” on page 371
<	“Standard input redirection” on page 373
> >	“Redirecting output to append to a file” on page 372
	“Redirecting output using pipes and filters” on page 374
banner	Writes ASCII character strings in large letters to standard output
capture	Allows terminal screens to be dumped to a file
clear	Clears the terminal screen
echo	Writes character strings to standard output
script	Allows terminal input and output to be copied to a file
tee	Displays the standard output of a program and copies it into a file

Chapter 30. Files

Files are used for all input and output (I/O) of information in the operating system, to standardize access to both software and hardware.

Input occurs when the contents of a file is modified or written to. *Output* occurs when the contents of one file is read or transferred to another file. For example, to create a printed copy of a file, the system reads the information from the text file and writes that information to the file representing the printer.

Types of files

The types of files recognized by the system are either **regular**, **directory**, or **special**. However, the operating system uses many variations of these basic types.

The following basic types of files exist:

regular	Stores data (text, binary, and executable)
directory	Contains information used to access other files
special	Defines a FIFO (first-in, first-out) pipe file or a physical device

All file types recognized by the system fall into one of these categories. However, the operating system uses many variations of these basic types.

Regular files

Regular files are the most common files and are used to contain data. Regular files are in the form of text files or binary files:

Text files

Text files are regular files that contain information stored in ASCII format text and are readable by the user. You can display and print these files. The lines of a text file must not contain NUL characters, and none can exceed {LINE_MAX} bytes in length, including the newline character.

The term *text file* does not prevent the inclusion of control or other nonprintable characters (other than NUL). Therefore, standard utilities that list text files as inputs or outputs are either able to process the special characters or they explicitly describe their limitations within their individual sections.

Binary files

Binary files are regular files that contain information readable by the computer. Binary files might be executable files that instruct the system to accomplish a job. Commands and programs are stored in executable, binary files. Special compiling programs translate ASCII text into binary code.

Text and binary files differ only in that text files have lines of less than {LINE_MAX} bytes, with no NUL characters, each terminated by a newline character.

Directory files

Directory files contain information that the system needs to access all types of files, but directory files do not contain the actual file data. As a result, directories occupy less space than a regular file and give the file system structure flexibility and depth. Each directory entry represents either a file or a subdirectory. Each entry contains the name of the file and the file's index node reference number (*i-node number*). The i-node number points to the unique index node assigned to the file. The i-node number describes the location of the data associated with the file. Directories are created and controlled by a separate set of commands.

Special files

Special files define devices for the system or are temporary files created by processes. The basic types of special files are FIFO (first-in, first-out), block, and character. FIFO files are also called *pipes*. Pipes are created by one process to temporarily allow communication with another process. These files cease to exist when the first process finishes. Block and character files define devices.

Every file has a set of permissions (called *access modes*) that determine who can read, modify, or execute the file.

To learn more about file access modes, see “File and directory access modes” on page 413.

File naming conventions

The name of each file must be unique within the directory where it is stored. This ensures that the file also has a unique path name in the file system.

File naming guidelines are:

- A file name can be up to 255 characters long and can contain letters, numbers, and underscores.
- The operating system is case-sensitive, which means it distinguishes between uppercase and lowercase letters in file names. Therefore, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory.
- File names should be as descriptive and meaningful as possible.
- Directories follow the same naming conventions as files.
- Certain characters have special meaning to the operating system. Avoid using these characters when you are naming files. These characters include the following:

```
/ \ " ' * ; - ? [ ] ( ) ~ ! $ { } < > # @ & |
```

- A file name is hidden from a normal directory listing if it begins with a dot (.). When the **ls** command is entered with the **-a** flag, the hidden files are listed along with regular files and directories.

File path names

The path name for each file and directory in the file system consists of the names of every directory that precedes it in the tree structure.

Because all paths in a file system originate from the **/**(root) directory, each file in the file system has a unique relationship to the root directory, known as the *absolute path name*. Absolute path names begin with the slash (**/**) symbol. For example, the absolute path name of file **h** could be **/B/C/h**. Notice that two files named **h** can exist in the system. Because the absolute paths to the two files are different, **/B/h** and **/B/C/h**, each file named **h** has a unique name within the system. Every component of a path name is a directory except the final component. The final component of a path name can be a file name.

Note: Path names cannot exceed 1023 characters in length.

Pattern matching with wildcards and metacharacters

Wildcard characters provide a convenient way to specify multiple file names or directory names.

The wildcard characters are asterisk (*****) and question mark (**?**). The metacharacters are open and close square brackets (**[]**), hyphen (**-**), and exclamation mark (**!**).

Pattern matching using the ***** wildcard character

Use the asterisk (*****) to match any sequence or string of characters.

The (*****) indicates any characters, including no characters.

See the following examples:

- If you have the following files in your directory:

```
1test 2test afile1 afile2 bfile1 file file1 file10 file2 file3
```

and you want to refer to only the files that begin with `file`, use:

```
file*
```

The files selected would be: **file**, **file1**, **file10**, **file2**, and **file3**.

- To refer to only the files that contain the word `file`, use:

```
*file*
```

The files selected would be: **afile1**, **afile2**, **bfile1**, **file**, **file1**, **file10**, **file2**, and **file3**.

Pattern matching using the ? wildcard character

Use the `?` to match any one character.

The `?` indicates any single character. See the following examples:

- To refer to only the files that start with **file** and end with a single character, use:

```
file?
```

The files selected would be: **file1**, **file2**, **file3**.

- To refer to only the files that start with **file** and end with any two characters, use:

```
file??
```

The file selected would be: **file10**.

Pattern matching using [] shell metacharacters

Metacharacters offer another type of wildcard notation by enclosing the desired characters within `[]`. It is like using the `?`, but it allows you to choose specific characters to be matched.

The `[]` also allow you to specify a range of values using the hyphen (`-`). To specify all the letters in the alphabet, use `[:alpha:]`. To specify all the lowercase letters in the alphabet, use `[:lower:]`.

See the following examples:

- To refer to only the files that end in 1 or 2, use:

```
*file[12]
```

The files selected would be: **afile1**, **afile2**, **file1**, and **file2**.

- To refer to only the files that start with any number, use:

```
[0123456789]* or [0-9]*
```

The files selected would be: **1test** and **2test**.

- To refer to only the files that do not begin with an `a`, use:

```
[!a]*
```

The files selected would be: **1test**, **2test**, **bfile1**, **file**, **file1**, **file10**, **file2**, and **file3**.

Pattern matching versus regular expressions

Regular expressions allow you to select specific strings from a set of character strings. The use of regular expressions is generally associated with text processing.

Regular expressions can represent a wide variety of possible strings. While many regular expressions can be interpreted differently depending on the current locale, internationalization features provide for contextual invariance across locales.

See the examples in the following comparison:

Pattern Matching	Regular Expression
*	.*
?	.
[!a]	[^a]
[abc]	[abc]
[:alpha:]	[:alpha:]

See the **awk** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

File handling procedures

There are many ways to work with the files on your system. Usually you create a text file with a text editor.

The common editors in the UNIX environment are **vi** and **ed**. Because several text editors are available, you can choose an editor you feel comfortable with.

You can also create files by using input and output redirection, as described in Chapter 29, “Input and output redirection,” on page 371. You can send the output of a command to a new file or append it to an existing file.

After creating and modifying files, you might have to copy or move files from one directory to another, rename files to distinguish different versions of a file, or give different names to the same file. You might also need to create directories when working on different projects.

Also, you might need to delete certain files. Your directory can quickly get cluttered with files that contain old or useless information. To release storage space on your system, ensure that you delete files that are no longer needed.

Deleting files (rm command)

Use the **rm** command to remove files you no longer need.

The **rm** command removes the entries for a specified file, group of files, or certain select files from a list within a directory. User confirmation, read permission, and write permission are not required before a file is removed when you use the **rm** command. However, you must have write permission for the directory containing the file.

The following are examples of how to use the **rm** command:

- To delete the file named **myfile**, type the following:

```
rm myfile
```
- To delete all the files in the **mydir** directory, one by one, type the following:

```
rm -i mydir/*
```

After each file name displays, type **y** and press **Enter** to delete the file. Or to keep the file, just press **Enter**.

See the **rm** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Moving and renaming files (mv command)

Use the **mv** command to move files and directories from one directory to another or to rename a file or directory. If you move a file or directory to a new directory without specifying a new name, it retains its original name.

Attention: The **mv** command can overwrite many existing files unless you specify the **-i** flag. The **-i** flag prompts you to confirm before it overwrites a file. The **-f** flag does not prompt you. If both the **-f** and **-i** flags are specified in combination, the last flag specified takes precedence.

Moving files with mv command

The following are examples of how to use the **mv** command:

- To move a file to another directory and give it a new name, type the following:

```
mv intro manual/chap1
```

This moves the **intro** file to the **manual/chap1** directory. The name **intro** is removed from the current directory, and the same file appears as **chap1** in the **manual** directory.

- To move a file to another directory, keeping the same name, type the following:

```
mv chap3 manual
```

This moves **chap3** to **manual/chap3**.

Renaming files with mv command

Use the **mv** command to change the name of a file without moving it to another directory.

To rename a file, type the following:

```
mv appendix apndx.a
```

This renames the **appendix** file to **apndx.a**. If a file named **apndx.a** already exists, its old contents are replaced with those of the **appendix** file.

See the **mv** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Copying files (cp command)

Use the **cp** command to create a copy of the contents of the file or directory specified by the *SourceFile* or *SourceDirectory* parameters into the file or directory specified by the *TargetFile* or *TargetDirectory* parameters.

If the file specified as the *TargetFile* exists, the copy writes over the original contents of the file without warning. If you are copying more than one *SourceFile*, the target must be a directory.

If a file with the same name exists at the new destination, the copied file overwrites the file at the new destination. Therefore, it is a good practice to assign a *new* name for the copy of the file to ensure that a file of the same name does not exist in the destination directory.

To place a copy of the *SourceFile* into a directory, specify a path to an existing directory for the *TargetDirectory* parameter. Files maintain their respective names when copied to a directory unless you specify a new file name at the end of the path. The **cp** command also copies entire directories into other directories if you specify the **-r** or **-R** flags.

You can also copy special-device files using the **-R** flag. Specifying **-R** causes the special files to be re-created under the new path name. Specifying the **-r** flag causes the **cp** command to attempt to copy the special files to regular files.

The following are examples of how to use the **cp** command:

- To make a copy of a file in the current directory, type the following:

```
cp prog.c prog.bak
```

This copies **prog.c** to **prog.bak**. If the **prog.bak** file does not already exist, then the **cp** command creates it. If it does exist, then the **cp** command replaces it with a copy of the **prog.c** file.

- To copy a file in your current directory into another directory, type the following:

```
cp jones /home/nick/clients
```

This copies the **jones** file to **/home/nick/clients/jones**.

- To copy all the files in a directory to a new directory, type the following:

```
cp /home/janet/clients/* /home/nick/customers
```

This copies only the files in the **clients** directory to the **customers** directory.

- To copy a specific set of files to another directory, type the following:

```
cp jones lewis smith /home/nick/clients
```

This copies the **jones**, **lewis**, and **smith** files in your current working directory to the **/home/nick/clients** directory.

- To use pattern-matching characters to copy files, type the following:

```
cp programs/*.c .
```

This copies the files in the **programs** directory that end with **.c** to the current directory, indicated by the single dot (**.**). You must type a space between the **c** and the final dot.

See the **cp** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Finding files (find command)

Use the **find** command to recursively search the directory tree for each specified *Path*, seeking files that match a Boolean expression written using the terms given in the following text.

The output from the **find** command depends on the terms specified by the *Expression* parameter.

The following are examples of how to use the **find** command:

- To list all files in the file system with the name **.profile**, type the following:

```
find / -name .profile
```

This searches the entire file system and writes the complete path names of all files named **.profile**. The slash (**/**) tells the **find** command to search the **/(root)** directory and all of its subdirectories.

To save time, limit the search by specifying the directories where you think the files might be.

- To list files having a specific permission code of 0600 in the current directory tree, type the following:

```
find . -perm 0600
```

This lists the names of the files that have *only* owner-read and owner-write permission. The dot (**.**) tells the **find** command to search the current directory and its subdirectories. For an explanation of permission codes, see the **chmod** command.

- To search several directories for files with certain permission codes, type the following:

```
find manual clients proposals -perm -0600
```

This lists the names of the files that have owner-read and owner-write permission and possibly other permissions. The **manual**, **clients**, and **proposals** directories and their subdirectories are searched. In the previous example, **-perm 0600** selects only files with permission codes that match 0600 exactly. In this example, **-perm -0600** selects files with permission codes that allow the accesses indicated by 0600 and other accesses above the 0600 level. This also matches the permission codes 0622 and 2744.

- To list all files in the current directory that have been changed during the current 24-hour period, type the following:

```
find . -ctime 1
```

- To search for regular files with multiple links, type the following:

```
find . -type f -links +1
```

This lists the names of the ordinary files (-type f) that have more than one link (-links +1).

Note: Every directory has at least two links: the entry in its parent directory and its own .(dot) entry. For more information on multiple file links, see the **ln** command.

- To search for all files that are exactly 414 bytes in length, type the following:

```
find . -size 414c
```

See the **find** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Displaying the file type (file command)

Use the **file** command to read the files specified by the *File* or **-fFileList** parameter, perform a series of tests on each one, and attempt to classify the files by type. The command then writes the file types to standard output.

If a file appears to be ASCII, the **file** command examines the first 512 bytes and determines its language. If a file does not appear to be ASCII, the **file** command further attempts to determine whether it is a binary data file or a text file that contains extended characters.

If the *File* parameter specifies an executable or object module file and the version number is greater than 0, the **file** command displays the version stamp.

The **file** command uses the **/etc/magic** file to identify files that have a magic number; that is, any file containing a numeric or string constant that indicates the type.

The following are examples of how to use the **file** command:

- To display the type of information the file named **myfile** contains, type the following:

```
file myfile
```

This displays the file type of **myfile** (such as directory, data, ASCII text, C program source, or archive).

- To display the type of each file named in the **filenames.lst** file, which contains a list of file names, type the following:

```
file -f filenames.lst
```

This displays the type of each file named in the **filenames.lst** file. Each file name must display on a separate line.

- To create the **filenames.lst** file that contains all the file names in the current directory, type the following:

```
ls > filenames.lst
```

Edit the **filenames.lst** file as desired.

See the **file** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Commands for displaying file contents (pg, more, page, and cat commands)

The **pg**, **more**, and **page** commands allow you to view the contents of a file and control the speed at which your files are displayed.

You can also use the **cat** command to display the contents of one or more files on your screen. Combining the **cat** command with the **pg** command allows you to read the contents of a file one full screen at a time.

You can also display the contents of files by using input and output redirection. See Chapter 29, “Input and output redirection,” on page 371 for more details on input and output redirection.

Using the **pg** command

Use the **pg** command to read the files named in the **File** parameter and writes them to standard output one screen at a time.

If you specify hyphen (-) as the **File** parameter or run the **pg** command without options, the **pg** command reads standard input. Each screen is followed by a prompt. If you press the Enter key, another screen displays. Subcommands used with the **pg** command let you review content that has already passed.

For example, to look at the contents of the file **myfile** one page at a time, type the following:

```
pg myfile
```

See the **pg** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Using the **more** or **page** commands

Use the **more** or **page** command to display continuous text one screen at a time.

It pauses after each screen and prints the *filename* and percent completed (for example, **myfile (7%)**) at the bottom of the screen. If you then press the Enter key, the **more** command displays an additional line. If you press the Spacebar, the **more** command displays another screen of text.

Note: On some terminal models, the **more** command clears the screen, instead of scrolling, before displaying the next screen of text.

For example, to view a file named **myfile**, type the following:

```
more myfile
```

Press the Spacebar to view the next screen.

See the **more** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

cat command

Use the **cat** command to read each *File* parameter in sequence and writes it to standard output.

See the following examples:

- To display the contents of the file **notes**, type the following:

```
cat notes
```

If the file is more than 24 lines long, some of it scrolls off the screen. To list a file one page at a time, use the **pg** command.

- To display the contents of the files **notes**, **notes2**, and **notes3**, type the following:

```
cat notes notes2 notes3
```

See the **cat** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Finding text strings within files (**grep** command)

Use the **grep** command to search the specified file for the pattern specified by the *Pattern* parameter and writes each matching line to standard output.

The following are examples of how to use the **grep** command:

- To search in a file named **pgm.s** for a pattern that contains some of the pattern-matching characters `*`, `^`, `?`, `[`, `]`, `\(`, `\)`, `\{`, and `\}`, in this case, lines starting with any lowercase or uppercase letter, type the following:

```
grep "^[a-zA-Z]" pgm.s
```

This displays all lines in the **pgm.s** file that begin with a letter.

- To display all lines in a file named **sort.c** that do not match a particular pattern, type the following:

```
grep -v bubble sort.c
```

This displays all lines that do not contain the word `bubble` in the **sort.c** file.

- To display lines in the output of the **ls** command that match the string `staff`, type the following:

```
ls -l | grep staff
```

See the **grep** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Sorting text files (sort command)

Use the **sort** command to alphabetize or sequence lines in the files specified by the **File** parameters and write the result to standard output.

If the **File** parameter specifies more than one file, the **sort** command concatenates the files and alphabetizes them as one file.

Note: The **sort** command is case-sensitive and orders uppercase letters before lowercase (this behavior is dependent on the locale).

In the following examples, the contents of the file named **names** are:

```
marta
denise
joyce
endrica
melanie
```

and the contents of the file named **states** are:

```
texas
colorado
ohio
```

- To display the sorted contents of the file named **names**, type the following:

```
sort names
```

The system displays information similar to the following:

```
denise
endrica
joyce
marta
melanie
```

- To display the sorted contents of the **names** and **states** files, type the following:

```
sort names states
```

The system displays information similar to the following:

```
colorado
denise
endrica
joyce
```

```
marta
melanie
ohio
texas
```

- To replace the original contents of the file named **names** with its sorted contents, type the following:

```
sort -o names names
```

This replaces the contents of the **names** file with the same data but in sorted order.

See the **sort** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Comparing files (diff command)

Use the **diff** command to compare text files. It can compare single files or the contents of directories.

When the **diff** command is run on regular files, and when it compares text files in different directories, the **diff** command tells which lines must be changed in the files so that they match.

The following are examples of how to use the **diff** command:

- To compare two files, type the following:

```
diff chap1.bak chap1
```

This displays the differences between the **chap1.bak** and **chap1** files.

- To compare two files while ignoring differences in the amount of white space, type the following:

```
diff -w prog.c.bak prog.c
```

If the two files differ only in the number of spaces and tabs between words, the **diff -w** command considers the files to be the same.

See the **diff** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Counting words, lines, and bytes in files (wc command)

Use the **wc** command to count the number of lines, words, and bytes in the files specified by the *File* parameter.

If a file is not specified for the *File* parameter, standard input is used. The command writes the results to standard output and keeps a total count for all named files. If flags are specified, the ordering of the flags determines the ordering of the output. A *word* is defined as a string of characters delimited by spaces, tabs, or newline characters.

When files are specified on the command line, their names are printed along with the counts.

See the following examples:

- To display the line, word, and byte counts of the file named **chap1**, type the following:

```
wc chap1
```

This displays the number of lines, words, and bytes in the **chap1** file.

- To display only byte and word counts, type the following:

```
wc -cw chap*
```

This displays the number of bytes and words in each file where the name starts with **chap**, and displays the totals.

See the **wc** command in the *AIX 5L Version 5.3 Commands Reference, Volume 6* for the complete syntax.

Displaying the first lines of files (head command)

Use the **head** command to write to standard output the first few lines of each of the specified files or of the standard input.

If no flag is specified with the **head** command, the first 10 lines are displayed by default.

For example, to display the first five lines of the **Test** file, type the following:

```
head -5 Test
```

See the **head** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Displaying the last lines of files (tail command)

Use the **tail** command to write the file specified by the *File* parameter to standard output beginning at a specified point.

See the following examples:

- To display the last 10 lines of the **notes** file, type the following:

```
tail notes
```
- To specify the number of lines to start reading from the end of the **notes** file, type the following:

```
tail -20 notes
```
- To display the **notes** file one page at a time, beginning with the 200th byte, type the following:

```
tail -c +200 notes | pg
```
- To follow the growth of the file named **accounts**, type the following:

```
tail -f accounts
```

This displays the last 10 lines of the **accounts** file. The **tail** command continues to display lines as they are added to the **accounts** file. The display continues until you press the (Ctrl-C) key sequence to stop the display.

See the **tail** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Cutting sections of text files (cut command)

Use the **cut** command to write selected bytes, characters, or fields from each line of a file to standard output.

See the following examples:

- To display several fields of each line of a file, type the following:

```
cut -f1,5 -d: /etc/passwd
```

This displays the login name and full user name fields of the system password file. These are the first and fifth fields (-f1,5) separated by colons (-d:).

- If the **/etc/passwd** file looks like this:

```
su:*:0:0:User with special privileges:/:usr/bin/sh
daemon:*:1:1::/etc:
bin:*:2:2::usr/bin:
sys:*:3:3::usr/src:
adm:*:4:4:System Administrator:/var/adm:usr/bin/sh
pierre:*:200:200:Pierre Harper:/home/pierre:usr/bin/sh
joan:*:202:200:Joan Brown:/home/joan:usr/bin/sh
```

the **cut** command produces:

```
su:User with special privileges
daemon:
bin:
sys:
adm:System Administrator
pierre:Pierre Harper
joan:Joan Brown
```

See the **cut** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Pasting sections of text files (paste command)

Use the **paste** command to merge the lines of up to 12 files into one file.

See the following examples:

- If you have a file named **names** that contains the following text:

```
rachel
jerry
mark
linda
scott
```

and another file named **places** that contains the following text:

```
New York
Austin
Chicago
Boca Raton
Seattle
```

and another file named **dates** that contains the following text:

```
February 5
March 13
June 21
July 16
November 4
```

To paste the text of the files **names**, **places**, and **dates** together, type the following:

```
paste names places dates > npd
```

This creates a file named **npd** that contains the data from the **names** file in one column, the **places** file in another, and the **dates** file in a third. The **npd** file now contains the following:

```
rachel      New York      February 5
jerry       Austin        March 13
mark        Chicago       June 21
linda       Boca Raton    July 16
scott       Seattle       November 4
```

A tab character separates the name, place, and date on each line. These columns do not align, because the tab stops are set at every eighth column.

- To separate the columns with a character other than a tab, type the following:

```
paste -d"!@" names places dates > npd
```

This alternates ! and @ as the column separators. If the **names**, **places**, and **dates** files are the same as in example 1, then the **npd** file contains the following:

```
rachel!New York@February 5
jerry!Austin@March 13
mark!Chicago@June 21
linda!Boca Raton@July 16
scott!Seattle@November 4
```

- To list the current directory in four columns, type the following:

```
ls | paste - - - -
```

Each hyphen (-) tells the **paste** command to create a column containing data read from the standard input. The first line is put in the first column, the second line in the second column, and so on.

See the **paste** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Numbering lines in text files (nl command)

Use the **nl** command to read the specified file (standard input by default), numbers the lines in the input, and writes the numbered lines to standard output.

See the following examples:

- To number only the lines that are not blank, type the following:

```
nl chap1
```

This displays a numbered listing of **chap1**, numbering only the lines that are not blank in the body sections.

- To number all lines, type the following:

```
nl -ba chap1
```

This numbers all the lines in the file named **chap1**, including blank lines.

See the **nl** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

Removing columns in text files (colrm command)

Use the **colrm** command to remove specified columns from a file. Input is taken from standard input. Output is sent to standard output.

If the command is called with one parameter, the columns of each line from the specified column to the last column are removed. If the command is called with two parameters, the columns from the first specified column to the second specified column are removed.

Note: Column numbering starts with column 1.

See the following examples:

- To remove columns from the **text.fil** file, type the following:

```
colrm 6 < text.fil
```

If **text.fil** contains:

```
123456789
```

then the **colrm** command displays:

```
12345
```

See the **colrm** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

File and directory links

Links are connections between a file name and an index node reference number (i-node number), the internal representation of a file. Because directory entries contain file names paired with i-node numbers, every directory entry is a link. The i-node number actually identifies the file, not the file name. By using links, any i-node number or file can be known by many different names.

For example, i-node number 798 contains a memo regarding June sales in the Omaha office. Presently, the directory entry for this memo is as follows:

i-node Number	File Name
798	memo

Because this information relates to information stored in the **sales** and **omaha** directories, linking is used to share the information where it is needed. Using the **ln** command, links are created to these directories. Now the file has three file names as follows:

i-node Number	File Name
798	memo
798	sales/june
798	omaha/junesales

When you use the **pg** or **cat** command to view the contents of any of the three file names, the same information is displayed. If you edit the contents of the i-node number from any of the three file names, the contents of the data displayed by all of the file names will reflect any changes.

Types of links

There are two types of links: hard and symbolic.

Links are created with the **ln** command and are of the following types:

hard link	Allows access to the data of a file from a new file name. Hard links ensure the existence of a file. When the last hard link is removed, the i-node number and its data are deleted. Hard links can be created only between files that are in the same file system.
symbolic link	Allows access to data in other file systems from a new file name. The symbolic link is a special type of file that contains a path name. When a process encounters a symbolic link, the process may search that path. Symbolic links do not protect a file from deletion from the file system.

Note: The user who creates a file retains ownership of that file no matter how many links are created. Only the owner of the file or the root user can set the access mode for that file. However, changes can be made to the file from a linked file name with the proper access mode.

A file or directory exists as long as there is one hard link to the i-node number for that file. In the long listing displayed by the **ls -l** command, the number of hard links to each file and subdirectory is given. All hard links are treated equally by the operating system, regardless of which link was created first.

Linking files (ln command)

Linking files with the **ln** command is a convenient way to work with the same data as if it were in more than one place.

Links are created by giving alternate names to the original file. The use of links allows a large file, such as a database or mailing list, to be shared by several users without making copies of that file. Not only do links save disk space, but changes made to one file are automatically reflected in all the linked files.

The **ln** command links the file designated in the **SourceFile** parameter to the file designated by the **TargetFile** parameter or to the same file name in another directory specified by the **TargetDirectory** parameter. By default, the **ln** command creates hard links. To use the **ln** command to create symbolic links, add the **-s** flag.

Note: You cannot link files across file systems without using the **-s** flag.

If you are linking a file to a new name, you can list only one file. If you are linking to a directory, you can list more than one file.

The **TargetFile** parameter is optional. If you do not designate a target file, the **ln** command creates a file in your current directory. The new file inherits the name of the file designated in the **SourceFile** parameter.

See the following examples:

- To create a link to a file named **chap1**, type the following:

```
ln -f chap1 intro
```

This links **chap1** to the new name, **intro**. When the **-f** flag is used, the file name **intro** is created if it does not already exist. If **intro** does exist, the file is replaced by a link to **chap1**. Both the **chap1** and **intro** file names refer to the same file.

- To link a file named **index** to the same name in another directory named **manual**, type the following:

```
ln index manual
```

This links **index** to the new name, **manual/index**.

- To link several files to names in another directory, type the following:

```
ln chap2 jim/chap3 /home/manual
```

This links **chap2** to the new name **/home/manual/chap2** and **jim/chap3** to **/home/manual/chap3**.

- To use the **ln** command with pattern-matching characters, type the following:

```
ln manual/* .
```

Note: You must type a space between the asterisk and the period.

This links all files in the **manual** directory into the current directory, dot (**.**), giving them the same names they have in the **manual** directory.

- To create a symbolic link, type the following:

```
ln -s /tmp/toc toc
```

This creates the symbolic link, **toc**, in the current directory. The **toc** file points to the **/tmp/toc** file. If the **/tmp/toc** file exists, the **cat toc** command lists its contents.

- To achieve identical results without designating the **TargetFile** parameter, type the following:

```
ln -s /tmp/toc
```

See the **ln** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Command for removing linked files

The **rm** command removes the link from the file name that you indicate.

When one of several hard-linked file names is deleted, the file is not completely deleted because it remains under the other name. When the last link to an i-node number is removed, the data is removed as well. The i-node number is then available for reuse by the system.

See the **rm** command in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

DOS files

The AIX operating system allows you to work with DOS files on your system.

Copy to a diskette the DOS files you want to work with. Your system can read these files into a base operating system directory in the correct format and back onto the diskette in DOS format.

Note: The wildcard characters * and ? (asterisk and question mark) do not work correctly with the commands discussed in this section (although they do with the base operating system shell). If you do not specify a file name extension, the file name is matched as if you had specified a blank extension.

Copying DOS files to base operating system files

Use the **dosread** command to copy the specified DOS file to the specified base operating system file.

Note: DOS file-naming conventions are used with one exception. Because the backslash (\) character can have special meaning to the base operating system, use a slash (/) character as the delimiter to specify subdirectory names in a DOS path name.

See the following examples:

- To copy a text file named **chap1.doc** from a DOS diskette to the base operating file system, type the following:

```
dosread -a chap1.doc chap1
```

This copies the DOS text file **\CHAP1.DOC** on the **/dev/fd0** default device to the base operating system file **chap1** in the current directory.

- To copy a binary file from a DOS diskette to the base operating file system, type the following:

```
dosread -D/dev/fd0 /survey/test.dta /home/fran/testdata
```

This copies the **\SURVEY\TEST.DTA** DOS data file on **/dev/fd0** to the base operating system file **/home/fran/testdata**.

See the **dosread** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Copying base operating system files to DOS files

Use the **doswrite** command to copy the specified base operating system file to the specified DOS file.

Note: DOS file-naming conventions are used with one exception. Because the backslash (\) character can have special meaning to the base operating system, use a slash (/) character as the delimiter to specify subdirectory names in a DOS path name.

See the following examples:

- To copy a text file named **chap1** from the base operating file system to a DOS diskette, type the following:

```
doswrite -a chap1 chap1.doc
```

This copies the base operating system file **chap1** in the current directory to the DOS text file **\CHAP1.DOC** on **/dev/fd0**.

- To copy a binary file named **/survey/test.dta** from the base operating file system to a DOS diskette, type the following:

```
doswrite -D/dev/fd0 /home/fran/testdata /survey/test.dta
```


This copies the base operating system data file **/home/fran/testdata** to the DOS file **\SURVEY\TEST.DTA** on **/dev/fd0**.

See the **doswrite** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Deleting DOS files

Use the **dosdel** command to delete the specified DOS file.

Note: DOS file-naming conventions are used with one exception. Because the backslash (\) character can have special meaning to the base operating system, use a slash (/) character as the delimiter to specify subdirectory names in a DOS path name.

The **dosdel** command converts lowercase characters in the file or directory name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial slash (/).

For example, to delete a DOS file named **file.ext** on the default device (**/dev/fd0**), type the following:

```
dosdel file.ext
```

See the **dosdel** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Displaying contents of a DOS directory

Use the **dosdir** command to display information about the specified DOS files or directories.

Note: DOS file-naming conventions are used with one exception. Because the backslash (\) character can have special meaning to the base operating system, use a slash (/) character as the delimiter to specify subdirectory names in a DOS path name.

The **dosdir** command converts lowercase characters in the file or directory name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

For example, to read a directory of the DOS files on **/dev/fd0**, type the following:

```
dosdir
```

The command returns the names of the files and disk-space information, similar to the following.

```
PG3-25.TXT  
PG4-25.TXT  
PG5-25.TXT  
PG6-25.TXT  
Free space: 312320 bytes
```

See the **dosdir** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2* for the complete syntax.

Command summary for files

The following are commands for files, file handling procedures, and DOS files. There is also a list of commands for linking files and directories.

Table 12. Commands for files

*	Wildcard, matches any characters
?	Wildcard, matches any single character

Table 12. Commands for files (continued)

[] Metacharacters, matches enclosed characters.

Table 13. Commands for file handling procedures

cat	Concatenates or displays files
cmp	Compares two files
colrm	Extracts columns from a file
cp	Copies files
cut	Writes out selected bytes, characters, or fields from each line of a file
diff	Compares text files
file	Determines the file type
find	Finds files with a matching expression
grep	Searches a file for a pattern
head	Displays the first few lines or bytes of a file or files
more	Displays continuous text one screen at a time on a display screen
mv	Moves files
nl	Numbers lines in a file
pg	Formats files to the display
rm	Removes (unlinks) files or directories
paste	Merges the lines of several files or subsequent lines in one file
sort	Sorts files, merges files that are already sorted, and checks files to determine if they have been sorted
tail	Writes a file to standard output, beginning at a specified point
wc	Counts the number of lines, words, and bytes in a file

Table 14. Command for linking files and directories

ln Links files and directories

Table 15. Commands for DOS files

dosdel	Deletes DOS files
dosdir	Lists the directory for DOS files
dosread	Copies DOS files to Base Operating System files
doswrite	Copies Base Operating System files to DOS files

Chapter 31. Printers, print jobs, and queues

Depending on the printer, you can control the appearance and characteristics of the final output.

The printers need not be located in the same area as the system unit and the system console. A printer can be attached directly to a local system, or a print job can be sent over a network to a remote system.

To handle print jobs with maximum efficiency, the system places each job into a queue to await printer availability. The system can save output from one or more files in the queue. As the printer produces the output from one file, the system processes the next job in the queue. This process continues until each job in the queue has been printed.

For detailed information about printers, print jobs, and queues, see the *Printers and printing*.

Printing terminology

The following list defines terms commonly used with printing.

local printer

When you attach a printer to a node or host, the printer is referred to as a *local printer*.

print job

A *print job* is a unit of work to be run on a printer. A print job can consist of printing one or more files, depending on how the print job is requested. The system assigns a unique job number to each job it prints.

print spooler

The *spooler* used for printing provides a generic spooling function that can be used for queuing various types of jobs, including print jobs queued to a printer. The spooler does not normally know what type of job it is queuing. When the system administrator defines a spooler queue, the purpose of the queue is defined by the spooler backend program that is specified for the queue. For example, if the spooler backend program is the **piobe** command (the printer I/O backend), the queue is a print queue.

Likewise, if the spooler backend program is a compiler, the queue is for compile jobs. When the spooler's **qdaemon** process selects a job from a spooler queue, it runs the job by invoking the backend program specified by the system administrator when the queue was defined.

The main spooler command is the **enq** command. Although you can invoke this command directly to queue a print job, three front-end commands are defined for submitting a print job: the **lp**, **lpr**, and **qprt** commands. A print request issued by one of these commands is first passed to the **enq** program, which then places the information about the file in the queue for the **qdaemon** to process.

printer backend

The *printer backend* is a collection of programs called by the spooler's **qdaemon** process to manage a print job that is queued for printing. The printer backend performs the following functions:

- Receives from the **qdaemon** process a list of one or more files to be printed
- Uses printer and formatting attribute values from the database, overridden by flags entered on the command line
- Initializes the printer before printing a file
- Runs filters as necessary to convert the print data stream to a format supported by the printer
- Provides filters for simple formatting of ASCII documents
- Provides support for printing national language characters

- Passes the filtered print data stream to the printer device driver
- Generates header and trailer pages
- Generates multiple copies
- Reports paper out, intervention required, and printer error conditions
- Reports problems detected by the filters
- Cleans up after a print job is canceled
- Provides a print environment that a system administrator can customize to address specific printing needs

printer/plotter device

A special file in the **/dev** directory for the device. This file can be used by redirection (for example, `cat FileName > /dev/lp0`). Settings for the device driver can be displayed and changed using Web-based System Manager (type `wsm`, and then select **Devices**) or use the **lsdev** and **chdev** commands. Before printer commands can access a printer device, a print queue must be created for the device, or the printer must be configured in the printer backend in **/etc/qconfig**.

qdaemon

The **qdaemon** is a process that runs in the background controlling the queues. It is usually started by the **startsrc** command when the system is turned on. **startsrc** is a process to the **srcmstr** daemon that is started from **/etc/inittab**.

The **qdaemon** keeps track of the print requests in the **/var/spool/lpd/qdir** directory and ensures that the jobs are sent to the proper printer at the proper time. It also keeps track of the status of the printers and stores printer usage data for system accounting purposes (for example, **lpstat** and **enq -A** commands). This information is held in the **/var/spool/lpd/stat** directory.

If the **qdaemon** is stopped, it will be restarted by the **srcmstr** process.

Note: Do not stop the **srcmstr** process; it controls other daemons running on your system.

queue The *queue* is where you direct a print job. It is a stanza in the **/etc/qconfig** file whose name is the name of the queue. It points to the associated queue device. The following is a sample listing:

```
Msal:
    device = lp0
```

Usually, queues are created through Web-based System Manager.

queue device

The *queue device* is the stanza in the **/etc/qconfig** file that usually follows the local queue stanza. It specifies the **/dev** file (printer device) that should be printed to and the backend that should be used. Following is a sample listing:

```
lp0:
    file = /dev/lp0
    header = never
    trailer = never
    access = both
    backend = /usr/lpd/piobe
```

In the previous example, `lp0` is the device name, and the rest of the lines define how the device is used.

Adding a printer through Web-based System Manager (type `wsm`, and then select **Devices**) creates a standard queue device entry to an existing queue.

Note:

- There can be more than one queue device associated with a single queue.
- There will not be a file entry in the **/etc/qconfig** file when you are using a remote printer. The queue directs the file to the server.

real printer

A *real printer* is the printer hardware attached to a serial or parallel port at a unique hardware device address. The printer device driver in the kernel communicates with the printer hardware and provides an interface between the printer hardware and a virtual printer.

remote printer

A *remote print system* allows nodes that are not directly linked to a printer to have printer access.

To use remote printing facilities, the individual nodes must be connected to a network using the Transmission Control Protocol/Internet Protocol (TCP/IP) and must support the required TCP/IP applications.

virtual printer

A *virtual printer*, usually called a *virtual printer definition*, is a file containing a set of attribute values that describe a particular data stream for a particular printer. Before a print job can be placed in a queue, a virtual printer definition must exist for both the print queue and the queue device. See the **mkvirprt** command for more information.

Command for starting a print job (qprt command)

You can use the **qprt** command to start a print job.

You can use the **qprt** or **smit** commands to request a print job and specify the following:

- Name of the file to print
- Print queue name
- Number of copies to print
- Whether to make a copy of the file on the remote host
- Whether to erase the file after printing
- Whether to send notification of the job status
- Whether to send notification of the job status by the system mail
- Burst status
- User name for "Delivery To" label
- Console acknowledgment message for remote print
- File acknowledgment message for remote print
- Priority level

The following are prerequisites for starting a print job:

- For local print jobs, the printer must be physically attached to your system or, in the case of a network printer, attached and configured on the network.
- For remote print jobs, your system must be configured to communicate with the remote print server.
- Before you can print a file, you must have *read* access to it. To remove a file after it has printed, you must have *write* access to the directory that contains the file.

Use the **qprt** command to create and queue a print job to print the file you specify. If you specify more than one file, all the files together make up one print job. These files are printed in the order specified on the command line.

The basic format of the **qprt** command is:

```
qprt -PQueueName FileName
```

Some useful **qprt** command flags are:

-b <i>Number</i>	Specifies the bottom margin. The bottom margin is the number of blank lines to be left at the bottom of each page.
-B <i>Value</i>	Specifies whether burst pages (continuous-form pages separated at perforations) should be printed. The <i>Value</i> variable consists of a two-character string. The first character applies to header pages. The second character applies to trailer pages. Each of the two characters can be one of the following: a Always prints the (header or trailer) page for each file in each print job. n Never prints the (header or trailer) page. g Prints the (header or trailer) page once for each print job (group of files). For example, the -B ga flag specifies that a header page be printed at the beginning of each print job and that a trailer page be printed after each file in each print job. Note: In a remote print environment, the default is determined by the remote queue on the server.
-e <i>Option</i>	Specifies whether emphasized print is wanted. + Indicates emphasized print is wanted. ! Indicates emphasized print is not wanted.
-E <i>Option</i>	Specifies whether double-high print is wanted. + Indicates double-high print is wanted. ! Indicates double-high print is not wanted.
-f <i>FilterType</i>	A one-character identifier that specifies a filter through which your print file or files are to be passed before being sent to the printer. The available filter identifiers are p , which invokes the pr filter, and n , which processes output from the troff command.
-i <i>Number</i>	Causes each line to be indented the specified number of spaces. The <i>Number</i> variable must be included in the page width specified by the -w flag.
-K <i>Option</i>	Specifies whether condensed print is wanted. + Indicates condensed print is wanted. ! Indicates condensed print is not wanted.
-l <i>Number</i>	Sets the page length to the specified number of lines. If the <i>Number</i> variable is 0, the page length is ignored, and the output is considered to be one continuous page. The page length includes the top and bottom margins and indicates the printable length of the paper.
-L <i>Option</i>	Specifies whether lines wider than the page width should be wrapped to the next line or truncated at the right margin. + Indicates that long lines should wrap to the next line. ! Indicates that long lines should not wrap but instead should be truncated at the right margin.
-N <i>Number</i>	Specifies the number of copies to be printed. If this flag is not specified, one copy is printed.
-p <i>Number</i>	Sets the pitch to <i>Number</i> characters per inch. Typical values for <i>Number</i> are 10 and 12. The actual pitch of the characters printed is also affected by the values for the -K (condensed) flag and the -W (double-wide) flag.
-P <i>Queue[:QueueDevice]</i>	Specifies the print queue name and the optional queue device name. If this flag is not specified, the default printer is assumed.

-Q <i>Value</i>	Specifies paper size for the print job. The <i>Value</i> for paper size is printer-dependent. Typical values are 1 for letter-size paper, 2 for legal, and so on. Consult your printer manual for the values assigned to specific paper sizes.
-t <i>Number</i>	Specifies the top margin. The top margin is the number of blank lines to be left at the top of each page.
-w <i>Number</i>	Sets the page width to the number of characters specified by the <i>Number</i> variable. The page width must include the number of indentation spaces specified with the -i flag.
-W <i>Option</i>	Specifies whether double-wide print is wanted. + Indicates double-wide print is wanted. ! Indicates double-wide print is not wanted.
-z <i>Value</i>	Rotates page printer output the number of quarter-turns clockwise as specified by the <i>Value</i> variable. The length (-l) and width (-w) values are automatically adjusted accordingly. 0 Portrait 1 Landscape right 2 Portrait upside-down 3 Landscape left.
-# <i>Value</i>	Specifies a special function. j Displays the job number for the specified print job h Queues the print job, but puts it in the HELD state until it is released again. v Validates the specified printer backend flag values. This validation is useful in checking for illegal flag values at the time of submitting a print job. If the validation is not specified, an incorrect flag value will stop the print job later when the job is actually being processed.

The following list contains examples of how to use the **qprt** command flags:

- To request that the file **myfile** be printed on the first available printer configured for the default print queue using default values, type the following:

```
qprt myfile
```
- To request that the file **myfile** be printed on a specific queue using specific flag values and to validate the flag values at the time of print job submission, type the following:

```
qprt -f p -e + -Pfastest -# v somefile
```

This passes the **myfile** file through the **pr** filter command (the **-f p** flag) and prints it using emphasized mode (the **-e +** flag) on the first available printer configured for the queue named **fastest** (the **-Pfastest** flag).

- To print **myfile** on legal-size paper, type the following:

```
qprt -Q2 myfile
```
- To print three copies of each of the files **new.index.c**, **print.index.c**, and **more.c** at the print queue **Msp1**, type the following:

```
qprt -PMsp1 -N 3 new.index.c print.index.c more.c
```
- To print three copies of the concatenation of three files **new.index.c**, **print.index.c**, and **more.c**, type the following:

```
cat new.index.c print.index.c more.c | qprt -PMsp1 -N 3
```

Note: The base operating system also supports the BSD UNIX print command (**lpr**) and the System V UNIX print command (**lp**). See the **lpr** and **lp** commands in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the exact syntax.

See the **qprt** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

To start a print job using SMIT, type the following:

```
smit qprt
```

Canceling a print job (qcan command)

You can cancel any job in the print queue. When you cancel a print job, you are prompted to provide the name of the print queue where the job resides and the job number to be canceled.

The following are prerequisites:

- For local print jobs, the printer must be physically attached to your system or, in the case of a network printer, attached and configured on the network.
- For remote print jobs, your system must be configured to communicate with the remote print server.

This procedure applies to both local and remote print jobs.

Canceling a print job using Web-based System Manager

To cancel a print job using the Web-based System Manager fast path, type the following:

```
wsm printers
```

In the **Printer Queues** window, select the print job, and then use the menus to cancel it from a print queue.

Canceling a print job using the qcan command

Use the **qcan** command to cancel either a particular job number in a local or remote print queue, or all jobs in a local print queue. To determine the job number, enter the **qchk** command.

The basic format of the **qcan** command is:

```
qcan -PQueueName -x JobNumber
```

See the **qcan** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

The following list contains examples of how to use the **qcan** command:

- To cancel job number 123 on whichever printer the job is on, type the following:

```
qcan -x 123
```
- To cancel all jobs queued on printer lp0, type the following:

```
qcan -X -Plp0
```

Note: The base operating system also supports the BSD UNIX cancel print command (**lprm**) and the System V UNIX cancel print command (**cancel**). See the **lprm** and **cancel** commands in the *AIX 5L Version 5.3 Commands Reference* for more information and the complete syntax.

Canceling a print job using the SMIT command

To cancel a print job using SMIT, type the following:

```
smit qcan
```

Prioritizing a print job (qpri command)

You can change the priority of a print job.

You can only assign job priority on local queues. Higher values indicate a higher priority for the print job. The default priority is 15. The maximum priority is 20 for most users, and 30 for users with root user privilege and members of the printq group (group 9).

Note: You cannot assign priority to a remote print job.

In order to prioritize a print job, the printer must be physically attached to your system.

Prioritizing a print job using Web-based System Manager

To change the priority of a queued print job using Web-based System Manager, type `wsm`, and then select **Printers**.

In the **Printer Queues** window, select the print job, and then use the menus to set the priority for that job in a local print queue.

Prioritizing a print job using the qpri command

Use the `qpri` command to reassign the priority of a print job that you submitted. If you have root user authority or belong to the printq group, you can assign priority to any job while it is in the print queue.

The basic format of the `qpri` command is:

```
qpri -# JobNumber -a PriorityLevel
```

See the `qpri` command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

The following list contains examples of how to use the `qpri` command:

- To change job number 123 to priority number 18, type the following:

```
qpri -# 123 -a 18
```
- To prioritize a local print job as it is submitted, type the following:

```
qpri -PQueueName -R PriorityLevel FileName
```

Prioritizing a print job using the smit command

To change the priority of a print job using SMIT, type the following:

```
smit qpri
```

Moving a print job to another print queue (qmov command)

You can move a print job to another print queue.

Note: You cannot move a remote print job to another print queue.

In order to move print job to another print queue, the printer must be physically attached to your system.

Moving a print job to another print queue using Web-based System Manager

To move a print job to another queue using Web-based System Manager, type `wsm`, and then select **Printers**.

In the **Printer Queues** window, select the print job, and then use the menus to move it from one print queue to another.

Moving a print job to another print queue using the qmov command

Use the **qmov** command to move a print job to another print queue. You can either move a particular print job, or you can move all the print jobs on a specified print queue or all the print jobs sent by a specified user. To determine the print job number, enter the **qchk** command. For more information, see “Checking print job status (qchk command)” on page 405.

The basic format of the **qmov** command is:

```
qmov -mNewQueue {[ -#JobNumber ] [ -PQueue ] [ -uUser ]}
```

See the **qmov** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

The following list contains examples of how to use the **qmov** command:

- To move job number 280 to print queue hp2, type the following:

```
qmov -mhp2 -#280
```
- To move all print jobs on print queue hp4D to print queue hp2, type the following:

```
qmov -mhp2 -Php4D
```

Moving a print job to another print queue using the smit command

To move a print job using SMIT, type the following:

```
smit qmov
```

Holding and releasing a print job (qhld command)

You can hold a print job after it has been sent to the print queue. You can later release the print job for printing with the same commands.

Note: You cannot hold and release remote print jobs.

In order to hold or release a print job, the printer must be physically attached to your system.

Holding or releasing print jobs using Web-based System Manager

To hold or release a print job using Web-based System Manager, type `wsm`, and then select **Printers**.

In the **Printer Queues** window, select the print job, and then use the menus to put it on hold or to release a held job for printing.

Holding or releasing print jobs using the qhld command

Use the **qhld** command to put a print job on hold after you have sent it. You can either put a particular print job on hold, or you can hold all the print jobs on a specified print queue. To determine the print job number, enter the **qchk** command. For more information, see “Checking print job status (qchk command)” on page 405.

The basic format of the **qhld** command is:

```
qhld [ -r ] {[ -#JobNumber ] [ -PQueue ] [ -uUser ]}
```

See the **qhld** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

The following list contains examples of how to use the **qhld** command:

- To hold job number 452 on whichever print queue the job is on, type the following:

```
qhld -#452
```
- To hold all jobs queued on print queue hp2, type the following:

```
qhld -Php2
```
- To release job number 452 on whichever print queue the job is on, type the following:

```
qhld -#452 -r
```

- To release all jobs queued on print queue hp2, type the following:

```
qhld -Php2 -r
```

Holding or releasing print jobs using the smit command

To hold or release a print job using SMIT, type the following:

```
smit qhld
```

Checking print job status (qchk command)

You can display the current status information for specified job numbers, queues, printers, or users.

The following are prerequisites:

- For local print jobs, the printer must be physically attached to your system or, in the case of a network printer, attached and configured on the network.
- For remote print jobs, your system must be configured to communicate with the remote print server.

Checking print job status using Web-based System Manager

To check the status of a print job using Web-based System Manager, type `wsm`, and then select **Printers**.

In the **Printer Queues** window, select the print job, and then use the menus to check its status.

Checking print job status using the qchk command

Use the **qchk** command to display the current status information regarding specified print jobs, print queues, or users.

The basic format of the **qchk** command is:

```
qchk -P QueueName -# JobNumber -u OwnerName
```

See the **qchk** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

The following list contains examples of how to use the **qchk** command:

- To display the default print queue, type the following:

```
qchk -q
```
- To display the long status of all queues until empty, while updating the screen every 5 seconds, type the following:

```
qchk -A -L -w 5
```
- To display the status for print queue lp0, type the following:

```
qchk -P lp0
```
- To display the status for job number 123, type the following:

```
qchk -# 123
```
- To check the status of all jobs in all queues, type the following:

```
qchk -A
```

Note: The base operating system also supports the BSD UNIX check print queue command (**lpq**) and the System V UNIX check print queue command (**lpstat**). See the **lpq** and **lpstat** commands in the *AIX 5L Version 5.3 Commands Reference, Volume 3* for the complete syntax.

Checking print job status using the smit command

To check a print job's status using SMIT, type the following:

Print queue status conditions

The following are some of the status conditions that a print queue can have.

DEV_BUSY	<p>Indicates that:</p> <ul style="list-style-type: none"> • More than one queue is defined to a printer device (1p0) and another queue is currently using the printer device. • qdaemon attempted to use the printer port device (1p0), but another application is currently using that printer device <p>To recover from a DEV_BUSY, wait until the queue or application has released the printer device or cancel the job or process that is using the printer port.</p>
DEV_WAIT	<p>Indicates that the queue is waiting on the printer because the printer is offline, out of paper, jammed, or the cable is loose, bad, or wired incorrectly.</p> <p>To recover from a DEV_WAIT, correct the problem that caused it to wait. It may be easier for diagnostic testing to use the enq command to move all queued jobs from the DEV_WAIT queue to another queue that is either printing or is DOWN. After the problem is corrected, you can move any unprinted job back to the original queue.</p>
DOWN	<p>A queue will usually go into a DOWN state after it has been in the DEV_WAIT state. This situation occurs when the printer device driver cannot tell if the printer is there due to absence of correct signaling. However, some printers may not have the capability to signal the queuing system that it is offline, and instead signals that it is off. If the printer device signals or appears to be off, the queue will go into the DOWN state.</p> <p>To recover from a DOWN state, correct the problem that has brought the queue down and have the system administrator bring the queue back up. The queue <i>must</i> be manually brought up before it can be used again.</p>
HELD	Specifies that a print job is held. The print job will not be processed by the spooler until it is released.
QUEUED	Specifies that a print file is queued and is waiting in line to be printed.
READY	Specifies that everything involved with the queue is ready to queue and print a job.
RUNNING	Specifies that a print file is printing.

Command for formatting files for printing (pr command)

The **pr** command performs simple formatting of the files you send to be printed. You pipe the output of the **pr** command to the **qprt** command to format your text.

Some useful **pr** command flags are:

-d	Double-spaces the output.
-h "String"	Displays the specified string, enclosed in quotes (" "), instead of the file name as the page header. The flag and string should be separated by a space.
-l Lines	Overrides the 66-line default and resets the page length to the number of lines specified by the <i>Lines</i> variable. If the <i>Lines</i> value is smaller than the sum of both the header and trailer depths (in lines), the header and trailer are suppressed (as if the -t flag were in effect).
-m	Merges files. Standard output is formatted so the pr command writes one line from each file specified by a <i>File</i> variable, side by side into text columns of equal fixed widths, based on the number of column positions. This flag should not be used with the <i>-Column</i> flag.

-n [<i>Width</i>][<i>Character</i>]	Provides line numbering based on the number of digits specified by the <i>Width</i> variable. The default is 5 digits. If the <i>Character</i> (any nondigit character) variable is specified, it is appended to the line number to separate it from what follows on the line. The default character separator is the ASCII Tab character.
-o <i>Offset</i>	Indents each line by the number of character positions specified by the <i>Offset</i> variable. The total number of character positions per line is the sum of the width and offset. The default value of <i>Offset</i> is 0.
-s <i>Character</i>	Separates columns by the single character specified by the <i>Character</i> variable instead of by the appropriate number of spaces. The default value for <i>Character</i> is an ASCII Tab character.
-t	Does not display the five-line identifying header and the five-line footer. Stops after the last line of each file without spacing to the end of the page.
-w <i>Width</i>	Sets the number of column positions per line to the value specified by the <i>Width</i> variable. The default value is 72 for equal-width multicolumn output. There is no limit otherwise. If the -w flag is not specified and the -s flag is specified, the default width is 512 column positions.
- <i>Column</i>	Sets the number of columns to the value specified by the <i>Column</i> variable. The default value is 1. This option should not be used with the -m flag. The -e and -i flags are assumed for multicolumn output. A text column should never exceed the length of the page (see the -l flag). When this flag is used with the -t flag, use the minimum number of lines to write the output.
+ <i>Page</i>	Begins the display with the page number specified by the <i>Page</i> variable. The default value is 1.

See the **pr** command in the *AIX 5L Version 5.3 Commands Reference, Volume 4* for the complete syntax.

The following is a list of examples of how **pr** command flags can be used:

- To print a file named **prog.c** with headings and page numbers on the printer, type the following:

```
pr prog.c | qprt
```

This command adds page headings to **prog.c** and sends it to the **qprt** command. The heading consists of the date the file was last modified, the file name, and the page number.

- To specify a title for a file named **prog.c**, type the following:

```
pr -h "MAIN PROGRAM" prog.c | qprt
```

This prints **prog.c** with the title MAIN PROGRAM in place of the file name. The modification date and page number are still printed.

- To print a file named **word.lst** in multiple columns, type the following:

```
pr -3 word.lst | qprt
```

This prints the **word.lst** file in three vertical columns.

- To print several files side by side on the paper, type the following:

```
pr -m -h "Members and Visitors" member.lst visitor.lst | qprt
```

This prints **member.lst** and **visitor.lst** side by side with the title Members and Visitors.

- To modify a file named **prog.c** for later use, type the following:

```
pr -t -e prog.c > prog.notab.c
```

This command replaces tab characters in **prog.c** with spaces and puts the result in **prog.notab.c**. Tab positions are at columns 9, 17, 25, 33, and so on. The **-e** flag tells the **pr** command to replace the tab characters; the **-t** flag suppresses the page headings.

- To print a file named **myfile** in two columns, in landscape, and in 7-point text, type the following:

```
pr -l66 -w172 -2 myfile | qprt -z1 -p7
```

Command and flags for printing ASCII files on a PostScript printer

The Text Formatting System includes the `enscript` filter for converting ASCII print files to PostScript® for printing on a PostScript printer.

This filter is called by the `qrpt -da` command when submitting a print job to a PostScript print queue.

The following are prerequisites:

- The printer must be physically attached to your system.
- The printer must be configured and defined.
- The transcript portion of Text Formatting Services must be installed.

Several flags may be specified with the `qrpt` command to customize the output when submitting ASCII files to a PostScript print queue:

<code>-1+</code>	Adds page headings.
<code>-2+</code>	Formats the output in two columns.
<code>-3+</code>	Prints the page headings, dates, and page numbers in a fancy style. This is sometimes referred to as "gaudy" mode.
<code>-4+</code>	Prints the file, even if it contains unprintable characters.
<code>-5+</code>	Lists characters that are not included in a font.
<code>-h string</code>	Specifies a string to be used for page headings. If this flag is not specified, the heading consists of the file name, modification date, and page number.
<code>-l value</code>	Specifies the maximum number of lines printed per page. Depending on the point size, fewer lines per page may actually appear.
<code>-L!</code>	Truncates lines longer than the page width.
<code>-p</code>	Specifies the point size. If this flag is not specified, a point size of 10 is assumed, unless two-column rotated mode (<code>-2+ -z1</code>) is specified, in which case a value of 7 is used.
<code>-s</code>	Specifies the font style. If this flag is not specified, the Courier font is used. The PostScript printer must have access to the specified font. Acceptable values are: Courier-Oblique Helvetica Helvetica-Oblique Helvetica-Narrow Helvetica-Narrow-Oblique NewCenturySchlbk-Italic Optima Optima-Oblique Palatino-Roman Palatino-Italic Times-Roman Times-Italic
<code>-z1</code>	Rotates the output 90 degrees (landscape mode).

The following is a list of examples of how these `qrpt` command flags can be used:

- To send the ASCII file `myfile.ascii` to the PostScript printer named `Mpsps1`, type the following:
`qrpt -da -PMpsps1 myfile.ascii`

- For example, to send the ASCII file **myfile.ascii** to the PostScript printer named Msps1 and print out in the Helvetica font, type the following:

```
qprt -da -PMsps1 -sHelvetica myfile.ascii
```
- To send the ASCII file **myfile.ascii** to the PostScript printer named Msps1 and print out in the point size 9, type the following:

```
qprt -da -PMsps1 -p9 myfile.ascii
```

Automating the conversion of ASCII to PostScript

You can automate the conversion of an ASCII file to a PostScript file.

Many applications that generate PostScript print files follow the convention of making the first two characters of the PostScript file **%!**, which identifies the print file as a PostScript print file. To configure the system to detect ASCII print files submitted to a PostScript print queue and automatically convert them to PostScript files before sending them to the PostScript printer, perform these steps:

1. At the prompt, type the following:

```
smit chpq
```
2. Type the PostScript queue name, or use the List feature to select from a list of queues.
3. Select **Printer Setup** menu option.
4. Change the value of **AUTOMATIC detection of print file TYPE to be done?** field to yes.

Any of the following commands now convert an ASCII file to a PostScript file and print it on a PostScript printer. To convert **myfile.ascii**, type any of the following at the command line:

```
qprt -Pps myfile.ps myfile.ascii
lpr -Pps myfile.ps myfile.ascii
lp -dps myfile.ps myfile.ascii
```

where *ps* is a PostScript print queue.

Overriding automatic determination of print file types

You might need to override the automatic determination of print file type for PostScript printing in the following situations.

- To print a PostScript file named **myfile.ps** that does not begin with **%!**, type the following at the command line:

```
qprt -ds -Pps myfile.ps
```
- To print the source listing of a PostScript file named **myfile.ps** that begins with **%!**, type the following at the command line:

```
qprt -da -Pps myfile.ps
```

Command summary for printing

The following are commands for printing.

cancel	Cancels requests to a line printer.
lp	Sends requests to a line printer.
lpq	Examines the spool queue.
lpr	Enqueues print jobs.
lprm	Removes jobs from the line printer spooling queue.
lpstat	Displays line printer status information.
pr	Writes a file to standard output.

qcan	Cancels a print job.
qchk	Displays the status of a print queue.
qhld	Holds or releases a print job.
qmov	Moves a print job to another print queue.
qpri	Prioritizes a job in the print queue.
qprt	Starts a print job.

Chapter 32. File and system security

The goal of computer security is the protection of information stored on the computer system.

Information security is aimed at the following:

Integrity	The value of all information depends upon its accuracy. If unauthorized changes are made to data, this data loses some or all of its value.
Privacy	The value of much information depends upon its secrecy.
Availability	Information must be readily available.

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 up-front planning can save a lot of time later.

Security threats

Threats to information security can arise from several types of behavior.

Carelessness	Information security is often violated due to the carelessness of the authorized users of the system. For example, if you are careless with your password, no other security mechanisms can prevent unauthorized access to your account and data.
Browsing	Many security problems are caused by browsers, who are authorized users of the system exploring the system looking for carelessly protected data.
Penetration	Penetration represents deliberate attacks upon the system. An individual trying to penetrate the system will study it for security vulnerabilities and deliberately plan attacks designed to exploit those weaknesses.

Although system penetration usually represents the greatest threat to information security, do not underestimate problems caused by carelessness or browsing.

Basic security

Every system should maintain the level of security represented by the following basic security policies.

Basic security policies that should be followed are:

- “Backups for security”
- “Identification and authentication” on page 412
- “Login user IDs” on page 412
- “Unattended terminals” on page 412

Backups for security

Physically secure, reliable, and up-to-date system backups comprise the single most important security policy.

With a good system backup, you can recover from any system problems with minimal loss. Document your backup policy and include information regarding the following:

- How often backups will be made
- What types of backups (system, data, or incremental) will be made
- How backup tapes will be verified
- How backup tapes will be stored

For more information, see Chapter 6, “Backup methods,” on page 145.

Identification and authentication

Identification and authentication establish your identity.

You are required to log in to the system. You supply your user name 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, you are logged in to that account; you acquire the access rights and privileges of the account.

Because the password is the only protection for your account, select and guard your password carefully. Many attempts to break into a system start with attempts to guess passwords. The operating system provides significant password protection by storing user passwords separately from other user information. The encrypted passwords and other security-relevant data for users are stored in the **/etc/security/passwd** file. This file should be accessible only by the root user. With this restricted access to the encrypted passwords, an attacker cannot decipher the password with a program that simply cycles through all possible or likely passwords.

It is still possible to guess passwords by repeatedly attempting to log in to an account. If the password is trivial or is infrequently changed, such attempts might easily succeed.

Login user IDs

The operating system can identify users by their *login user ID*.

The login user ID allows the system to trace all user actions to their source. After a user logs in to the system and before the initial user program is run, the system sets the login ID of the process to the user ID found in the user database. All subsequent processes during the login session are tagged with this ID. These tags provide a trail of all activities performed by the login user ID.

You can reset the *effective user ID*, *real user ID*, *effective group ID*, *real group ID*, and *supplementary group ID* during the session, but you cannot change the login user ID.

Unattended terminals

All systems are vulnerable if terminals are left logged in and unattended. The most serious problem occurs when a system manager leaves a terminal unattended that has been enabled with root authority. In general, users should log out anytime they leave their terminals.

You can force a terminal to log out after a period of inactivity by setting the **TMOUT** and **TIMEOUT** parameters in the **/etc/profile** file. The **TMOUT** parameter works in the **ksh** (Korn) shell, and the **TIMEOUT** parameter works in the **bsh** (Bourne) shell. For more information about the **TMOUT** parameter, see “Parameter substitution in the Korn shell or POSIX shell” on page 449. For more information about the **TIMEOUT** parameter, see “Variable substitution in the Bourne shell” on page 492.

The following example, taken from a **.profile** file, forces the terminal to log out after an hour of inactivity:

```
T0=3600
echo "Setting Autologout to $T0"
TIMEOUT=$T0
TMOUT=$T0
export TIMEOUT TMOUT
```

Note: You can override the **TMOUT** and **TIMEOUT** values in the **/etc/profile** file by specifying different values in the **.profile** file in your home directory.

File ownership and user groups

Initially, a file's owner is identified by the user ID of the person who created the file.

The owner of a file determines who may read, write (modify), or execute the file. Ownership can be changed with the **chown** command.

Every user ID is assigned to a group with a unique group ID. The system manager creates the groups of users when setting up the system. When a new file is created, the operating system assigns permissions to the user ID that created it, to the group ID containing the file owner, and to a group called others, consisting of all other users. The `id` command shows your user ID (UID), group ID (GID), and the names of all groups you belong to.

In file listings (such as the listings shown by the `ls` command), the 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.

Changing file or directory ownership

Use the `chown` command to change the owner of your files.

When you specify the `-R` option, the `chown` command recursively descends through the directory structure from the specified directory. When symbolic links are encountered, the ownership of the file or directory pointed to by the link is changed; the ownership of the symbolic link is not changed.

Note: Only the root user can change the owner of another file. Errors are not displayed when the `-f` option is specified.

For example, to change the owner of the `program.c` file, type the following:

```
chown jim program.c
```

The user-access permissions for the `program.c` file now apply to jim. As the owner, jim can use the `chmod` command to permit or deny other users access to the `program.c` file.

See the `chown` command for the complete syntax.

File and directory access modes

Every file has an owner. For new files, the user who creates the file is the owner of that file. The owner assigns an *access mode* to the file. Access modes grant other system users permission to read, modify, or execute the file. Only the file's owner or users with root authority can change the access mode of a file.

There are the three classes of users: user/owner, group, and all others. Access is granted to these user classes in some combination of three modes: read, write, or execute. When a new file is created, the default permissions are read, write, and execute permission for the user who created the file. The other two groups have read and execute permission. The following table illustrates the default file-access modes for the three classes of user groups:

Classes	Read	Write	Execute
Owner	Yes	Yes	Yes
Group	Yes	No	Yes
Others	Yes	No	Yes

The system determines who has permission and the level of permission they have for each of these activities. Access modes are represented both symbolically and numerically in the operating system.

Symbolic representation of access modes

Access modes are represented symbolically.

- r** Indicates read permission, which allows users to view the contents of a file.
- w** Indicates write permission, which allows users to modify the contents of a file.

- x** Indicates execute permission. For executable files (ordinary files that contain programs), execute permission means that the program can be run. For directories, execute permission means the contents of the directory can be searched.

The access modes for files or directories are represented by nine characters. The first three characters represent the current **Owner** permissions, the second set of three characters represents the current **Group** permissions, and the third set of three characters represents the current settings for the **Other** permissions. A hyphen (-) in the nine-character set indicates that no permission is given. For example, a file with the access modes set to `rw-r-xr-x` gives read and execute permission to all three groups and write permission only to the owner of the file. This is the symbolic representation of the default setting.

The **ls** command, when used with the **-l** (lower case L) flag, gives a detailed listing of the current directory. The first 10 characters in the **ls -l** listing show the file type and permissions for each of the three groups. The **ls -l** command also lists the owner and group associated with each file and directory.

The first character indicates the type of file. The remaining nine characters contain the file permission information for each of the three classes of users. The following symbols are used to represent the type of file:

- Regular files
- d** Directory
- b** Block special files
- c** Character special files
- p** Pipe special files
- l** Symbolic links
- s** Sockets

For example, this is a sample **ls -l** listing:

```
-rwxrwxr-x 2 janet acct 512 Mar 01 13:33 january
```

Here, the first hyphen (-) indicates a regular file. The next nine characters (`rwxrwxr-x`) represent the User, Group, and Other access modes, as discussed above. `janet` is the file owner, and `acct` is the name of Janet's group. `512` is the file size in bytes, `Mar 01 13:33` is the last date and time of modification, and `january` is the file name. The `2` indicates how many links exist to the file.

Numeric representation of access modes

Numerically, read access is represented by a value of 4, write permission is represented by a value of 2, and execute permission is represented by a value of 1. The total value between 1 and 7 represents the access mode for each group (user, group, and other).

The following table illustrates the numeric values for each level of access:

Total Value	Read	Write	Execute
0	-	-	-
1	-	-	1
2	-	2	-
3	-	2	1
4	4	-	-
5	4	-	1
6	4	2	-
7	4	2	1

When a file is created, the default file access mode is 755. This means the user has read, write, and execute permissions (4+2+1=7), the group has read and execute permission (4+1=5), and all others have read and execute permission (4+1=5). To change access permission modes for files you own, run the **chmod** (change mode) command.

Displaying group information

Use the **lsgroup** command to display the attributes of all the groups on the system (or of specified groups). If one or more attributes cannot be read, the **lsgroup** command lists as much information as possible.

The attribute information displays as *Attribute=Value* definitions, each separated by a blank space.

1. To list all of the groups on the system, type the following:

```
lsgroup ALL
```

The system displays each group, group ID, and all of the users in the group in a list similar to the following:

```
system 0      arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
staff  1      john,ryan,flynn,daveb,jzitt,glover,maple,ken,gordon,mbrady
bin    2      root,bin
sys    3      root,su,bin,sys
```

2. To display specific attributes for all groups, do either of the following:

- You can list attributes in the form *Attribute=Value* separated by a blank space. This is the default style. For example, to list the ID and users for all of the groups on the system, type the following:

```
lsgroup -a id users ALL | pg
```

A list similar to the following is displayed:

```
system id=0 users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build
staff id=1 users=john,ryan,flynn,daveb,jzitt,glover,maple,ken
```

- You can also list the information in stanza format. For example, to list the ID and users for all of the groups on the system in stanza format, type the following:

```
lsgroup -a -f id users ALL | pg
```

A list similar to the following is displayed:

```
system:
  id=0
  users=pubs,ctw,geo,root,chucka,noer,su,dea,backup,build
```

```
staff:
  id=1
  users=john,ryan,flynn,daveb,jzitt,glover,maple,ken
```

```
bin:
  id=2
  users=root,bin
```

```
sys:
  id=3
  users=root,su,bin,sys
```

3. To display all attributes for a specific group, you can use one of two styles for listing specific attributes for all groups:

- You can list each attribute in the form *Attribute=Value* separated by a blank space. This is the default style. For example, to list all attributes for the group system, type the following:

```
lsgroup system
```

A list similar to the following is displayed:

```
system id=0 users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
```

- You can also list the information in stanza format. For example, to list all attributes for the group `bin` in stanza format, type the following:

```
lsgroup -f system
```

A list similar to the following is displayed:

```
system:
  id=0  users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
```

4. To list specific attributes for a specific group, type the following:

```
lsgroup -a Attributes Group
```

For example, to list the ID and users for group `bin`, type the following:

```
lsgroup -a id users bin
```

A list similar to the following is displayed:

```
bin id=2 users=root,bin
```

See the **lsgroup** command for the complete syntax.

Changing file or directory permissions

Use the **chmod** command to change the permissions of your files.

1. To add a type of permission to the **chap1** and **chap2** files, type the following:

```
chmod g+w chap1 chap2
```

This adds write permission for group members to the files **chap1** and **chap2**.

2. To make several permission changes at once to the `mydir` directory, type the following:

```
chmod go-w+x mydir
```

This denies (-) group members (**g**) and others (**o**) the permission to create or delete files (**w**) in the **mydir** directory and allows (+) group members and others to search the **mydir** directory or use (**x**) it in a path name. This is equivalent to the following command sequence:

```
chmod g-w mydir
chmod o-w mydir
chmod g+x mydir
chmod o+x mydir
```

3. To permit only the owner to use a shell procedure named **cmd** as a command, type the following:

```
chmod u=rwx,go= cmd
```

This gives read, write, and execute permission to the user who owns the file (**u=rwx**). It also denies the group and others the permission to access `cmd` in any way (**go=**).

4. To use the numeric mode form of the **chmod** command to change the permissions of the **text**, file type the following:

```
chmod 644 text
```

This sets read and write permission for the owner, and it sets read-only mode for the group and others.

See the **chmod** command for the complete syntax.

Access control lists

Access control consists of protected information resources that specify who can be granted access to such resources.

The operating system allows for need-to-know or discretionary security. The owner of an information resource can grant other users read or write access rights for that resource. A user who is granted access

rights to a resource can transfer those rights to other users. This security allows for user-controlled information flow in the system; the owner of an information resource defines the access permissions to the object.

Users have user-based access only to the objects that they own. Typically, users receive either the group permissions or the default permissions for a resource. The major task in administering access control is to define the group memberships of users, because these memberships determine the users' access rights to the files that they do not own.

Access control lists for file system objects

File system objects are typically associated with an Access Control List (ACL), which normally consists of series of Access Control Entries (ACEs). Each ACE defines the identity and its related access rights.

To maintain access control lists, use the **aclget**, **acledit**, **aclput**, and the **aclconvert** commands. For information about these commands, see **aclget**, **acledit**, **aclput** and **aclconvert**.

Note that ACL is typically stored and managed on the media by the physical file system (PFS). AIX 5.3 provides an infrastructure for physical file systems to support and manage multiple ACL types. The JFS2 file system shipped with AIX supports two ACL types:

- AIXC
- NFS4

Earlier file systems supported only the AIXC ACL type as in the previous AIX releases. These ACL types are discussed in detail in the *Security*.

AIXC access control list type

The AIXC (AIX Classic) ACL type provides for the ACL behavior as defined on previous releases of AIX. This ACL type consists of the regular base mode bits and extended permissions (ACEs).

With extended permissions, you can permit or deny file access to specific individuals or groups without changing the base permissions.

Note: The AIXC ACL for a file cannot exceed one memory page (approximately 4096 bytes) in size.

The **chmod** command in numeric mode (with octal notations) can set base permissions and attributes. The **chmod** subroutine, which the command calls, disables extended permissions. Extended permissions are disabled if you use the numeric mode of the **chmod** command on a file that has an ACL. The symbolic mode of the **chmod** command does not disable extended permissions when the ACL associated is of type AIXC. For more information on numeric and symbolic mode, refer to the **chmod** command. For information about the **chmod** command, see **chmod**.

Base permissions

AIXC ACL specific base permissions are the traditional file-access modes assigned to the file owner, file group, and other users. The access modes are read (r), write (w), and execute/search (x).

Note: AIXC ACL type Base Permissions will be same as the file mode bits stored in the file system object's inode headers. That is, the information in base mode bits is same as the value returned by file system when **stat** is performed on the file system object.

In an access control list, base permissions are in the following format, with the **Mode** parameter expressed as rwx (with a hyphen (-) replacing each unspecified permission):

```
base permissions:
  owner(name): Mode
  group(group): Mode
  others: Mode
```

Attributes

Three attributes can be added to an access control list:

setuid (SUID)

Set-user-ID mode bit. This attribute sets the effective and saved user IDs of the process to the owner ID of the file on execution.

setgid (SGID)

Set-group-ID mode bit. This attribute sets the effective and saved group IDs of the process to the group ID of the file on execution.

savetext (SVTX)

Saves the text in a text file format.

The above attributes are added in the following format:

```
attributes: SUID, SGID, SVTX
```

Extended permissions

AIXC ACL extended permissions allow the owner of a file to more precisely define access to that file. Extended permissions modify the base file permissions (owner, group, others) by permitting, denying, or specifying access modes for specific individuals, groups, or user and group combinations. Permissions are modified through the use of keywords.

The permit, deny, and specify keywords are defined as follows:

permit

Grants the user or group the specified access to the file

deny Restricts the user or group from using the specified access to the file

specify

Precisely defines the file access for the user or group

If a user is denied a particular access by either a deny or a specify keyword, no other entry can override that access denial.

The enabled keyword must be specified in the ACL for the extended permissions to take effect. The default value is the disabled keyword.

In an AIXC ACL, extended permissions are in the following format:

```
extended permissions:
  enabled | disabled
  permit  Mode  UserInfo...:
  deny    Mode  UserInfo...:
  specify Mode  UserInfo...:
```

Use a separate line for each permit, deny, or specify entry. The **Mode** parameter is expressed as rwx (with a hyphen (-) replacing each unspecified permission). The **UserInfo** parameter is expressed as u:UserName, or g:GroupName, or a comma-separated combination of u:UserName and g:GroupName.

Note: If more than one user name is specified in an entry, that entry cannot be used in an access control decision because a process has only one user ID.

NFS4 access control list type

JFS2 file system in AIX also supports NFS4 ACL type. This ACL implementation follows the ACL definition as specified in NFS4 version 4 protocol related RFC.

This ACL provides much finer granular control over the access rights and also provides for features such as inheritance. NFS4 ACL consists of an array of ACEs. Each ACE defines access rights for an identity. As defined in the RFC, the main components of NFS4 ACE are as follows:

```
struct nfsace4 {  
  
    acetyp4         type;  
    aceflag4        flag;  
    acemask4        access_mask;  
    utf8str_mixed   who;  
  
};
```

Where:

type Bit mask that defines the type of the ACE. Details such as whether this ACE allows access or denies access are defined here.

flag Bit mask that describes the inheritance aspects of the ACE. Defines whether this ACE is applicable to the file system object, or its children, or both.

access_mask

Bit mask that defines various access rights possible. Rights defined include, read, write, execute, create, delete, create child, delete child, etc.

who This null-terminated string defines the identity of the person to which this ACE will apply. Note that per RFC, the size of this string is unlimited, and a loose definition allows for defining domains within NFS version 4 networks to manage access control. Natively (most of the time) AIX does not interpret this string, and each ACE is associated with an AIX-understood identity (such as **uid** or **gid**). It is expected that the NFS version 4 file system will interpret these strings as necessary to convert them to OS-understood user or group IDs. AIX only understands some of the special **who** strings defined in the RFC.

In AIX, use the **aclget**, **acledit**, **aclput**, and the **aclconvert** commands to manage NFS4 ACLs. For information about these commands, see **aclget**, **acledit**, **aclput** and **aclconvert**.

Note: Any type of **chmod** command will erase the file's ACL.

Access control list example for AIXC

The following is an example of an AIXC access control list (ACL).

The following is an example of an AIXC ACL:

```
attributes: SUID  
base permissions:  
  owner(frunk): rw-  
  group(system): r-x  
  others: ---  
extended permissions:  
  enabled  
  permit rw- u:dhs  
  deny r-- u:chas, g:system  
  specify r-- u:john, g:gateway, g:mail  
  permit rw- g:account, g:finance
```

The parts of the ACL and their meanings are as follows:

- The first line indicates that the setuid bit is turned on.
- The next line, which introduces the base permissions, is optional.
- The next three lines specify the base permissions. The owner and group names in parentheses are for information only. Changing these names does not alter the file owner or file group. Only the **chown** command and the **chgrp** command can change these file attributes. For more information about these commands, see **chown** and **chgrp**.

- The next line, which introduces the extended permissions, is optional.
- The next line indicates that the extended permissions that follow are enabled.
- The last four lines are the extended entries.
- The first extended entry grants user dhs read (r) and write (w) permission on the file.
- The second extended entry denies read (r) access to user chas only when he is a member of the system group.
- The third extended entry specifies that as long as user john is a member of both the gateway group and the mail group, this user has read (r) access. If user john is not a member of both groups, this extended permission does not apply.
- The last extended entry grants any user in **both** the account group and the finance group read (r) and write (w) permission.

Note: More than one extended entry can be applied to a process, with restrictive modes taking precedence over permissive modes.

For more information and a complete syntax, see the **acledit** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1*.

Access control list access authorization

The owner of the information resource is responsible for managing access rights. Resources are protected by permission bits, which are included in the mode of the object.

For AIXC ACL, the permission bits define the access permissions granted to the owner of the object, the group of the object, and for the others default class. AIXC ACL type supports three different modes of access (read, write, and execute) that can be granted separately.

When a user logs in to an account (using the **login** or **su** command), the user IDs and group IDs assigned to that account are associated with the user's processes. These IDs determine the access rights of the process.

For files, directories, named pipes, and devices (special files) with an associated AIX ACL, access is authorized as follows:

- For each access control entry (ACE) in the access control list (ACL), the identifier list is compared to the identifiers of the process. If there is a match, the process receives the permissions and restrictions defined for that entry. The logical unions for both permissions and restrictions are computed for each matching entry in the ACL. If the requesting process does not match any of the entries in the ACL, it receives the permissions and restrictions of the default entry.
- If the requested access mode is permitted (included in the union of the permissions) and is not restricted (included in the union of the restrictions), access is granted. Otherwise, access is denied.

Further, for an AIXC ACL type, the identifier list of an ACL matches a process if all identifiers in the list match the corresponding type of effective identifier for the requesting process. A USER-type identifier matched is equal to the effective user ID of the process, and a GROUP-type identifier matches if it is equal to the effective group ID of the process or to one of the supplementary group IDs. For instance, an ACE with an identifier list such as the following:

```
USER:fred, GROUP:philosophers, GROUP:software_programmer
```

would match a process with an effective user ID of fred and a group set of:

```
philosophers, philanthropists, software_programmer, doc_design
```

but would not match for a process with an effective user ID of fred and a group set of:

```
philosophers, iconoclasts, hardware_developer, graphic_design
```

Note that an ACE with an identifier list of the following would match for both processes:

```
USER:fred, GROUP:philosophers
```

In other words, the identifier list in the ACE functions is a set of conditions that must hold for the specified access to be granted.

The discretionary access control mechanism allows for effective access control of information resources and provides for separate protection of the confidentiality and integrity of the information. Owner-controlled access control mechanisms are only as effective as users make them. All users must understand how access permissions are granted and denied, and how these are set.

Note that file system objects with an associated NFS4 ACL type, access checks are based on various ACEs that form the ACL as per the rules setup in the NFS version 4 protocol-related RFC. Identity matching is done based on the user ID or group ID or special who strings defined in the ACE against the process's credentials. If a match occurs, the access rights requested are checked against the access rights defined in the ACE. If any of the access rights are allowed, those will be taken out, and the compare operation continues on to the next ACE. This process is continued until either the ACL end is reached, or all the access rights are met, or if any of the access rights requested are denied. The following steps capture the access checking in the case of a file system object with an associated NFS4 ACL:

1. For each access control entry (ACE) in the access control list (ACL), the identifier list is compared to the identifiers of the process. Identity checks include the user ID or group ID defined in the ACE. Also, if the identity is defined as **special** with strings such as OWNER@, a match will occur if the calling process is by the owner of the file. If there is a match, the process receives the access rights defined for that entry. Else, continue to the next ACE.
2. Requested access rights are compared with the access rights retrieved from ACE entry. If any of the access rights requested are explicitly denied by the ACE, then the access checking process is ended, and the requesting process will be denied access.
3. If some of the requested access rights are met by the ACE, then those access rights will be taken out from the requests access rights list, and the compare operation continues to the next ACE.
4. If all of the requested access rights are met by the ACEs, then the requested access is allowed.
5. If ACL end is reached before all of the requested access rights are resolved, then the access is denied.

Note that apart from the ACL type-based access checks, individual physical file systems might also choose to provide for privilege-based access to the file system objects. For example, an owner might always at least have the permission to modify the ACL, irrespective of the existing ACL access rights. A process with a user ID of 0 is known as a root user process. These processes are generally allowed all access permissions. However, if a root user process requests execute permission for a program, access is granted only if execute permission is granted to at least one user.

All access permission checks for these objects are made at the system call level when the object is first accessed. Because System V Interprocess Communication (SVIPC) objects are accessed statelessly, checks are made for every access. However, it is possible that checks are made by the physical file systems at open time of the file system object and not at the time of read or write operation. For objects with file system names, it is necessary to be able to resolve the name of the actual object. Names are resolved either relatively (to the process' working directory) or absolutely (to the process' root directory). All name resolution begins by searching one of these.

Command for displaying access control information (**aclget** command)

The **aclget** command displays the access control information of a file. The information that you view includes attributes, base permissions, and extended permissions.

For example, to display the access control information for the **status** file, type the following:

```
aclget status
```

The access control information that displays includes a list of attributes, base permissions, and extended permissions. For an example, see “Access control list example and description.”

See the **aclget** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Setting access control information (aclput command)

To set the access control information for a file, use the **aclput** command.

Note: The access control list for a file cannot exceed one memory page (approximately 4096 bytes) in size.

See the following examples:

For example, to set the access control information for the status file with the access control information stored in the **acldefs** file, type the following:

```
aclput -i acldefs status
```

To set the access control information for the status file with the same information used for the **plans** file, type the following:

```
aclget plans | aclput status
```

For more information and the complete syntax, see the **aclput** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1*.

Access control list example and description

The following is an example and description of access control lists (ACLs).

The following is an example of an ACL:

```
attributes: SUID
base permissions:
  owner(frank): rw-
  group(system): r-x
  others: ---
extended permissions:
  enabled
  permit rw- u:dhs
  deny r-- u:chas, g:system
  specify r-- u:john, g:gateway, g:mail
  permit rw- g:account, g:finance
```

The parts of the ACL and their meanings are the following:

- The first line indicates that the **setuid** bit is turned on.
- The next line, which introduces the base permissions, is optional.
- The next three lines specify the base permissions. The owner and group names in parentheses are for information only. Changing these names does not alter the file owner or file group. Only the **chown** command and the **chgrp** command can change these file attributes. For more information about these commands, see **chown** and **chgrp**.
- The next line, which introduces the extended permissions, is optional.
- The next line indicates that the extended permissions that follow are enabled.
- The last four lines are the extended entries. The first extended entry grants user dhs read (r) and write (w) permission on the file.
- The second extended entry denies read (r) access to user chas only when he is a member of the system group.

- The third extended entry specifies that as long as user john is a member of both the gateway group and the mail group, has read (r) access. If user john is not a member of both groups, this extended permission does not apply.
- The last extended entry grants any user in **both** the account group and the finance group read (r) and write (w) permission.

Note: More than one extended entry can be applied to a process, with restrictive modes taking precedence over permissive modes.

See the **acledit** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Editing access control information (acledit command)

Use the **acledit** command to change the access control information of a file. The command displays the current access control information and lets the file owner change it.

Before making any changes permanent, the command asks if you want to proceed. For information about the **acledit** command, see **acledit**.

Note: The *EDITOR* environment variable must be specified with a complete path name; otherwise, the **acledit** command will fail.

The access control information that displays is ACL type specific and includes a list of attributes, base permissions, and extended permissions. For an example, see “Access control list example and description” on page 422.

For example, to edit the access control information of the **plans** file, type the following:

```
acledit plans
```

See the **acledit** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

Locking your terminal (lock or xlock command)

Use the **lock** command to lock your terminal. The **lock** command requests your password, reads it, and requests the password a second time to verify it.

In the interim, the command locks the terminal and does not relinquish it until the password is received the second time. The timeout default value is 15 minutes, but this can be changed with the *-Number* flag.

Note: If your interface is AIXwindows, use the **xlock** command in the same manner.

For example, to lock your terminal under password control, type the following:

```
lock
```

You are prompted for the password twice so the system can verify it. If the password is not repeated within 15 minutes, the command times out.

To reserve a terminal under password control with a timeout interval of 10 minutes, type the following:

```
lock -10
```

See the **lock** or the **xlock** command in the *AIX 5L Version 5.3 Commands Reference* for the complete syntax.

Command summary for file and system security

The following are commands for file system and security.

acledit	Edits the access control information of a file
aclget	Displays the access control information of a file
aclput	Sets the access control information of a file
chmod	Changes permission modes
chown	Changes the user associated with a file
lock	Reserves a terminal
lsgroup	Displays the attributes of groups
xlock	Locks the local X display until a password is entered

Chapter 33. User environment customization

The operating system provides various commands and initialization files that enable you to customize the behavior and the appearance of your user environment.

You can also customize some of the default resources of the applications you use on your system. Defaults are initiated by the program at startup. When you change the defaults, you must exit and then restart the program for the new defaults take effect.

For information about customizing the behavior and appearance of the Common Desktop Environment, see the *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*.

System startup files

When you log in, the shell defines your user environment after reading the initialization files that you have set up. The characteristics of your user environment are defined by the values given to your environment variables. You maintain this environment until you log out of the system.

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 shell first runs the commands to set up your system environment in the **/etc/environment** file and then evaluates the commands contained in the **/etc/profile** file. After these files are run, the system then checks to see if you have a **.profile** file in your home directory. If the **.profile** file exists, the system runs this file. The **.profile** file will specify if an environment file also exists. If an environment file exists (usually named **.env**), the system then runs this file and sets up your environment variables.

The **/etc/environment**, **/etc/profile**, and **.profile** files are run once at login time. The **.env** file, on the other hand, is run every time you open a new shell or a window.

/etc/environment file

The first file that the operating system uses at login time is the **/etc/environment** file. The **/etc/environment** file contains variables specifying the basic environment for all processes.

When a new process begins, the **exec** subroutine makes an array of strings available that have the form *Name=Value*. This array of strings is called the *environment*. Each name defined by one of the strings is called an *environment variable* or *shell variable*. The **exec** subroutine allows the entire environment to be set at one time.

When you log in, the system sets environment variables from the **/etc/environment** file before reading your login profile, named **.profile**. The following variables make up the basic environment:

HOME	The full path name of the user's login or HOME directory. The login program sets this to the name specified in the /etc/passwd file.
LANG	The locale name currently in effect. The LANG variable is initially set in the /etc/profile file at installation time.
NLSPATH	The full path name for message catalogs.
LOCPATH	The full path name of the location of National Language Support tables.
PATH	The sequence of directories that commands, such as sh , time , nice and nohup , search when looking for a command whose path name is incomplete.

TZ	The time zone information. The TZ environment variable is initially set by the /etc/profile file, the system login profile.
-----------	---

For detailed information about the **/etc/environment** file, see the *AIX 5L Version 5.3 Files Reference*.

/etc/profile file

The second file that the operating system uses at login time is the **/etc/profile** file.

The **/etc/profile** 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 **/etc/profile** file for all users on the system. Only the system administrator can change this file.

The following example is a typical **/etc/profile** file:

```
#Set file creation mask
unmask 022
#Tell me when new mail arrives
MAIL=/usr/mail/$LOGNAME
#Add my /bin directory to the shell search sequence
PATH=/usr/bin:/usr/sbin:/etc::
#Set terminal type
TERM=1ft
#Make some environment variables global
export MAIL PATH TERM
```

For detailed information about the **/etc/profile** file, see the *AIX 5L Version 5.3 Files Reference*.

.profile file

The **.profile** file is present in your home (**\$HOME**) directory and lets you customize your individual working environment.

Because the **.profile** file is hidden, use the **ls -a** command to list it.

After the **login** program adds the **LOGNAME** (login name) and **HOME** (login directory) variables to the environment, the commands in the **\$HOME/.profile** file are executed if the file is present. The **.profile** file contains your individual profile that overrides the variables set in the **/etc/profile** file. The **.profile** file is often used to set exported environment variables and terminal modes. You can customize your environment by modifying the **.profile** file. Use the **.profile** file to control the following defaults:

- Shells to open
- Prompt appearance
- Keyboard sound

The following example is a typical **.profile** file:

```
PATH=/usr/bin:/etc:/home/bin1:/usr/lpp/tps4.0/user::
epath=/home/gsc/e3:
export PATH epath
csh
```

This example has defined two path variables (**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 **/etc/profile** file) to determine login shell variables. You can also customize other shell environments. For example, use the **.cshrc** file and **.kshrc** file to customize a C shell and a Korn shell, respectively, when each type of shell is started.

.env file

A fourth file that the operating system uses at login time is the **.env** file, if your **.profile** contains the following line: `export ENV=$HOME/.env`

The **.env** file lets you customize your individual working environment variables. Because the **.env** file is hidden, use the **ls -a** command to list it. For more information about the **ls** command, see **ls**. The **.env** file contains the individual user environment variables that override the variables set in the **/etc/environment** file. You can customize your environment variables as desired by modifying your **.env** file.

The following example is a typical **.env** file:

```
export myid=`id | sed -n -e 's/).*$/' -e 's/^.*(//p'`
#set prompt: login & system name & path
if [ $myid = root ]
    then typeset -x PSCH='#:\${PWD}> '
        PS1="#:\${PWD}> "
    else typeset -x PSCH='>'
        PS1="$LOGNAME@$UNAME:\${PWD}> "
        PS2=">"
        PS3="#?"
fi
export PS1 PS2 PS3
#setup my command aliases
alias ls="/bin/ls -CF" \
      d="/bin/ls -Fa1 | pg" \
      rm="/bin/rm -i" \
      up="cd .."
```

Note: When modifying the **.env** file, ensure that newly created environment variables do not conflict with standard variables such as *MAIL*, *PS1*, *PS2*, and *IFS*.

AIXwindows startup files

Different computer systems have different ways of starting the X Server and AIXwindows.

Because different computer systems have different ways of starting the X Server and AIXwindows, consult with your system administrator to learn how to get started. Usually, the X Server and AIXwindows are started from a shell script that runs automatically when you log in. You might, however, find that you need to start the X Server or AIXwindows, or both.

If you log in and find that your display is functioning as a single terminal with no windows displayed, you can start the X Server by typing the following:

```
xinit
```

Note: Before entering this command, make sure that the pointer rests within a window that has a system prompt.

If this command does not start the X Server, check with your system administrator to ensure that your search path contains the X11 directory containing executable programs. The appropriate path might differ from one system to another.

If you log in and find one or more windows without frames, you can start AIXwindows Window Manager by typing the following:

```
mwm &
```

Because AIXwindows permits customization both by programmers writing AIXwindows applications and by users, you might find that mouse buttons or other functions do not operate as you might expect from reading this documentation. You can reset your AIXwindows environment to the default behavior by pressing and holding the following four keys:

Alt-Ctrl-Shift-!

You can return to the customized behavior by pressing this key sequence again. If your system does not permit this combination of keystrokes, you can also restore default behavior from the default root menu.

.xinitrc file

The **xinit** command uses a customizable shell script file that lists the X Client programs to start. The **.xinitrc** file in your home directory controls the windows and applications that start when you start AIXwindows.

The **xinit** command works with shell scripts in the following order:

1. The **xinit** command first looks for the **\$XINITRC** environment variable to start AIXwindows.
2. If the **\$XINITRC** environment variable is not found, the **xinit** command looks for the **\$HOME/.xinitrc** shell script.
3. If the **\$HOME/.xinitrc** shell script is not found, the **xinit** command starts the **/usr/lib/X11/\$LANG/xinitrc** shell script.
4. If **/usr/lib/X11/\$LANG/xinitrc** is not found, it looks for the **/usr/lpp/X11/defaults/\$LANG/xinitrc** shell script. If that script is not found, it searches for the **/usr/lpp/X11/defaults/xinitrc** shell script.
5. The **xinitrc** shell script starts commands, such as the **mwm** (AIXwindows Window Manager), **aixterm**, and **xclock** commands.

The **xinit** command performs the following operations:

- Starts an X Server on the current display
- Sets up the **\$DISPLAY** environment variable
- Runs the **xinitrc** file to start the X Client programs

The following example shows the part of the **xinitrc** file you can customize:

```
# This script is invoked by /usr/lpp/X11/bin/xinit

.
.
.
*****
# Start the X clients. Change the following lines to      *
# whatever command(s) you desire!                       *
# The default clients are an analog clock (xclock), an lft *
# terminal emulator (aixterm), and the Motif Window Manager *
# (mwm). *
*****
exec mwm
```

.Xdefaults file

If you work in an AIXwindows interface, you can customize this interface with the **.Xdefaults** file. AIXwindows allows you to specify your preferences for visual characteristics, such as colors and fonts.

Many aspects of a windows-based application's appearance and behavior are controlled by sets of variables called *resources*. The visual or behavioral aspect of a resource is determined by its assigned value. There are several different types of values for resources. For example, resources that control color can be assigned predefined values such as *DarkSlateBlue* or *Black*. Resources that specify dimensions are assigned numeric values. Some resources take Boolean values (*True* or *False*).

If you do not have a **.Xdefaults** file in your home directory, you can create one with any text editor. After you have this file in your home directory, you can set resource values in it as you wish. A sample default file called **Xdefaults.tmpl** is in the **/usr/lpp/X11/defaults** directory.

The following example shows part of a typical **.Xdefaults** file:

```
*AutoRaise: on
*DeIconifyWarp: on
*warp: on
*TitleFont: andysans12
*scrollBar: true
*font: Rom10.500
Mwm*menu*foreground: black
Mwm*menu*background: CornflowerBlue
Mwm*menu*RootMenu*foreground: black
Mwm*menu*RootMenu*background: CornflowerBlue
Mwm*icon*foreground: grey25
Mwm*icon*background: LightGray
Mwm*foreground: black
Mwm*background: LightSkyBlue
Mwm*bottomShadowColor: Blue1
Mwm*topShadowColor: CornflowerBlue
Mwm*activeForeground: white
Mwm*activeBackground: Blue1
Mwm*activeBottomShadowColor: black
Mwm*activeTopShadowColor: LightSkyBlue
Mwm*border: black
Mwm*highlight: white

aixterm.foreground: green
aixterm.background: black
aixterm.fullcursor: true
aixterm.ScrollKey: on
aixterm.autoRaise: true
aixterm.autoRaiseDelay: 2
aixterm.boldFont: Rom10.500
aixterm.geometry: 80x25
aixterm.iconFont: Rom8.500
aixterm.iconStartup: false
aixterm.jumpScroll: true
aixterm.reverseWrap: true
aixterm.saveLines: 500
aixterm.scrollInput: true
aixterm.scrollKey: false
aixterm.title: AIX
```

.mwmrc file

Most of the features that you want to customize can be set with resources in your **.Xdefaults** file. However, key bindings, mouse button bindings, and menu definitions for your window manager are specified in the supplementary **.mwmrc** file, which is referenced by resources in the **.Xdefaults** file.

If you do not have a **.mwmrc** file in your home directory, you can copy it as follows:

```
cp /usr/lib/X11/system.mwmrc .mwmrc
```

Because the **.mwmrc** file overrides the system-wide effects of the **system.mwmrc** file, your specifications do not interfere with the specifications of other users.

The following example shows part of a typical **system.mwmrc** file:

```
# DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc)
#
# menu pane descriptions
#
# Root Menu Description
```

```

Menu RootMenu
{ "Root Menu"      f.title
  no-label        f.separator
  "New Window"    f.exec "aixterm &"
  "Shuffle Up"    f.circle_up
  "Shuffle Down" f.circle_down
  "Refresh"       f.refresh
  no-label        f.separator
  "Restart"       f.restart
  "Quit"          f.quit_mwm
}

# Default Window Menu Description

Menu DefaultWindowMenu MwmWindowMenu
{ "Restore"  _R  Alt<Key>F5    f.normalize
  "Move"     _M  Alt<Key>F7    f.move
  "Size"     _S  Alt<Key>F8    f.resize
  "Minimize" _n  Alt<Key>F9    f.minimize
  "Maximize" _x  Alt<Key>F10   f.maximize
  "Lower"    _L  Alt<Key>F3    f.lower
  no-label   f.separator
  "Close"    _C  Alt<Key>F4    f.kill
}

# no acclerator window menu
Menu NoAccWindowMenu
{
  "Restore"  _R    f.normalize
  "Move"     _M    f.move
  "Size"     _S    f.resize
  "Minimize" _n    f.minimize
  "Maximize" _x    f.maximize
  "Lower"    _L    f.lower
  no-label   f.separator
  "Close"    _C    f.kill
}

Keys DefaultKeyBindings
{
  Shift<Key>Escape    icon|window    f.post_wmenu
  Meta<Key>space      icon|window    f.post_wmenu
  Meta<Key>Tab        root|icon|window f.next_key
  Meta Shift<Key>Tab  root|icon|window f.prev_key
  Meta<Key>Escape     root|icon|window f.next_key
  Meta Shift<Key>Escape root|icon|window f.prev_key
  Meta Ctrl Shift<Key>exclam root|icon|window f.set_behavior
}

#
# button binding descriptions
#

Buttons DefaultButtonBindings
{
  <Btn1Down>      frame|icon    f.raise
  <Btn3Down>      frame|icon    f.post_wmenu
  <Btn1Down>      root          f.menu RootMenu
  <Btn3Down>      root          f.menu RootMenu
  Meta<Btn1Down>  icon|window    f.lower
  Meta<Btn2Down>  window|icon    f.resize
  Meta<Btn3Down>  window          f.move
}

Buttons PointerButtonBindings
{
  <Btn1Down>      frame|icon    f.raise
  <Btn2Down>      frame|icon    f.post_wmenu
  <Btn3Down>      frame|icon    f.lower
}

```

```

<Btn1Down>      root          f.menu  RootMenu
Meta<Btn2Down>  window|icon  f.resize
Meta<Btn3Down>  window|icon  f.move
}
#
# END OF mwm RESOURCE DESCRIPTION FILE
#

```

Exporting shell variables (export shell command)

A *local* shell variable is a variable known only to the shell that created it. If you start a new shell, the old shell's variables are unknown to it. If you want the new shells that you open to use the variables from an old shell, export the variables to make them *global*.

You can use the **export** command to make local variables global. To make your local shell variables global automatically, export them in your **.profile** file.

Note: Variables can be exported down to child shells but not exported up to parent shells.

See the following examples:

- To make the local shell variable *PATH* global, type the following:
export path
- To list all your exported variables, type the following:
export

The system displays information similar to the following:

```

DISPLAY=unix:0
EDITOR=vi
ENV=$HOME/.env
HISTFILE=/u/denise/.history
HISTSIZE=500
HOME=/u/denise
LANG=En_US
LOGNAME=denise
MAIL=/usr/mail/denise
MAILCHECK=0
MAILMSG=**YOU HAVE NEW MAIL.
USE THE mail COMMAND TO SEE YOUR MAILPATH=/usr/mail/denise?denise has mail !!!
MAILRECORD=/u/denise/.Outmail
PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/u/bin1
PWD=/u/denise
SHELL=/bin/ksh

```

Changing the default font (chfont command)

To change the default font at system startup, use the **chfont** or **smit** command. A *font palette* is a file that the system uses to define and identify the fonts it has available.

Note: To run the **chfont** command, you must have root authority.

chfont command

See the following examples on how to use the **chfont** command:

- To change the active font to the fifth font in the font palette, type the following:
chfont -a5
- To change the font to an italic, roman, and bold face of the same size, type the following:
chfont -n /usr/lpp/fonts/It114.snf /usr/lpp/fonts/Bld14.snf /usr/lpp/fonts/Rom14.snf

See the **chfont** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1* for the complete syntax.

smit command

The **chfont** command can also be run using **smit**.

To select the active font, type the following:

```
smit chfont
```

To select the font palette, type the following:

```
smit chfontpl
```

Changing control keys (stty command)

To change the keys that your terminal uses for control keys, use the **stty** command.

Your changes to control keys remain in effect until you log out. To make your changes permanent, place them in your **.profile** file.

See the following examples:

- To assign Ctrl-Z as the interrupt key, type the following:

```
stty intr ^Z
```

Be sure to place a space character between `intr` and `^Z`.

- To reset all control keys to their default values, type the following:

```
stty sane
```

- To display your current settings, type the following:

```
stty -a
```

See the **stty** command in the *AIX 5L Version 5.3 Commands Reference, Volume 5* for the complete syntax.

Changing your system prompt

You can change your system prompt.

Your shell uses the following prompt variables:

PS1	Prompt used as the normal system prompt
PS2	Prompt used when the shell expects more input
PS3	Prompt used when you have root authority

You can change any of your prompt characters by changing the value of its shell variable. Your prompt changes remain in effect until you log out. To make your changes permanent, place them in your **.env** file.

See the following examples:

- To display the current value of the **PS1** variable, type the following:

```
echo "prompt is $PS1"
```

The system displays information similar to the following:

```
prompt is $
```

- To change your prompt to Ready>, type the following:

```
PS1="Ready> "
```

- To change your continuation prompt to Enter more->, type the following:
PS2="Enter more->"
- To change your root prompt to Root->, type the following:
PS3="Root-> "

Chapter 34. Shells

Your interface to the operating system is called a *shell*.

The shell is the outermost layer of the operating system. Shells incorporate a programming language to control processes and files, as well as to start and control other programs. The shell manages the interaction between you and the operating system by prompting you for input, interpreting that input for the operating system, and then handling any resulting output from the operating system.

Shells provide a way for you to communicate with the operating system. This communication is carried out either interactively (input from the keyboard is acted upon immediately) or as a shell script. A *shell script* is a sequence of shell and operating system commands that is stored in a file.

When you log in to the system, the system locates the name of a shell program to execute. After it is executed, the shell displays a command prompt. This prompt is usually a \$ (dollar sign). When you type a command at the prompt and press the Enter key, the shell evaluates the command and attempts to carry it out. Depending on your command instructions, the shell writes the command output to the screen or redirects the output. It then returns the command prompt and waits for you to type another command.

A *command line* is the line on which you type. It contains the shell prompt. The basic format for each line is as follows:

```
$ Command Argument(s)
```

The shell considers the first word of a command line (up to the first blank space) as the command and all subsequent words as arguments.

Note: When **libc.a** is moved or renamed, the Killed error message is displayed from the shell because there is no **libc.a** file available for the system to load and run the utilities. The **recsh** command invokes the recovery shell, which provides an ability to rename **libc.a** if it is accidentally moved.

Shell features

There are advantages to using the shell as an interface to the system.

The primary advantages of interfacing to the system through a shell are as follows:

- **Wildcard substitution in file names (pattern-matching)**

Carries out commands on a group of files by specifying a pattern to match, rather than specifying an actual file name.

For more information, see:

- “File name substitution in the Korn shell or POSIX shell” on page 456
- “File name substitution in the Bourne shell” on page 497
- “File name substitution in the C shell” on page 513

- **Background processing**

Sets up lengthy tasks to run in the background, freeing the terminal for concurrent interactive processing.

For more information, see the **bg** command in the following:

- “Job control in the Korn shell or POSIX shell” on page 472
- “C shell built-in commands” on page 500

Note: The Bourne shell does not support job control.

- **Command aliasing**

Gives an alias name to a command or phrase. When the shell encounters an alias on the command line or in a shell script, it substitutes the text to which the alias refers.

For more information, see:

- “Command aliasing in the Korn shell or POSIX shell” on page 447
- “Alias substitution in the C shell” on page 511

Note: The Bourne shell does not support command aliasing.

- **Command history**

Records the commands you enter in a history file. You can use this file to easily access, modify, and reissue any listed command.

For more information, see the **history** command in the following:

- “Korn shell or POSIX shell command history” on page 445
- “C shell built-in commands” on page 500
- “History substitution in the C shell” on page 508

Note: The Bourne shell does not support command history.

- **File name substitution**

Automatically produces a list of file names on a command line using pattern-matching characters.

For more information, see:

- “File name substitution in the Korn shell or POSIX shell” on page 456
- “File name substitution in the Bourne shell” on page 497
- “File name substitution in the C shell” on page 513

- **Input and output redirection**

Redirects input away from the keyboard and redirects output to a file or device other than the terminal. For example, input to a program can be provided from a file and redirected to the printer or to another file.

For more information, see:

- “Input and output redirection in the Korn shell or POSIX shell” on page 457
- “Input and output redirection in the Bourne shell” on page 498
- “Input and output redirection in the C shell” on page 517

- **Piping**

Links any number of commands together to form a complex program. The standard output of one program becomes the standard input of the next.

For more information, see the *pipeline* definition in “Shells terminology” on page 437.

- **Shell variable substitution**

Stores data in user-defined variables and predefined shell variables.

For more information, see:

- “Parameter substitution in the Korn shell or POSIX shell” on page 449
- “Variable substitution in the Bourne shell” on page 492
- “Variable substitution in the C shell” on page 512

Available shells

The following are the shells that are provided with AIX.

- Korn shell (started with the **ksh** command)
- Bourne shell (started with the **bsh** command)
- Restricted shell (a limited version of the Bourne shell, and started with the **Rsh** command)
- POSIX shell (also known as the Korn Shell, and started with the **psh** command)

- Default shell (started with the **sh** command)
- C shell (started with the **cs** command)
- Trusted shell (a limited version of the Korn shell, and started with the **tsh** command)
- Remote shell (started with the **rsh** command)

The *login shell* refers to the shell that is loaded when you log in to the computer system. Your login shell is set in the **/etc/passwd** file. The *Korn shell* is the standard operating system login shell and is backward-compatible with the Bourne Shell (see “Bourne shell” on page 483).

The *default* or *standard shell* refers to the shell linked to and started with the **/usr/bin/sh** command. The Bourne shell is set up as the default shell and is a subset of the Korn shell.

The **/usr/bin/sh** command resides as a copy of the Korn shell, which is **/usr/bin/ksh**. Therefore, the Korn shell can be substituted as the default shell. The POSIX shell, which is invoked by the **/usr/bin/psh** command, resides as a link to the **/usr/bin/sh** command.

Shells terminology

The terms and definitions in this table are helpful in understanding shells.

blank	A blank is one of the characters in the blank character class defined in the LC_CTYPE category. In the POSIX shell, a blank is either a tab or space.
built-in command	A command that the shell executes without searching for it and creating a separate process.
command	A sequence of characters in the syntax of the shell language. The shell reads each command and carries out the desired action either directly or by invoking separate utilities.
comment	Any word that begins with pound sign (#). The word and all characters that follow it, until the next newline character, are ignored.
identifier	A sequence of letters, digits, or underscores from the portable character set, starting with a letter or underscore. The first character of an identifier must not be a digit. Identifiers are used as names for aliases, functions, and named parameters.

list	<p>A sequence of one or more pipelines separated by one of the following symbols: semicolon (;), ampersand (&), double ampersand (&&), or double bar (). The list is optionally ended by one of the following symbols: semicolon (;), ampersand (&), or bar ampersand (&).</p> <ul style="list-style-type: none"> ; Sequentially processes the preceding pipeline. The shell carries out each command in turn and waits for the most recent command to complete. & Asynchronously processes the preceding pipeline. The shell carries out each command in turn, processing the pipeline in the background without waiting for it to complete. & Asynchronously processes the preceding pipeline and establishes a two-way pipe to the parent shell. The shell carries out each command in turn, processing the pipeline in the background without waiting for it to complete. The parent shell can read from and write to the standard input and output of the spawned command by using the read -p and print -p commands. Only one such command can be active at any given time. && Processes the list that follows this symbol only if the preceding pipeline returns an exit value of zero (0). Processes the list that follows this symbol only if the preceding pipeline returns a nonzero exit value. <p>The semicolon (;), ampersand (&), and bar ampersand (&) have a lower priority than the double ampersand (&&) and double bar (). The ;, &, and & symbols have equal priority among themselves. The && and symbols are equal in priority. One or more newline characters can be used instead of a semicolon to delimit two commands in a list.</p> <p>Note: The & symbol is valid only in the Korn shell.</p>
-------------	---

metacharacter	<p>Each metacharacter has a special meaning to the shell and causes termination of a word unless it is quoted. Metacharacters are: pipe (), ampersand (&), semicolon (;), less-than sign (<), greater-than sign (>), left parenthesis ((), right parenthesis ()), dollar sign (\$), backquote (`), backslash (\), right quote ('), double quotation marks ("), newline character, space character, and tab character. All characters enclosed between single quotation marks are considered quoted and are interpreted literally by the shell. The special meaning of metacharacters is retained if not quoted. (Metacharacters are also known as <i>parser metacharacters</i> in the C shell.)</p>
----------------------	--

parameter assignment list	<p>Includes one or more words of the form <i>Identifier=Value</i> in which spaces surrounding the equal sign (=) must be balanced. That is, leading and trailing blanks, or no blanks, must be used.</p> <p>Note: In the C shell, the parameter assignment list is of the form set Identifier=Value. The spaces surrounding the equal sign (=) are required.</p>
----------------------------------	--

pipeline	<p>A sequence of one or more commands separated by pipe (). Each command in the pipeline, except possibly the last command, is run as a separate process. However, the standard output of each command that is connected by a pipe becomes the standard input of the next command in the sequence. If a list is enclosed with parentheses, it is carried out as a simple command that operates in a separate subshell.</p> <p>If the reserved word ! does not precede the pipeline, the exit status will be the exit status of the last command specified in the pipeline. Otherwise, the exit status is the logical NOT of the exit status of the last command. In other words, if the last command returns zero, the exit status will be 1. If the last command returns greater than zero, the exit status will be zero.</p> <p>The format for a pipeline is as follows:</p> <pre>[!] command1 [command2 ...]</pre> <p>Note: Early versions of the Bourne shell used the caret (^) to indicate a pipe.</p>
-----------------	---

shell variable	A name or parameter to which a value is assigned. Assign a variable by typing the variable name, an equal sign (=), and then the value. The variable name can be substituted for the assigned value by preceding the variable name with a dollar sign (\$). Variables are particularly useful for creating a short notation for a long path name, such as \$HOME for the home directory. A predefined variable is one whose value is assigned by the shell. A user-defined variable is one whose value is assigned by a user.
simple command	A sequence of optional parameter assignment lists and redirections, in any sequence. They are optionally followed by commands, words, and redirections. They are terminated by ;, , &, , &&, &, or a newline character. The command name is passed as parameter 0 (as defined by the exec subroutine). The value of a simple command is its exit status of zero if it terminates normally or nonzero if it terminates abnormally. The sigaction, sigvec, or signal subroutine includes a list of signal-exit status values.
subshell	A shell that is running as a child of the login shell or the current shell.
wildcard character	Also known as a <i>pattern-matching character</i> . The shell associates them with assigned values. The basic wildcards are ?, *, [set], and [!set]. Wildcard characters are particularly useful when performing file name substitution.
word	A sequence of characters that does not contain any blanks. Words are separated by one or more metacharacters.

Creating and running a shell script

A *shell script* is a file that contains one or more commands. Shell scripts provide an easy way to carry out tedious commands, large or complicated sequences of commands, and routine tasks. When you enter the name of a shell script file, the system executes the command sequence contained by the file.

You can create a shell script by using a text editor. Your script can contain both operating system commands and shell built-in commands.

The following steps are general guidelines for writing shell scripts:

1. Using a text editor, create and save a file. You can include any combination of shell and operating system commands in the shell script file. By convention, shell scripts that are not set up for use by many users are stored in the **\$HOME/bin** directory.

Note: The operating system does not support the **setuid** or **setgid** subroutines within a shell script.

2. Use the **chmod** command to allow only the owner to run (or execute) the file. For example, if your file is named **script1**, type the following:

```
chmod u=rwx script1
```

3. Type the script name on the command line to run the shell script. To run the **script1** shell script, type the following:

```
script1
```

Note: You can run a shell script without making it executable if a shell command (**ksh**, **bsh**, or **cs**) precedes the shell script file name on the command line. For example, to run a nonexecutable file named **script1** under the control of the Korn shell, type the following:

```
ksh script1
```

Specifying a shell for a script file

When you run an executable shell script in either the Korn (the POSIX Shell) or Bourne shell, the commands in the script are carried out under the control of the current shell (the shell from which the script is started) unless you specify a different shell. When you run an executable shell script in the C shell, the commands in the script are carried out under the control of the Bourne shell (**/usr/bin/bsh**) unless you specify a different shell.

You can run a shell script in a specific shell by including the shell within the shell script.

To run an executable shell script under a specific shell, type `#!Path` on the first line of the shell script, and press Enter. The `#!` characters identify the file type. The *Path* variable specifies the path name of the shell from which to run the shell script.

For example, to run the **bsh** script in the Bourne shell, type the following:

```
#!/usr/bin/bsh
```

When you precede a shell script file name with a shell command, the shell specified on the command line overrides any shell specified within the script file itself. Therefore, typing `ksh myfile` and pressing Enter runs the file named **myfile** under the control of the Korn shell, even if the first line of **myfile** is `#!/usr/bin/csh`.

Korn shell or POSIX shell commands

The Korn shell is an interactive command interpreter and command programming language. It conforms to the Portable Operating System Interface for Computer Environments (POSIX), an international standard for operating systems.

POSIX is not an operating system, but is a *standard* aimed at portability of applications, at the source level, across many systems. POSIX features are built on top of the Korn shell. The Korn shell (also known as the POSIX shell) offers many of the same features as the Bourne and C shells, such as I/O redirection capabilities, variable substitution, and file name substitution. It also includes several additional command and programming language features:

Note: There is a restricted version of Korn shell available, called **rksh**. For more details, refer to the **rksh** command.

Arithmetic evaluation The Korn shell, or POSIX shell, can perform integer arithmetic using the built-in **let** command, using any base from 2 to 36. See “Arithmetic evaluation in the Korn shell or POSIX shell” on page 454 for more information about this feature.

In order to enable recognition of numbers starting with 0 (Octal) and 0x (Hexadecimal) in the Korn shell, run the following commands:

```
export XPG_SUS_ENV=ON
```

Exporting the `XPG_SUS_ENV` variable causes the commands that are run and the libraries that they use to be completely POSIX-compliant.

Note: Because the entire library system becomes POSIX-compliant, a given command’s default expected behavior might change.

```
export OCTAL_CONST=ON
```

Exporting this variable causes the interpretation of constants declared in the Korn shell to be POSIX-compliant as far as the recognition of octal and hexadecimal constants is concerned.

Command history The Korn shell, or POSIX shell, stores a file that records all of the commands you enter. You can use a text editor to alter a command in this history file and then reissue the command. For more information about the command history feature, see “Korn shell or POSIX shell command history” on page 445.

Coprocess facility Enables you to run programs in the background and send and receive information to these background processes. For more information, see “Coprocess facility” on page 459.

Editing The Korn shell, or POSIX shell, offers inline editing options that enable you to edit the command line. Editors similar to emacs, gmacs, and vi are available. See “Inline editing in the Korn shell or POSIX shell” on page 473 for more information about this feature.

A Korn shell command is one of the following:

- Simple command

- Pipeline
- List
- Compound command
- Function

When you issue a command in the Korn shell or POSIX shell, the shell evaluates the command and does the following:

- Makes all indicated substitutions.
- Determines whether the command contains a slash (/). If it does, the shell runs the program named by the specified path name.

If the command does not contain a slash (/), the Korn shell or POSIX shell continues with the following actions:

- Determines whether the command is a special built-in command. If it is, the shell runs the command within the current shell process.
For information about special built-in commands, see “Korn shell or POSIX shell built-in commands” on page 460.
- Compares the command to user-defined functions. If the command matches a user-defined function, then the positional parameters are saved and then reset to the arguments of the *function* call. When the function completes or issues a return, the positional parameter list is restored, and any trap set on EXIT within the function is carried out. The value of a function is the value of the last command executed. A function is carried out in the current shell process.
- If the command name matches the name of a regular built-in command, then that regular built-in command will be invoked.
For information about regular built-in commands, see “Korn shell or POSIX shell built-in commands” on page 460.
- Creates a process and attempts to carry out the command by using the **exec** command (if the command is neither a built-in command nor a user-defined function).

The Korn shell, or POSIX shell, searches each directory in a specified path for an executable file. The *PATH* shell variable defines the search path for the directory containing the command. Alternative directory names are separated with a colon (:). The default path is **/usr/bin:** (specifying the **/usr/bin** directory, and the current directory, in that order). The current directory is specified by two or more adjacent colons, or by a colon at the beginning or end of the path list.

If the file has execute permission but is not a directory or an **a.out** file, the shell assumes that it contains shell commands. The current shell process spawns a subshell to read the file. All nonexported aliases, functions, and named parameters are removed from the file. If the shell command file has *read* permission, or if the **setuid** or **setgid** bits are set on the file, then the shell runs an agent that sets up the permissions and carries out the shell with the shell command file passed down as an open file. A parenthesized command is run in a subshell without removing nonexported quantities.

Korn shell compound commands

A compound command can be a list of simple commands or a pipeline, or it can begin with a reserved word. Most of the time, you will use compound commands such as **if**, **while**, and **for** when you are writing shell scripts.

The following is a list of list of Korn shell or POSIX shell compound commands:

Command syntax	Description
for <i>Identifier</i> [in <i>Word</i> ...] ; do <i>List</i> ; done	Each time a for command is executed, the Identifier parameter is set to the next word taken from the in <i>Word</i> ... list. If the in <i>Word</i> ... command is omitted, then the for command executes the do <i>List</i> command once for each positional parameter that is set. Execution ends when there are no more words in the list. For more information on positional parameters, see "Parameters in the Korn shell" on page 449.
select <i>Identifier</i> [in <i>Word</i> ...] ; do <i>List</i> ; done	A select command prints on standard error (file descriptor 2) the set of words specified, each preceded by a number. If the in <i>Word</i> ... command is omitted, then the positional parameters are used instead. The PS3 prompt is printed and a line is read from the standard input. If this line consists of the number of one of the listed words, then the value of the <i>Identifier</i> parameter is set to the word corresponding to this number. If the line read from standard input is empty, the selection list is printed again. Otherwise, the value of the <i>Identifier</i> parameter is set to null. The contents of the line read from standard input is saved in the REPLY parameter. The <i>List</i> parameter is executed for each selection until a break or an end-of-file character is encountered. For more information on positional parameters, see "Parameters in the Korn shell" on page 449.
case <i>Word</i> in [(<i>Pattern</i> [<i>Pattern</i>] ...) <i>List</i> ;;] ... esac	A case command executes the <i>List</i> parameter associated with the first <i>Pattern</i> parameter that matches the <i>Word</i> parameter. The form of the patterns is the same as that used for file name substitution.
if <i>List</i> ; then <i>List</i> [elif <i>List</i> ; then <i>List</i>] ... [else <i>List</i>] ; fi	The <i>List</i> parameter specifies a list of commands to be run. The shell executes the if <i>List</i> command first. If a zero exit status is returned, it executes the then <i>List</i> command. Otherwise, the commands specified by the <i>List</i> parameter following the elif command are executed. If the value returned by the last command in the elif <i>List</i> command is zero, the then <i>List</i> command is executed. If the value returned by the last command in the then <i>List</i> command is zero, the else <i>List</i> command is executed. If no commands specified by the <i>List</i> parameters are executed for the else or then command, the if command returns a zero exit status.
while <i>List</i> ; do <i>List</i> ; done until <i>List</i> ; do <i>List</i> ; done	The <i>List</i> parameter specifies a list of commands to be run. The while command repeatedly executes the commands specified by the <i>List</i> parameter. If the exit status of the last command in the while <i>List</i> command is zero, the do <i>List</i> command is executed. If the exit status of the last command in the while <i>List</i> command is not zero, the loop terminates. If no commands in the do <i>List</i> command are executed, then the while command returns a zero exit status. The until command might be used in place of the while command to negate the loop termination test.
(<i>List</i>)	The <i>List</i> parameter specifies a list of commands to run. The shell executes the <i>List</i> parameter in a separate environment. Note: If two adjacent open parentheses are needed for nesting, you must insert a space between them to differentiate between the command and arithmetic evaluation.
{ <i>List</i> ; }	The <i>List</i> parameter specifies a list of commands to run. The <i>List</i> parameter is simply executed. Note: Unlike the metacharacters (), { } denote reserved words (used for special purposes, not as user-declared identifiers). To be recognized, these reserved words must appear at the beginning of a line or after a semicolon (;).
[[<i>Expression</i>]]	Evaluates the <i>Expression</i> parameter. If the expression is true, then the command returns a zero exit status.

Command syntax	Description
function <i>Identifier</i> { <i>List</i> ;} or function <i>Identifier</i> () { <i>List</i> ;}	Defines a function that is referred to by the <i>Identifier</i> parameter. The body of the function is the specified list of commands enclosed by { }. The () consists of two operators, so mixing blank characters with the <i>identifier</i> , (and) is permitted, but is not necessary.
time <i>Pipeline</i>	Executes the <i>Pipeline</i> parameter. The elapsed time, user time, and system time are printed to standard error.

Shell startup

You can start the Korn shell with the **ksh** command, **psh** command (POSIX shell), or the **exec** command.

If the shell is started by the **exec** command, and the first character of zero argument (**\$0**) is the hyphen (-), then the shell is assumed to be a login shell. The shell first reads commands from the **/etc/profile** file and then from either the **.profile** file in the current directory or from the **\$HOME/.profile** file, if either file exists. Next, the shell reads commands from the file named by performing parameter substitution on the value of the **ENV** environment variable, if the file exists.

If you specify the *File* [*Parameter*] parameter when invoking the Korn shell or POSIX shell, the shell runs the script file identified by the *File* parameter, including any parameters specified. The script file specified must have read permission; any **setuid** and **setgid** settings are ignored. The shell then reads the commands.

Note: Do not specify a script file with the **-c** or **-s** flags when invoking the Korn shell or POSIX shell.

For more information on positional parameters, see “Parameters in the Korn shell” on page 449.

Korn shell environment

All variables (with their associated values) known to a command at the beginning of its execution constitute its *environment*.

This environment includes variables that a command inherits from its parent process and variables specified as keyword parameters on the command line that calls the command. The shell interacts with the environment in several ways. When it is started, the shell scans the environment and creates a parameter for each name found, giving the parameter the corresponding value and marking it for export. Executed commands inherit the environment.

If you modify the values of the shell parameters or create new ones using the **export** or **typeset -x** commands, the parameters become part of the environment. The environment seen by any executed command is therefore composed of any name-value pairs originally inherited by the shell, whose values might be modified by the current shell, plus any additions that resulted from using the **export** or **typeset -x** commands. The executed command (subshell) will see any modifications it makes to the environment variables it has inherited, but for its child shells or processes to see the modified values, the subshell must export these variables.

The environment for any simple command or function is changed by prefixing with one or more parameter assignments. A parameter assignment argument is a word of the form *Identifier=Value*. Thus, the two following expressions are equivalent (as far as the execution of the command is concerned):

```
TERM=450 Command arguments
```

```
(export TERM; TERM=450; Command arguments)
```

Korn shell functions

The **function** reserved word defines shell functions. The shell reads and stores functions internally. Alias names are resolved when the function is read. The shell executes functions in the same manner as commands, with the arguments passed as positional parameters.

For more information on positional parameters, see “Parameters in the Korn shell” on page 449.

The Korn shell or POSIX shell executes functions in the environment from which functions are invoked. All of the following are shared by the function and the invoking script, and side effects can be produced:

- Variable values and attributes (unless you use **typeset** command within the function to declare a local variable)
- Working directory
- Aliases, function definitions, and attributes
- Special parameter \$
- Open files

The following are not shared between the function and the invoking script, and there are no side effects:

- Positional parameters
- Special parameter #
- Variables in a variable assignment list when the function is invoked
- Variables declared using **typeset** command within the function
- Options
- Traps. However, signals ignored by the invoking script will also be ignored by the function.

Note: In earlier versions of the Korn shell, traps other than **EXIT** and **ERR** were shared by the function as well as the invoking script.

If trap on **0** or **EXIT** is executed *inside* the body of a function, then the action is executed after the function completes, in the environment that called the function. If the trap is executed *outside* the body of a function, then the action is executed upon exit from the Korn shell. In earlier versions of the Korn shell, no trap on **0** or **EXIT** outside the body of a function was executed upon exit from the function.

When a function is executed, it has the same syntax-error and variable-assignment properties described in “Korn shell or POSIX shell built-in commands” on page 460.

The compound command is executed whenever the function name is specified as the name of a simple command. The operands to the command temporarily will become the positional parameters during the execution of the compound command. The special parameter # will also change to reflect the number of operands. The special parameter 0 will not change.

The **return** special command is used to return from function calls. Errors within functions return control to the caller.

Function identifiers are listed with the **-f** or **+f** option of the **typeset** special command. The **-f** option also lists the text of functions. Functions are undefined with the **-f** option of the **unset** special command.

Ordinarily, functions are unset when the shell executes a shell script. The **-xf** option of the **typeset** special command allows a function to be exported to scripts that are executed without a separate invocation of the shell. Functions that must be defined across separate invocations of the shell should be specified in the **ENV** file with the **-xf** option of the **typeset** special command.

The exit status of a function definition is zero if the function was not successfully declared. Otherwise, it will be greater than zero. The exit status of a function invocation is the exit status of the most recent command executed by the function.

Korn shell or POSIX shell command history

The Korn shell or POSIX shell saves commands entered from your terminal device to a history file.

If set, the *HISTFILE* variable value is the name of the history file. If the *HISTFILE* variable is not set or cannot be written, the history file used is **\$HOME/.sh_history**. If the history file does not exist and the Korn shell cannot create it, or if it does exist and the Korn shell does not have permission to append to it, then the Korn shell uses a temporary file as the history file. The shell accesses the commands of all interactive shells using the same named history file with appropriate permissions.

By default, the Korn shell or POSIX shell saves the text of the last 128 commands for nonroot users and 512 commands for the root user. The history file size (specified by the *HISTSIZE* variable) is not limited, although a very large history file can cause the Korn shell to start slowly.

Command history substitution

Use the **fc** built-in command to list or edit portions of the history file. To select a portion of the file to edit or list, specify the number or the first character or characters of the command.

You can specify a single command or range of commands.

If you do not specify an editor program as an argument to the **fc** regular built-in command, the editor specified by the *FCEDIT* variable is used. If the *FCEDIT* variable is not defined, then the **/usr/bin/ed** file is used. The edited command or commands are printed and run when you exit the editor.

The editor name hyphen (-) is used to skip the editing phase and run the command again. In this case, a substitution parameter of the form *01d=New* can be used to modify the command before it is run. For example, if *r* is aliased to **fc -e -**, then typing *r bad=good c* runs the most recent command that starts with the letter *c* and replaces the first occurrence of the *bad* string with the *good* string.

For more information about using the history shell command, see “Listing previously entered commands (history command)” on page 357 and the **fc** command in the *AIX 5L Version 5.3 Commands Reference, Volume 2*.

Quotation of characters in the Korn shell or POSIX shell

When you want the Korn shell or POSIX shell to read a character as a regular character, rather than with any normally associated meaning, you must *quote* it.

Each metacharacter has a special meaning to the shell and, unless quoted, causes termination of a word. The following characters are considered metacharacters by the Korn shell or POSIX shell and must be quoted if they are to represent themselves:

- pipe (|)
- ampersand (&)
- semicolon (;)
- less-than sign (<) and greater-than sign (>)
- left parenthesis (()) and right parenthesis ())
- dollar sign (\$)
- backquote (`) and single quotation mark (')
- backslash (\)
- double-quotation marks (")

- newline character
- space character
- tab character

To negate the special meaning of a metacharacter, use one of the quoting mechanisms in the following list.

Backslash	A backslash (\) that is not quoted preserves the literal value of the following character, with the exception of a newline character. If a newline character follows the backslash, then the shell interprets this as line continuation.
Single Quotation Marks	<p>Enclosing characters in single quotation marks (' ') preserves the literal value of each character within the single quotation marks. A single quotation mark cannot occur within single quotation marks.</p> <p>A backslash cannot be used to escape a single quotation mark in a string that is set in single quotation marks. An embedded quotation mark can be created by writing, for example: 'a\'`b', which yields a'b.</p>
Double Quotation Marks	<p>Enclosing characters in double quotation marks (" ") preserves the literal value of all characters within the double quotation marks, with the exception of the dollar sign, backquote, and backslash characters, as follows:</p> <p>\$ The dollar sign retains its special meaning introducing parameter expansion, a form of command substitution, and arithmetic expansion.</p> <p>The input characters within the quoted string that are also enclosed between \$(and the matching) will not be affected by the double quotation marks, but define that command whose output replaces the \$(...) when the word is expanded.</p> <p>Within the string of characters from an enclosed \${ to the matching }, there must be an even number of unescaped double quotation marks or single quotation marks, if any. A preceding backslash character must be used to escape a literal { or }.</p> <p>` The backquote retains its special meaning introducing the other form of command substitution. The portion of the quoted string, from the initial backquote and the characters up to the next backquote that is not preceded by a backslash, defines that command whose output replaces `... ` when the word is expanded.</p> <p>\ The backslash retains its special meaning as an escape character only when followed by one of the following characters: \$, `, ", \, or a newline character.</p>

A double quotation mark must be preceded by a backslash to be included within double quotation marks. When you use double quotation marks, if a backslash is immediately followed by a character that would be interpreted as having a special meaning, the backslash is deleted, and the subsequent character is taken literally. If a backslash does not precede a character that would have a special meaning, it is left in place unchanged, and the character immediately following it is also left unchanged. For example:

```
"\$"   ->  $
"\a"   ->  \a
```

The following conditions apply to metacharacters and quoting characters in the Korn or POSIX shell:

- The meanings of dollar sign, asterisk (\$*) and dollar sign, at symbol (\$@) are identical when not quoted, when used as a parameter assignment value, or when used as a file name.

- When used as a command argument, double quotation marks, dollar sign, asterisk, double quotation marks ("*\$") is equivalent to "\$1\$d\$2d...", where *d* is the first character of the IFS parameter.
- Double quotation marks, at symbol, asterisk, double quotation marks ("*\$@") are equivalent to "\$1" "\$2"
- Inside backquotes (`~), the backslash quotes the characters backslash (\), single quotation mark ('), and dollar sign (\$). If the backquotes occur within double quotation marks (" "), the backslash also quotes the double quotation marks character.
- Parameter and command substitution occurs inside double quotation marks (" ").
- The special meaning of reserved words or aliases is removed by quoting any character of the reserved word. You cannot quote function names or built-in command names.

Reserved words in the Korn shell or POSIX shell

The following reserved words have special meaning to the Korn shell or POSIX shell.

```
!      case   do
done   elif   else
esac   fi     for
function if    in
select then   time
until  while  {
}      [[    ]]
```

The reserved words are recognized only when they appear without quotation marks and when the word is used as the following:

- First word of a command
- First word following one of the reserved words other than **case**, **for**, or **in**
- Third word in a **case** or **for** command (only **in** is valid in this case)

Command aliasing in the Korn shell or POSIX shell

The Korn shell, or POSIX shell, allows you to create aliases to customize commands.

The **alias** command defines a word of the form `Name=String` as an alias. When you use an alias as the first word of a command line, the Korn shell checks to see if it is already processing an alias with the same name. If it is, the Korn shell does not replace the alias name. If an alias with the same name is not already being processed, the Korn shell replaces the alias name by the value of the alias.

The first character of an alias name can be any printable character except the metacharacters. The remaining characters must be the same as for a valid identifier. The replacement string can contain any valid shell text, including the metacharacters.

If the last character of the alias value is a blank, the shell also checks the word following the alias for alias substitution. You can use aliases to redefine special built-in commands but not to redefine reserved words. Alias definitions are not inherited across invocations of **ksh**. However, if you specify **alias -x**, the alias stays in effect for scripts invoked by name that do not invoke a separate shell. To export an alias definition and to cause child processes to have access to them, you must specify **alias -x** and the alias definition in your environment file.

Use the **alias** command to create, list, and export aliases.

Use the **unalias** command to remove aliases.

The format for creating an alias is as follows:

```
alias Name=String
```

where the **Name** parameter specifies the name of the alias, and the **String** parameter specifies the value of the alias.

The following exported aliases are predefined by the Korn shell but can be unset or redefined. It is not recommended that you change them, because this might later confuse anyone who expects the alias to work as predefined by the Korn shell.

```
autoload='typeset -fu'  
false='let 0'  
functions='typeset -f'  
hash='alias -t'  
history='fc -l'  
integer='typeset -i'  
nohup='nohup '  
r='fc -e -'  
true=':'  
type='whence -v'
```

Aliases are not supported on noninteractive invocations of the Korn shell (**ksh**); for example, in a shell script, or with the **-c** option in **ksh**, as in the following:

```
ksh -c alias
```

For more information about aliasing, see “Creating a command alias (alias shell command)” on page 360 and the **alias** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1*.

Tracked aliases

Frequently, aliases are used as shorthand for full path names. One aliasing facility option allows you to automatically set the value of an alias to the full path name of a corresponding command. This special type of alias is a *tracked* alias.

Tracked aliases speed execution by eliminating the need for the shell to search the *PATH* variable for a full path name.

The **set -h** command turns on command *tracking* so that each time a command is referenced, the shell defines the value of a tracked alias. This value is undefined each time you reset the *PATH* variable.

These aliases remain tracked so that the next subsequent reference will redefine the value. Several tracked aliases are compiled into the shell.

Tilde substitution

After the shell performs alias substitution, it checks each word to see if it begins with an unquoted tilde (~). If it does, the shell checks the word, up to the first slash (/), to see if it matches a user name in the */etc/passwd* file. If the shell finds a match, it replaces the ~ character and the name with the login directory of the matched user. This process is called *tilde substitution*.

The shell does not change the original text if it does not find a match. The Korn shell also makes special replacements if the ~ character is the only character in the word or followed by plus sign (+) or hyphen (-):

~	Replaced by the value of the <i>HOME</i> variable
~+	Replaced by the <i>\$PWD</i> variable (the full path name of the current directory)
~-	Replaced by the <i>\$OLDPWD</i> variable (the full path name of the previous directory)

In addition, the shell attempts tilde substitution when the value of a variable assignment parameter begins with a tilde ~ character.

Parameters in the Korn shell

Korn shell parameters are discussed below.

A parameter is defined as the following:

- Identifier of any of the characters asterisk (*), at sign (@), pound sign (#), question mark (?), hyphen (-), dollar sign (\$), and exclamation point (!). These are called *special parameters*.
- Argument denoted by a number (*positional parameter*)
- Parameter denoted by an identifier, with a value and zero or more attributes (*named parameter/variables*).

The **typeset** special built-in command assigns values and attributes to named parameters. The attributes supported by the Korn shell are described with the **typeset** special built-in command. Exported parameters pass values and attributes to the environment.

The value of a named parameter is assigned by:

```
Name=Value [ Name=Value ] ...
```

If the **-i** integer attribute is set for the **Name** parameter, then the **Value** parameter is subject to arithmetic evaluation. See “Arithmetic evaluation in the Korn shell or POSIX shell” on page 454 for more information about arithmetic expression evaluation.

The shell supports a one-dimensional array facility. An element of an array parameter is referenced by a subscript. A subscript is denoted by an arithmetic expression enclosed by brackets []. To assign values to an array, use set **-A Name Value ...**. The value of all subscripts must be in the range of 0 through 511. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array will be created, if necessary. Referencing an array without a subscript is equivalent to referencing the element 0.

Positional parameters are assigned values with the **set** special command. The **\$0** parameter is set from argument 0 when the shell is invoked. The \$ character is used to introduce parameters that can be substituted.

Parameter substitution in the Korn shell or POSIX shell

The Korn shell, or POSIX shell, lets you perform parameter substitutions.

The following are substitutable parameters:

`${Parameter}`

The shell reads all the characters from the `${` to the matching `}` as part of the same word, even if that word contains braces or metacharacters. The value, if any, of the specified parameter is substituted. The braces are required when the *Parameter* parameter is followed by a letter, digit, or underscore that is not to be interpreted as part of its name, or when a named parameter is subscripted.

If the specified parameter contains one or more digits, it is a *positional parameter*. A positional parameter of more than one digit must be enclosed in braces. If the value of the variable is * or @, each positional parameter, starting with \$1, is substituted (separated by a field separator character). If an array identifier with a subscript * or @ is used, then the value for each of the elements (separated by a field separator character) is substituted.

`${#Parameter}`

If the value of the *Parameter* parameter is * or @, the number of positional parameters is substituted. Otherwise, the length specified by the *Parameter* parameter is substituted.

<code>#{Identifier[*]}</code>	The number of elements in the array specified by the <i>Identifier</i> parameter is substituted.
<code>\${Parameter:-Word}</code>	If the <i>Parameter</i> parameter is set and is not null, then its value is substituted; otherwise, the value of the <i>Word</i> parameter is substituted.
<code>\${Parameter:=Word}</code>	If the <i>Parameter</i> parameter is not set or is null, then it is set to the value of the <i>Word</i> parameter. Positional parameters cannot be assigned in this way.
<code>\${Parameter:?Word}</code>	If the <i>Parameter</i> parameter is set and is not null, then substitute its value. Otherwise, print the value of the <i>Word</i> variable and exit from the shell. If the <i>Word</i> variable is omitted, then a standard message is printed.
<code>\${Parameter:+Word}</code>	If the <i>Parameter</i> parameter is set and is not null, then substitute the value of the <i>Word</i> variable.
<code>\${Parameter#Pattern} \${Parameter##Pattern}</code>	If the specified shell <i>Pattern</i> parameter matches the beginning of the value of the <i>Parameter</i> parameter, then the value of this substitution is the value of the <i>Parameter</i> parameter with the matched portion deleted. Otherwise, the value of the <i>Parameter</i> parameter is substituted. In the first form, the smallest matching pattern is deleted. In the second form, the largest matching pattern is deleted.
<code>\${Parameter%Pattern} \${Parameter%%Pattern}</code>	If the specified shell <i>Pattern</i> matches the end of the value of the <i>Parameter</i> variable, then the value of this substitution is the value of the <i>Parameter</i> variable with the matched part deleted. Otherwise, substitute the value of the <i>Parameter</i> variable. In the first form, the smallest matching pattern is deleted; in the second form, the largest matching pattern is deleted. In the previous expressions, the <i>Word</i> variable is not evaluated unless it is to be used as the substituted string. Thus, in the following example, the pwd command is executed only if the -d flag is not set or is null: <code>echo \${d:-\$(pwd)}</code>

Note: If the `:` is omitted from the previous expressions, the shell checks only whether the *Parameter* parameter is set.

Predefined special parameters in the Korn shell or POSIX shell

Some parameters are set automatically by the Korn shell or POSIX shell.

The following parameters are automatically set by the shell:

<code>@</code>	Expands the positional parameters, beginning with \$1. Each parameter is separated by a space. If you place " around \$@, the shell considers each positional parameter a separate string. If no positional parameters exist, the shell expands the statement to an unquoted null string.
<code>*</code>	Expands the positional parameters, beginning with \$1. The shell separates each parameter with the first character of the IFS parameter value. If you place " around \$*, the shell includes the positional parameter values in double quotation marks. Each value is separated by the first character of the IFS parameter.
<code>#</code>	Specifies the number (in decimals) of positional parameters passed to the shell, not counting the name of the shell procedure itself. The \$# parameter thus yields the number of the highest-numbered positional parameter that is set. One of the primary uses of this parameter is to check for the presence of the required number of arguments.
<code>-</code>	Supplies flags to the shell on invocation or with the set command.

?	Specifies the exit value of the last command executed. Its value is a decimal string. Most commands return 0 to indicate successful completion. The shell itself returns the current value of the \$? parameter as its exit value.
\$	Identifies the process number of this shell. Because process numbers are unique among all existing processes, this string of up to 5 digits is often used to generate unique names for temporary files. The following example illustrates the recommended practice of creating temporary files in a directory used only for that purpose: <pre>temp=\$HOME/temp/\$\$ ls >\$temp . . . rm \$temp</pre>
!	Specifies the process number of the most recent background command invoked.
zero (0)	Expands to the name of the shell or shell script.

Variables set by the Korn shell or POSIX shell

The following are variables that are set by the shell.

<i>underscore (_)</i>	Indicates initially the absolute path name of the shell or script being executed as passed in the environment. Subsequently, it is assigned the last argument of the previous command. This parameter is not set for commands that are asynchronous. This parameter is also used to hold the name of the matching MAIL file when checking for mail.
<i>ERRNO</i>	Specifies a value that is set by the most recently failed subroutine. This value is system-dependent and is intended for debugging purposes.
<i>LINENO</i>	Specifies the line number of the current line within the script or function being executed.
<i>OLDPWD</i>	Indicates the previous working directory set by the cd command.
<i>OPTARG</i>	Specifies the value of the last option argument processed by the getopts regular built-in command.
<i>OPTIND</i>	Specifies index of the last option argument processed by the getopts regular built-in command.
<i>PPID</i>	Identifies the process number of the parent of the shell.
<i>PWD</i>	Indicates the present working directory set by the cd command.
<i>RANDOM</i>	Generates a random integer, uniformly distributed between 0 and 32767. The sequence of random numbers can be initialized by assigning a numeric value to the <i>RANDOM</i> variable.
<i>REPLY</i>	Set by the select statement and by the read regular built-in command when no arguments are supplied.
<i>SECONDS</i>	Specifies the number of seconds since shell invocation is returned. If this variable is assigned a value, then the value returned upon reference will be the value that was assigned plus the number of seconds since the assignment.

Variables used by the Korn shell or POSIX shell

The following are variables that are used by the shell.

CDPATH	Indicates the search path for the cd (change directory) command.
COLUMNS	Defines the width of the edit window for the shell edit modes and for printing select lists.
EDITOR	If the value of this parameter ends in <i>emacs</i> , <i>gmacs</i> , or <i>vi</i> , and the <i>VISUAL</i> variable is not set with the set special built-in command, then the corresponding option is turned on.
ENV	If this variable is set, then parameter substitution is performed on the value to generate the path name of the script that will be executed when the shell is invoked. This file is typically used for alias and function definitions. This variable will be ignored for noninteractive shells.
FCEDIT	Specifies the default editor name for the fc regular built-in command.
FPATH	Specifies the search path for function definitions. This path is searched when a function with the -u flag is referenced and when a command is not found. If an executable file is found, then it is read and executed in the current environment.
HISTFILE	<p>If this variable is set when the shell is invoked, then the value is the path name of the file that will be used to store the command history.</p> <p>The initialization process for the history file can be dependent on the system start-up files because some start-up files can contain commands that effectively preempt the settings the user has specified for <i>HISTFILE</i> and <i>HISTSIZ</i>E. For example, function definition commands are recorded in the history file. If the system administrator includes function definitions in a system start-up file that is called before the <i>ENV</i> file or before <i>HISTFILE</i> or <i>HISTSIZ</i>E variable is set, the history file is initialized before the user can influence its characteristics.</p>
HISTSIZ E	If this variable is set when the shell is invoked, then the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128 commands for nonroot users and 512 commands for the root user.
HOME	Indicates the name of your login directory, which becomes the current directory upon completion of a login. The login program initializes this variable. The cd command uses the value of the <i>\$HOME</i> parameter as its default value. Using this variable rather than an explicit path name in a shell procedure allows the procedure to be run from a different directory without alterations.
IFS	Specifies IFS (internal field separators), which are normally space, tab, and newline, used to separate command words that result from command or parameter substitution and for separating words with the regular built-in command read . The first character of the <i>IFS</i> parameter is used to separate arguments for the <i>\$*</i> substitution.
LANG	Provides a default value for the <i>LC_*</i> variables.
LC_ALL	Overrides the value of the <i>LANG</i> and <i>LC_*</i> variables.
LC_COLLATE	Determines the behavior of range expression within pattern matching.
LC_CTYPE	Defines character classification, case conversion, and other character attributes.
LC_MESSAGES	Determines the language in which messages are written.
LINES	Determines the column length for printing select lists. Select lists print vertically until about two-thirds of lines specified by the <i>LINES</i> variable are filled.
MAIL	Specifies the file path name used by the mail system to detect the arrival of new mail. If this variable is set to the name of a mail file and the <i>MAILPATH</i> variable is not set, then the shell informs the user of new mail in the specified file.
MAILCHECK	Specifies how often (in seconds) the shell checks for changes in the modification time of any of the files specified by the <i>MAILPATH</i> or <i>MAIL</i> variables. The default value is 600 seconds. When the time has elapsed, the shell checks before issuing the next prompt.

MAILPATH	Specifies a list of file names separated by colons. If this variable is set, then the shell informs the user of any modifications to the specified files that have occurred during the period, in seconds, specified by the <i>MAILCHECK</i> variable. Each file name can be followed by a <i>?</i> and a message that will be printed. The message will undergo variable substitution with the <i>\$_</i> variable defined as the name of the file that has changed. The default message is you have mail in <i>\$_</i> .
NLSPATH	Determines the location of message catalogs for the processing of LC_MESSAGES .
PATH	Indicates the search path for commands, which is an ordered list of directory path names separated by colons. The shell searches these directories in the specified order when it looks for commands. A null string anywhere in the list represents the current directory.
PS1	Specifies the string to be used as the primary system prompt. The value of this parameter is expanded for parameter substitution to define the primary prompt string, which is a <i>\$</i> by default. The <i>!</i> character in the primary prompt string is replaced by the command number.
PS2	Specifies the value of the secondary prompt string, which is a <i>></i> by default.
PS3	Specifies the value of the selection prompt string used within a select loop, which is <i>#?</i> by default.
PS4	The value of this variable is expanded for parameter substitution and precedes each line of an execution trace. If omitted, the execution trace prompt is a <i>+</i> .
SHELL	Specifies the path name of the shell, which is kept in the environment.
SHELL PROMPT	When used interactively, the shell prompts with the value of the <i>PS1</i> parameter before reading a command. If at any time a new line is entered and the shell requires further input to complete a command, the shell issues the secondary prompt (the value of the <i>PS2</i> parameter).
TMOUT	Specifies the number of seconds a shell waits inactive before exiting. If the <i>TMOUT</i> variable is set to a value greater than zero (0), the shell exits if a command is not entered within the prescribed number of seconds after issuing the <i>PS1</i> prompt. (Note that the shell can be compiled with a maximum boundary that cannot be exceeded for this value.) Note: After the timeout period has expired, there is a 60-second pause before the shell exits.
VISUAL	If the value of this variable ends in <i>emacs</i> , <i>gmacs</i> , or <i>vi</i> , then the corresponding option is turned on.

The shell gives default values to the *PATH*, *PS1*, *PS2*, *MAILCHECK*, *TMOUT*, and *IFS* parameters, but the *HOME*, *SHELL*, *ENV*, and *MAIL* parameters are *not* set by the shell (although the *HOME* parameter is set by the **login** command).

Command substitution in the Korn shell or POSIX shell

The Korn Shell, or POSIX Shell, lets you perform command substitution. In command substitution, the shell executes a specified command in a subshell environment and replaces that command with its output.

To execute command substitution in the Korn shell or POSIX shell, type the following:

```
$(command)
```

or, for the backquoted version, type the following:

```
`command`
```

Note: Although the backquote syntax is accepted by **ksh**, it is considered obsolete by the X/Open Portability Guide Issue 4 and POSIX standards. These standards recommend that portable applications use the `$(command)` syntax.

The shell expands the command substitution by executing command in a subshell environment and replacing the command substitution (the text of command plus the enclosing `$ ()` or backquotes) with the standard output of the command, removing sequences of one or more newline characters at the end of the substitution.

In the following example, the `$ ()` surrounding the command indicates that the output of the **whoami** command is substituted:

```
echo My name is: $(whoami)
```

You can perform the same command substitution with:

```
echo My name is: `whoami`
```

The output from both examples for user dee is:

```
My name is: dee
```

You can also substitute arithmetic expressions by enclosing them in `()`. For example, the command:

```
echo Each hour contains=$((60 * 60)) seconds
```

produces the following result:

```
Each hour contains 3600 seconds
```

The Korn shell or POSIX shell removes all trailing newline characters when performing command substitution. For example, if your current directory contains the **file1**, **file2**, and **file3** files, the command:

```
echo $(ls)
```

removes the newline characters and produces the following output:

```
file1 file2 file3
```

To preserve newline characters, insert the substituted command in `" "`:

```
echo "$(ls)"
```

Arithmetic evaluation in the Korn shell or POSIX shell

The Korn shell or POSIX shell regular built-in **let** command enables you to perform integer arithmetic.

Constants are of the form **[Base]Number**. The **Base** parameter is a decimal number between 2 and 36 inclusive, representing the arithmetic base. The **Number** parameter is a number in that base. If you omit the **Base** parameter, the shell uses a base of 10.

Arithmetic expressions use the same syntax, precedence, and associativity of expression as the C programming language. All of the integral operators, other than double plus (`++`), double hyphen (`--`), question mark-colon (`?:`), and comma (`,`), are supported. The following table lists valid Korn shell or POSIX shell operators in decreasing order of precedence:

Operator	Definition
-	Unary minus
!	Logical negation
~	Bitwise negation
*	Multiplication
/	Division
%	Remainder
+	Addition

Operator	Definition
-	Subtraction
<<, >>	Left shift, right shift
<=, >=, <>, ==, !=	Comparison
&	Bitwise AND
^	Bitwise exclusive OR
	Bitwise OR
&&	Logical AND
	Logical OR
= *=, /=, &= +=, -=, <<=, >>=, &=, ^=, =	Assignment

Many arithmetic operators, such as `*`, `&`, `<`, and `>`, have special meaning to the Korn shell or POSIX shell. These characters must be quoted. For example, to multiply the current value of `y` by 5 and reassign the new value to `y`, use the expression:

```
let "y = y * 5"
```

Enclosing the expression in quotation marks removes the special meaning of the `*` character.

You can group operations inside **let** command expressions to force grouping. For example, in the expression:

```
let "z = q * (z - 10)"
```

the command multiplies `q` by the reduced value of `z`.

The Korn shell or POSIX shell includes an alternative form of the **let** command if only a single expression is to be evaluated. The shell treats commands enclosed in `(())` as quoted expressions. Therefore, the expression:

```
((x = x / 3))
```

is equivalent to:

```
let "x = x / 3"
```

Named parameters are referenced by name within an arithmetic expression without using the parameter substitution syntax. When a named parameter is referenced, its value is evaluated as an arithmetic expression.

Specify an internal integer representation of a named parameter with the **-i** flag of the **typeset** special built-in command. Using the **-i** flag, arithmetic evaluation is performed on the value of each assignment to a named parameter. If you do not specify an arithmetic base, the first assignment to the parameter determines the arithmetic base. This base is used when parameter substitution occurs.

Field splitting in the Korn shell or POSIX shell

After performing command substitution, the Korn shell scans the results of substitutions for those field separator characters found in the **IFS** (Internal Field Separator) variable. Where such characters are found, the shell splits the substitutions into distinct arguments.

The shell retains explicit null arguments (`"` or `'`) and removes implicit null arguments (those resulting from parameters that have no values).

- If the value of *IFS* is a space, tab, or newline character, or if it is not set, any sequence of space, tab, or newline characters at the beginning or end of the input will be ignored and any sequence of those characters within the input will delimit a field. For example, the following input yields two fields, **school** and **days**:

```
<newline><space><tab>school<tab><tab>days<space>
```

- Otherwise, and if the value of **IFS** is not null, the following rules apply in sequence. *IFS white space* is used to mean any sequence (zero or more instances) of white-space characters that are in the **IFS** value (for example, if **IFS** contains space/comma/tab, any sequence of space and tab characters is considered **IFS** white space).
 1. **IFS** white space is ignored at the beginning and end of the input.
 2. Each occurrence in the input of an **IFS** character that is not **IFS** white space, along with any adjacent **IFS** white space, delimits a field.
 3. Nonzero length **IFS** white space delimits a field.

File name substitution in the Korn shell or POSIX shell

The Korn shell, or POSIX shell, performs file name substitution by scanning each command word specified by the *Word* variable for certain characters.

If a command word includes the *****, **?** or **[** characters, and the **-f** flag has not been set, the shell regards the word as a pattern. The shell replaces the word with file names, sorted according to the collating sequence in effect in the current locale, that match that pattern. If the shell does not find a file name to match the pattern, it does not change the word.

When the shell uses a pattern for file name substitution, the **.** and **/** characters must be matched explicitly.

Note: The Korn shell does not treat these characters specially in other instances of pattern matching.

These pattern-matching characters indicate the following substitutions:

- * Matches any string, including the null string.
- ? Matches any single character.
- [...] Matches any one of the enclosed characters. A pair of characters separated by a hyphen (-) matches any character lexically within the inclusive range of that pair, according to the collating sequence in effect in the current locale. If the first character following the opening [is **!**, then any character not enclosed is matched. A hyphen (-) can be included in the character set by putting it as the first or last character.

You can also use the `[:charclass:]` notation to match file names within a range indication. This format instructs the system to match any single character belonging to `class`. The definition of which characters constitute a specific character class is present through the **LC_CTYPE** category of the `setlocale` subroutine. All character classes specified in the current locale are recognized.

The names of some of the character classes are as follows:

- **alnum**
- **alpha**
- **cntrl**
- **digit**
- **graph**
- **lower**
- **print**
- **punct**
- **space**

- **upper**
- **xdigit**

For example, `[[[:upper:]]]` matches any uppercase letter.

The Korn shell supports file name expansion based on collating elements, symbols, or equivalence classes.

A *PatternList* is a list of one or more patterns separated from each other with a `|`. Composite patterns are formed with one or more of the following:

<code>?(PatternList)</code>	Optionally matches any one of the given patterns
<code>*(PatternList)</code>	Matches zero or more occurrences of the given patterns
<code>+(PatternList)</code>	Matches one or more occurrences of the given patterns
<code>@(PatternList)</code>	Matches exactly one of the given patterns
<code>!(PatternList)</code>	Matches anything, except one of the given patterns

Pattern matching has some restrictions. If the first character of a file name is a dot (`.`), it can be matched only by a pattern that also begins with a dot. For example, `*` matches the file names **myfile** and **yourfile** but not the file names **.myfile** and **.yourfile**. To match these files, use a pattern such as the following:

```
.*file
```

If a pattern does not match any file names, then the pattern itself is returned as the result of the attempted match.

File and directory names should not contain the characters `*`, `?`, `[`, or `]` because they can cause infinite recursion (that is, infinite loops) during pattern-matching attempts.

Quote removal

Some characters will be removed if they are not quoted.

The quote characters, backslash (`\`), single quote (`'`), and double quote (`"`) that were present in the original word will be removed unless they have themselves been quoted.

Input and output redirection in the Korn shell or POSIX shell

Before the Korn shell executes a command, it scans the command line for redirection characters. These special notations direct the shell to redirect input and output.

Redirection characters can appear anywhere in a simple command or can precede or follow a command. They are not passed on to the invoked command.

The shell performs command and parameter substitution before using the **Word** or **Digit** parameter except as noted. File name substitution occurs only if the pattern matches a single file and blank interpretation is not performed.

<code><Word</code>	Uses the file specified by the Word parameter as standard input (file descriptor 0).
<code>>Word</code>	Uses the file specified by the Word parameter as standard output (file descriptor 1). If the file does not exist, the shell creates it. If the file exists and the noclobber option is on, an error results; otherwise, the file is truncated to zero length. Note: When multiple shells have the noclobber option set and they redirect output to the same file, there could be a race condition, which might result in more than one of these shell processes writing to the file. The shell does not detect or prevent such race conditions.

<code>> Word</code>	Same as the <code>>Word</code> command, except that this redirection statement overrides the noclobber option.
<code>>>Word</code>	Uses the file specified by the Word parameter as standard output. If the file currently exists, the shell appends the output to it (by first seeking the end-of-file character). If the file does not exist, the shell creates it.
<code><>Word</code>	Opens the file specified by the Word parameter for reading and writing as standard input.
<code><<[-]Word</code>	Reads each line of shell input until it locates a line containing only the value of the Word parameter or an end-of-file character. The shell does not perform parameter substitution, command substitution, or file name substitution on the file specified. The resulting document, called a <i>here document</i> , becomes the standard input. If any character of the Word parameter is quoted, no interpretation is placed upon the characters of the document.

The *here* document is treated as a single word that begins after the next newline character and continues until there is a line containing only the delimiter, with no trailing blank characters. Then the next *here* document, if any, starts. The format is as follows:

```
[n]<<word
  here document
delimiter
```

If any character in *word* is quoted, the delimiter is formed by removing the quote on *word*. The *here* document lines will not be expanded. Otherwise, the delimiter is the *word* itself. If no characters in *word* are quoted, all lines of the *here* document will be expanded for parameter expansion, command substitution, and arithmetic expansion.

The shell performs parameter substitution for the redirected data. To prevent the shell from interpreting the `\`, `$`, and single quotation mark (`'`) characters and the first character of the **Word** parameter, precede the characters with a `\` character.

If a hyphen (`-`) is appended to `<<`, the shell strips all leading tabs from the **Word** parameter and the document.

<code><&Digit</code>	Duplicates standard input from the file descriptor specified by the Digit parameter
<code>>& Digit</code>	Duplicates standard output in the file descriptor specified by the Digit parameter
<code><&-</code>	Closes standard input
<code>>&-</code>	Closes standard output
<code><&p</code>	Moves input from the co-process to standard input
<code>>&p</code>	Moves output to the co-process to standard output

If one of these redirection options is preceded by a digit, then the file descriptor number referred to is specified by the digit (instead of the default 0 or 1). In the following example, the shell opens file descriptor 2 for writing as a duplicate of file descriptor 1:

```
... 2>&1
```

The order in which redirections are specified is significant. The shell evaluates each redirection in terms of the (*FileDescriptor*, *File*) association at the time of evaluation. For example, in the statement:

```
... 1>File 2>&1
```

the file descriptor 1 is associated with the file specified by the **File** parameter. The shell associates file descriptor 2 with the file associated with file descriptor 1 (*File*). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had previously been) and file descriptor 1 would be associated with the file specified by the **File** parameter.

If a command is followed by an ampersand (&) and job control is not active, the default standard input for the command is the empty file `/dev/null`. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input and output specifications.

For more information about redirection, see Chapter 29, “Input and output redirection,” on page 371.

Coprocess facility

The Korn shell, or POSIX shell, allows you to run one or more commands as background processes. These commands, run from within a shell script, are called *coprocesses*.

Designate a coprocess by placing the `|&` operator after a command. Both standard input and output of the command are piped to your script.

A coprocess must meet the following restrictions:

- Include a newline character at the end of each message
- Send each output message to standard output
- Clear its standard output after each message

The following example demonstrates how input is passed to and returned from a coprocess:

```
echo "Initial process"
./FileB.sh |&
read -p a b c d
echo "Read from coprocess: $a $b $c $d"
print -p "Passed to the coprocess"
read -p a b c d
echo "Passed back from coprocess: $a $b $c $d"

FileB.sh
    echo "The coprocess is running"
    read a b c d
    echo $a $b $c $d
```

The resulting standard output is as follows:

```
Initial process
Read from coprocess: The coprocess is running
Passed back from coprocess: Passed to the coprocess
```

Use the **print -p** command to write to the coprocess. Use the **read -p** command to read from the coprocess.

Redirection of coprocess input and output

The standard input and output of a coprocess is reassigned to a numbered file descriptor by using I/O redirection.

For example, the command:

```
exec 5>&p
```

moves the input of the coprocess to file descriptor 5.

After this coprocess has completed, you can use standard redirection syntax to redirect command output to the coprocess. You can also start another coprocess. Output from both coprocesses is connected to the same pipe and is read with the **read -p** command. To stop the coprocess, type the following:

```
read -u5
```

Exit status in the Korn shell or POSIX shell

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. Otherwise, the shell returns the exit status of the last command carried out.

The shell reports detected runtime errors by printing the command or function name and the error condition. If the number of the line on which an error occurred is greater than 1, then the line number is also printed in [] (brackets) after the command or function name.

For a noninteractive shell, an error encountered by a special built-in or other type of command will cause the shell to write a diagnostic message as shown in the following table:

Error	Special Built-In	Other Utilities
Shell language syntax error	will exit	will exit
Utility syntax error (option or operand error)	will exit	will not exit
Redirection error	will exit	will not exit
Variable assignment error	will exit	will not exit
Expansion error	will exit	will exit
Command not found	not applicable	may exit
Dot script not found	will exit	not applicable

If any of the errors shown as "will (may) exit" occur in a subshell, the subshell will (may) exit with a nonzero status, but the script containing the subshell will not exit because of the error.

In all cases shown in the table, an interactive shell will write a diagnostic message to standard error, without exiting.

Korn shell or POSIX shell built-in commands

Special commands are built in to the Korn shell and POSIX shell and executed in the shell process.

Unless otherwise indicated, the output is written to file descriptor 1 and the exit status is zero (0) if the command does not contain any syntax errors. Input and output redirection is permitted. There are two types of built-in commands: *special built-in commands* and *regular built-in commands*.

Special built-in commands differ from regular built-in commands in the following ways:

- A syntax error in a special built-in command might cause the shell executing the command to end. This does not happen if you have a syntax error in a regular built-in command. If a syntax error in a special built-in command does not end the shell program, the exit value is nonzero.
- Variable assignments specified with special built-in commands remain in effect after the command completes.
- I/O redirections are processed after parameter assignments.

In addition, words that are in the form of a parameter assignment following the **export**, **readonly**, and **typeset** special commands are expanded with the same rules as a parameter assignment. Tilde substitution is performed after the =, and word-splitting and file name substitution are not performed.

For an alphabetical listing of these commands, see "List of Korn shell or POSIX shell special built-in commands" on page 470 and "Korn shell or POSIX shell regular built-in commands" on page 470.

Special built-in command descriptions for the Korn shell or POSIX shell

Special commands are built into the Korn shell and POSIX shell and executed in the shell process.

The special built-in commands of the Korn shell are described below:

:	eval	newgrp	shift
.	exec	readonly	times
break	exit	return	trap
continue	export	set	typeset unset

: [<i>Argument ...</i>]	Expands only arguments. It is used when a command is necessary, as in the <i>then</i> condition of an if command, but nothing is to be done by the command.
----------------------------------	--

. <i>File</i> [<i>Argument ...</i>]	Reads the complete specified file and then executes the commands. The commands are executed in the current shell environment. The search path specified by the <i>PATH</i> variable is used to find the directory containing the specified file. If any arguments are specified, they become the positional parameters. Otherwise, the positional parameters are unchanged. The exit status is the exit status of the most recent command executed. See “Parameter substitution in the Korn shell or POSIX shell” on page 449 for more information on positional parameters. Note: The .File [<i>Argument ...</i>] command reads the entire file before any commands are carried out. Therefore, the alias and unalias commands in the file do not apply to any functions defined in the file.
--	---

break [<i>n</i>]	Exits from the enclosing for , while , until , or select loop, if one exists. If you specify the <i>n</i> parameter, the command breaks the number of levels specified by the <i>n</i> parameter. The value of <i>n</i> is any integer equal to or greater than 1.
---------------------------	--

continue [<i>n</i>]	Resumes the next iteration of the enclosing for , while , until , or select loop. If you specify the <i>n</i> parameter, the command resumes at the <i>n</i> th enclosing loop. The value of <i>n</i> is any integer equal to or greater than 1.
------------------------------	---

eval [<i>Argument ...</i>]	Reads the specified arguments as input to the shell and executes the resulting command or commands.
-------------------------------------	---

exec [<i>Argument ...</i>]	Executes the command specified by the argument in place of this shell (without creating a new process). Input and output arguments can appear and affect the current process. If you do not specify an argument, the exec command modifies file descriptors as prescribed by the input and output redirection list. In this case, any file descriptor numbers greater than 2 that are opened with this mechanism are closed when invoking another program.
-------------------------------------	---

exit [<i>n</i>]	Exits the shell with the exit status specified by the <i>n</i> parameter. The <i>n</i> parameter must be an unsigned decimal integer with range 0-255. If you omit the <i>n</i> parameter, the exit status is that of the most recent command executed. An end-of-file character also exits the shell unless the ignoreeof option of the set special command is turned on.
--------------------------	--

export -p [<i>Name</i> [= <i>Value</i>]] ...	Marks the specified names for automatic export to the environment of subsequently executed commands. -p writes to standard output the names and values of all exported variables, in the following format: "export %s= %s\n", <name> <value>
---	---

newgrp [<i>Group</i>]	Equivalent to the exec/usr/bin/newgrp [<i>Group</i>] command. Note: This command does not return.
--------------------------------	--

readonly -p [<i>Name</i> = <i>Value</i>] ...	<p>Marks the names specified by the <i>Name</i> parameter as read-only. These names cannot be changed by subsequent assignment.</p> <p>-p writes to standard output the names and values of all exported variables, in the following format:</p> <pre>"export %s= %s\n", <name> <value></pre>
return [<i>n</i>]	<p>Causes a shell function to return to the invoking script. The return status is specified by the <i>n</i> parameter. If you omit the <i>n</i> parameter, the return status is that of the most recent command executed. If you invoke the return command outside of a function or a script, then it is the same as an exit command.</p>
set [+ - abCefhkmnostuvx] [+ -o <i>Option</i>]... [+ - A <i>Name</i>] [<i>Argument</i> ...]	<p>If no options or arguments are specified, the set command writes the names and values of all shell variables in the collation sequence of the current locale. When options are specified, they will set or unset attributes of the shell, described as follows:</p> <ul style="list-style-type: none"> -A Array assignment. Unsets the <i>Name</i> parameter and assigns values sequentially from the specified <i>Argument</i> parameter list. If the +A flag is used, the <i>Name</i> parameter is not unset first. -a Exports automatically all subsequent parameters that are defined. -b Notifies the user asynchronously of background job completions. -C Equivalent to set -o noclobber. -e Executes the ERR trap, if set, and exits if a command has a nonzero exit status. This mode is disabled while reading profiles. -f Disables file name substitution. -h Designates each command as a tracked alias when first encountered. -k Places all parameter-assignment arguments in the environment for a command, not only those arguments that precede the command name. -m Runs background jobs in a separate process and prints a line upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells. For more information, see "Job control in the Korn shell or POSIX shell" on page 472. -n Reads commands and checks them for syntax errors, but does not execute them. This flag is ignored for interactive shells.

-o *Option*

Prints current option settings and an error message if you do not specify an argument. You can set more than one option on a single **ksh** command line. If the **+o** flag is used, the specified option is unset. When arguments are specified, they will cause positional parameters to be set or unset. Arguments, as specified by the *Option* variable, can be one of the following:

allexport

Same as the **-a** flag.

bgnice

Runs all background jobs at a lower priority. This is the default mode.

emacs

Enters an emacs-style inline editor for command entry.

errexit Same as the **-e** flag.

gmacs

Enters a gmacs-style inline editor for command entry.

ignoreeof

Does not exit the shell when it encounters an end-of-file character. To exit the shell, you must use the **exit** command or press the Ctrl-D key sequence more than 11 times.

keyword

Same as the **-k** flag.

Note: This flag is for backward compatibility with the Bourne shell only. Its use is strongly discouraged.

markdirs

Appends a backslash / to all directory names that are a result of file name substitution.

monitor

Same as the **-m** flag.

noclobber

Prevents redirection from truncating existing files. When you specify this option, a vertical bar must follow the redirection symbol (>|) to truncate a file.

noexec

Same as the **-n** flag.

noglob

Same as the **-f** flag.

nolog Prevents function definitions in **.profile** and **\$ENV** files from being saved in the history file.

nounset

Same as the **-u** flag.

privileged

Same as the **-p** flag.

-
- trackall** Same as the **-h** flag.
 - verbose** Same as the **-v** flag.
 - vi** Enters the insert mode of a vi-style inline editor for command entry. Entering escape character 033 puts the editor into the move mode. A return sends the line.
 - viraw** Processes each character as it is typed in vi mode.
 - xtrace** Same as the **-x** flag.
 - p** Disables processing of the **\$HOME/.profile** file and uses the **/etc/suid_profile** file instead of the **ENV** file. This mode is enabled whenever the effective user ID (UID) or group ID (GID) is not equal to the real UID or GID. Turning off this option sets the effective UID or GID to the real UID and GID.
Note: The system does not support the **-p** option because the operating system does not support **setuid** shell scripts.
 - s** Sorts the positional parameters lexicographically.
 - t** Exits after reading and executing one command.
Note: This flag is for backward compatibility with the Bourne shell only. Its use is strongly discouraged.
 - u** Treats unset parameters as errors when substituting.
 - v** Prints shell input lines as they are read.
 - x** Prints commands and their arguments as they are executed.
 - Turns off the **-x** and **-v** flags and stops examining arguments for flags.
 - Prevents any flags from being changed. This option is useful in setting the **\$1** parameter to a value beginning with **-**. If no arguments follow this flag, the positional parameters are not set.

Preceding any of the **set** command flags with a **+** rather than a **-** turns off the flag. You can use these flags when you invoke the shell. When **'set +o'** is invoked without any arguments, it displays the current option settings in a format that is suitable for re-input to the shell as commands that achieve the same option setting. The current set of flags is found in the **\$-** parameter. Unless you specify the **-A** flag, the remaining arguments are positional parameters and are assigned, in order, to **\$1**, **\$2**, . . . , and so on. If no arguments are given, the names and values of all named parameters are printed to standard output.

shift [<i>n</i>]	Renames the positional parameters, beginning with \$n+1 ... through \$1 The default value of the <i>n</i> parameter is 1. The <i>n</i> parameter is any arithmetic expression that evaluates to a nonnegative number less than or equal to the \$# parameter.
times	Prints the accumulated user and system times for the shell and for processes run from the shell.

trap [*Command*]
[*Signal*] ...

Runs the specified command when the shell receives the specified signal or signals. The *Command* parameter is read once when the trap is set and once when the trap is taken. The *Signal* parameter can be given as a number or as the name of the signal. Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective.

If the command is -, all traps are reset to their original values. If you omit the command and the first signal is a numeric signal number, then the **ksh** command resets the value of the *Signal* parameter or parameters to the original values.

Note: If you omit the command and the first signal is a symbolic name, the signal is interpreted as a command.

If the value of the *Signal* parameter is the **ERR** signal, the specified command is carried out whenever a command has a nonzero exit status. If the signal is **DEBUG**, then the specified command is carried out after each command. If the value of the *Signal* parameter is the **0** or **EXIT** signal and the **trap** command is executed inside the body of a function, the specified command is carried out after the function completes. If the *Signal* parameter is **0** or **EXIT** for a **trap** command set outside any function, the specified command is carried out on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

For a complete list of *Signal* parameter values used in the **trap** command without the **SIG** prefix, see the **sigaction**, **sigvec**, or **signal** subroutine in the *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 2*.

typeset [+HLRZfirtux
[*n*]] [*Name*[=*Value*]] ...

Sets attributes and values for shell parameters. When invoked inside a function, a new instance of the *Name* parameter is created. The parameter value and type are restored when the function completes. You can specify the following flags with the **typeset** command:

- H** Provides AIX-to-host-file mapping on non-AIX machines.
 - L** Left-justifies and removes leading blanks from the *Value* parameter. If the *n* parameter has a nonzero value, it defines the width of the field; otherwise, it is determined by the width of the value of its first assignment. When the parameter is assigned, it is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading zeros are removed if the **-Z** flag is also set. The **-R** flag is turned off.
 - R** Right-justifies and fills with leading blanks. If the *n* parameter has a nonzero value, it defines the width of the field; otherwise, it is determined by the width of the value of its first assignment. The field remains filled with blanks or is truncated from the end if the parameter is reassigned. The **L** flag is turned off.
 - Z** Right-justifies and fills with leading zeros if the first nonblank character is a digit and the **-L** flag has not been set. If the *n* parameter has a nonzero value, it defines the width of the field; otherwise, it is determined by the width of the value of its first assignment.
 - f** Indicates that the names refer to function names rather than parameter names. No assignments can be made and the only other valid flags are **-t**, **-u**, and **-x**. The **-t** flag turns on execution tracing for this function. The **-u** flag causes this function to be marked undefined. The *FPATH* variable is searched to find the function definition when the function is referenced. The **-x** flag allows the function definition to remain in effect across shell scripts that are not a separate invocation of the **ksh** command.
 - i** Identifies the parameter as an integer, making arithmetic faster. If the *n* parameter has a nonzero value, it defines the output arithmetic base; otherwise, the first assignment determines the output base.
 - l** Converts all uppercase characters to lowercase. The **-u** uppercase conversion flag is turned off.
 - r** Marks the names specified by the *Name* parameter as read-only. These names cannot be changed by subsequent assignment.
-

-
- t** Tags the named parameters. Tags can be defined by the user and have no special meaning to the shell.
 - u** Converts all lowercase characters to uppercase characters. The **-l** lowercase flag is turned off.
 - x** Marks the name specified by the *Name* parameter for automatic export to the environment of subsequently executed commands.
- Using a + rather than a - turns off the **typeset** command flags. If you do not specify *Name* parameters but do specify flags, a list of names (and optionally the values) of the parameters that have these flags set is printed. (Using a + rather than a - keeps the values from being printed.) If you do not specify any names or flags, the names and attributes of all parameters are printed.
-

unset [-fv] Name ... Unsets the values and attributes of the parameters given by the list of names. If **-v** is specified, *Name* refers to a variable name, and the shell will unset it and remove it from the environment. Read-only variables cannot be unset. Unsetting the *ERRNO*, *LINENO*, *MAILCHECK*, *OPTARG*, *OPTIND*, *RANDOM*, *SECONDS*, *TMOU*T, and underscore (`_`) variables removes their special meanings even if they are subsequently assigned.

If the **-f** flag is set, then *Name* refers to a function name, and the shell will unset the function definition.

Regular built-in command descriptions for the Korn shell or POSIX shell

The built-in commands for the Korn or POSIX shells are described here.

The Korn shell provides the following regular built-in commands:

alias	fg	print	ulimit	
bg	getopts	pwd	umask	
cd	jobs	read	unalias	
command	kill	setgroups	wait	
echo	let	setenv	test	whence
fc				

alias [-t] [-x] *[AliasName[= String]]*
 ... Creates or redefines alias definitions or writes existing alias definitions to standard output.
 For more information, see the **alias** command.

bg [JobID...] Puts each specified job into the background. The current job is put in the background if a *JobID* parameter is not specified. See “Job control in the Korn shell or POSIX shell” on page 472 for more information about job control.

For more information about running jobs in the background, see the **bg** command.

cd [Argument]

cd <i>Old New</i>	<p>This command can be in either of two forms. In the first form, it changes the current directory to the one specified by the <i>Argument</i> parameter. If the value of the <i>Argument</i> parameter is a hyphen (-), the directory is changed to the previous directory. The HOME shell variable is the default value of the <i>Argument</i> parameter. The PWD variable is set to the current directory.</p> <p>The CDPATH shell variable defines the search path for the directory containing the value of the <i>Argument</i> parameter. Alternative directory names are separated by a colon (:). The default path is null, specifying the current directory. The current directory is specified by a null path name, which appears immediately after the equal sign or between the colon delimiters anywhere in the path list. If the specified argument begins with a slash (/), the search path is not used. Otherwise, each directory in the path is searched for the argument.</p> <p>The second form of the cd command substitutes the string specified by the <i>New</i> variable for the string specified by the <i>Old</i> variable in the current directory name, PWD, and tries to change to this new directory.</p>
--------------------------	--

command [-p] <i>CommandName</i> [<i>Argument ...</i>]	
command [-v -V] <i>CommandName</i>	<p>Causes the shell to treat the specified command and arguments as a simple command, suppressing shell-function lookup.</p> <p>For more information, see the command command.</p>

echo [<i>String ...</i>]	<p>Writes character strings to standard output. See the echo command for usage and description. The -n flag is not supported.</p>
-----------------------------------	---

fc [-r] [-e <i>Editor</i>] [<i>First</i> [<i>Last</i>]]	
fc -l [-n] [-r] [<i>First</i> [<i>Last</i>]]	
fc -s [<i>Old= New</i>] [<i>First</i>]	<p>Displays the contents of your command history file or invokes an editor to modify and re-execute commands previously entered in the shell.</p> <p>For more information, see the fc command.</p>

fg [<i>JobID</i>]	<p>Brings each job specified into the foreground. If you do not specify any jobs, the command brings the current job into the foreground.</p> <p>For more information about running jobs in the foreground, see the fg command.</p>
----------------------------	--

getopts <i>OptionString</i> <i>Name</i> [<i>Argument ...</i>]	<p>Checks the <i>Argument</i> parameter for legal options.</p> <p>For more information, see the getopts command.</p>
---	---

jobs [-l -n -p] [<i>JobID ...</i>]	<p>Displays the status of jobs started in the current shell environment. If no specific job is specified with the <i>JobID</i> parameter, status information for all active jobs is displayed. If a job termination is reported, the shell removes that job's process ID from the list of those known by the current shell environment.</p> <p>For more information, see the jobs command.</p>
---	---

kill [-s { <i>SignalName</i> <i>SignalNumber</i> }] <i>ProcessID...</i>	<p>Sends a signal (by default, the SIGTERM signal) to a running process. This default action normally stops processes. If you want to stop a process, specify the process ID (PID) in the <i>ProcessID</i> variable. The shell reports the PID of each process that is running in the background (unless you start more than one process in a pipeline, in which case the shell reports the number of the last process). You can also use the ps command to find the process ID number of commands.</p>
kill [- <i>SignalName</i> - <i>SignalNumber</i>] <i>ProcessID...</i>	
kill -l [<i>ExitStatus</i>]	

For more information, see the **kill** command.

let <i>Expression ...</i>	Evaluates specified arithmetic expressions. The exit status is 0 if the value of the last expression is nonzero, and 1 otherwise. See “Arithmetic evaluation in the Korn shell or POSIX shell” on page 454 for more information.
print [-Rnprsu [<i>n</i>]] [<i>Argument ...</i>]	<p>Prints shell output. If you do not specify any flags, or if you specify the hyphen (-) or double hyphen (--) flags, the arguments are printed to standard output as described by the echo command. The flags do the following:</p> <ul style="list-style-type: none"> -R Prints in raw mode (the escape conventions of the echo command are ignored). The -R flag prints all subsequent arguments and flags other than -n. -n Prevents a newline character from being added to the output. -p Writes the arguments to the pipe of the process run with & instead of to standard output. -r Prints in raw mode. The escape conventions of the echo command are ignored. -s Writes the arguments to the history file instead of to standard output. -u Specifies a one-digit file descriptor unit number, <i>n</i>, on which the output is placed. The default is 1.
pwd	Equivalent to <code>print -r - \$PWD</code> .
	Note: The internal Korn shell pwd command does not support symbolic links.
read [-prsu [<i>n</i>]] [<i>Name?Prompt</i>] [<i>Name...</i>]	<p>Takes shell input. One line is read and broken up into fields, using the characters in the IFS variable as separators.</p> <p>For more information, see the read command.</p>
setgroups	Executes the <code>/usr/bin/setgroups</code> command, which runs as a separate shell. See the setgroups command for information on this command. There is one difference, however. The setgroups built-in command invokes a subshell, but the setgroups command replaces the currently executing shell. Because the built-in command is supported only for compatibility, it is recommended that scripts use the absolute path name <code>/usr/bin/setgroups</code> rather than the shell built-in command.
setsenv	Executes the <code>/usr/bin/setsenv</code> command, which replaces the currently executing shell. See the setsenv command for information on this command.
test	Same as [<i>expression</i>]. See “Conditional expressions for the Korn shell or POSIX shell” on page 471 for usage and description.

ulimit [-HSacdfmst] [*Limit*]

Sets or displays user-process resource limits as defined in the `/etc/security/limits` file. This file contains the following default limits:

```
fsize = 2097151
core = 2048
cpu = 3600
data = 131072
rss = 65536
stack = 8192
```

These values are used as default settings when a user is added to the system. The values are set with the **mkuser** command when the user is added to the system or changed with the **chuser** command.

Limits are categorized as either soft or hard. Users might change their soft limits, up to the maximum set by the hard limits, with the **ulimit** command. You must have root user authority to change resource hard limits.

Many systems do not contain one or more of these limits. The limit for a specified resource is set when the *Limit* parameter is specified. The value of the *Limit* parameter can be a number in the unit specified with each resource or the value `unlimited`. You can specify the following **ulimit** command flags:

- H** Specifies that the hard limit for the given resource is set. If you have root user authority, you can increase the hard limit. Any user can decrease it.
- S** Specifies that the soft limit for the given resource is set. A soft limit can be increased up to the value of the hard limit. If neither the **-H** or **-S** options are specified, the limit applies to both.
- a** Lists all of the current resource limits.
- c** Specifies the number of 512-byte blocks on the size of core dumps.
- d** Specifies the size, in KB, of the data area.
- f** Specifies the number of 512-byte blocks for files written by child processes (files of any size can be read).
- m** Specifies the number of KB for the size of physical memory.
- n** Specifies the limit on the number of file descriptors a process might have open.
- s** Specifies the number of KB for the size of the stack area.
- t** Specifies the number of seconds to be used by each process.

The current resource limit is printed when you omit the *Limit* variable. The soft limit is printed unless you specify the **-H** flag. When you specify more than one resource, the limit name and unit is printed before the value. If no option is given, the **-f** flag is assumed. When you change the value, set both hard and soft limits to *Limit* unless you specify **-H** or **-S**.

For more information about user and system resource limits, see the **getrlimit**, **setrlimit**, or **vlimit** subroutine.

umask [-S] [*Mask*]

Determines file permissions. This value, along with the permissions of the creating process, determines a file's permissions when the file is created. The default is 022. If the *Mask* parameter is not specified, the **umask** command displays to standard output the file-mode creation mask of the current shell environment.

For more information about file permissions, see the **umask** command.

unalias { **-a** | *AliasName...* }

Removes the definition for each alias name specified, or removes all alias definitions if the **-a** flag is used. Alias definitions are removed from the current shell environment.

For more information, see the **unalias** command.

wait [<i>ProcessID...</i>]	Waits for the specified job and terminates. If you do not specify a job, the command waits for all currently active child processes. The exit status from this command is that of the process for which it waits.
	For more information, see the wait command.
whence [-pv] <i>Name</i> ...	Indicates, for each name specified, how it would be interpreted if used as a command name. When used without either flag, whence will display the absolute path name, if any, that corresponds to each name.
-p	Performs a path search for the specified name or names even if these are aliases, functions, or reserved words.
-v	Produces a more verbose report that specifies the type of each name.

List of Korn shell or POSIX shell special built-in commands

Special commands are built into the Korn shell and POSIX shell and executed in the shell process.

: (colon)	Expands only arguments.
. (dot)	Reads a specified file and then executes the commands.
break	Exits from the enclosing for , while , until , or select loop, if one exists.
continue	Resumes the next iteration of the enclosing for , while , until , or select loop.
eval	Reads the arguments as input to the shell and executes the resulting command or commands.
exec	Executes the command specified by the <i>Argument</i> parameter, instead of this shell, without creating a new process.
exit	Exits the shell whose exit status is specified by the <i>n</i> parameter.
export	Marks names for automatic export to the environment of subsequently executed commands.
newgrp	Equivalent to the exec/usr/bin/newgrp [<i>Group ...</i>] command.
readonly	Marks the specified names read-only.
return	Causes a shell to return to the invoking script.
set	Unless options or arguments are specified, writes the names and values of all shell variables in the collation sequence of the current locale.
shift	Renames positional parameters.
times	Prints the accumulated user and system times for both the shell and the processes run from the shell.
trap	Runs a specified command when the shell receives a specified signal or signals.
typeset	Sets attributes and values for shell parameters.
unset	Unsets the values and attributes of the specified parameters.

Korn shell or POSIX shell regular built-in commands

The following is a list of the Korn shell or POSIX shell regular built-in commands.

alias	Prints a list of aliases to standard output.
bg	Puts specified jobs in the background.
cd	Changes the current directory to the specified directory or substitutes the current string with the specified string.
echo	Writes character strings to standard output.
fc	Selects a range of commands from the last <i>HISTSIZE</i> variable command typed at the terminal. Re-executes the specified command after old-to-new substitution is performed.
fg	Brings the specified job to the foreground.
getopts	Checks the <i>Argument</i> parameter for legal options.
jobs	Lists information for the specified jobs.
kill	Sends the TERM (terminate) signal to specified jobs or processes.
let	Evaluates specified arithmetic expressions.
print	Prints shell output.

pwd	Equivalent to the print -r -\$PWD command.
read	Takes shell input.
ulimit	Sets or displays user process resource limits as defined in the /etc/security/limits file.
umask	Determines file permissions.
unalias	Removes the parameters in the list of names from the alias list.
wait	Waits for the specified job and terminates.
whence	Indicates how each specified name would be interpreted if used as a command name.

For more information, see “Korn shell or POSIX shell built-in commands” on page 460.

Conditional expressions for the Korn shell or POSIX shell

A conditional expression is used with the `[[` compound command to test attributes of files and to compare strings.

Word splitting and file name substitution are not performed on words appearing between `[[` and `]]`. Each expression is constructed from one or more of the following unary or binary expressions:

-a <i>File</i>	True, if the specified file is a symbolic link that points to another file that does exist.
-b <i>File</i>	True, if the specified file exists and is a block special file.
-c <i>File</i>	True, if the specified file exists and is a character special file.
-d <i>File</i>	True, if the specified file exists and is a directory.
-e <i>File</i>	True, if the specified file exists.
-f <i>File</i>	True, if the specified file exists and is an ordinary file.
-g <i>File</i>	True, if the specified file exists and its setgid bit is set.
-h <i>File</i>	True, if the specified file exists and is a symbolic link.
-k <i>File</i>	True, if the specified file exists and its sticky bit is set.
-n <i>String</i>	True, if the length of the specified string is nonzero.
-o <i>Option</i>	True, if the specified option is on.
-p <i>File</i>	True, if the specified file exists and is a FIFO special file or a pipe.
-r <i>File</i>	True, if the specified file exists and is readable by the current process.
-s <i>File</i>	True, if the specified file exists and has a size greater than 0.
-t <i>FileDescriptor</i>	True, if specified file descriptor number is open and associated with a terminal device.
-u <i>File</i>	True, if the specified file exists and its setuid bit is set.
-w <i>File</i>	True, if the specified file exists and the write bit is on. However, the file will not be writable on a read-only file system even if this test indicates true.
-x <i>File</i>	True, if the specified file exists and the execute flag is on. If the specified file exists and is a directory, then the current process has permission to search in the directory.
-z <i>String</i>	True, if length of the specified string is 0.
-L <i>File</i>	True, if the specified file exists and is a symbolic link.
-O <i>File</i>	True, if the specified file exists and is owned by the effective user ID of this process.
-G <i>File</i>	True, if the specified file exists and its group matches the effective group ID of this process.
-S <i>File</i>	True, if the specified file exists and is a socket.
<i>File1</i> -nt <i>File2</i>	True, if <i>File1</i> exists and is newer than <i>File2</i> .
<i>File1</i> -ot <i>File2</i>	True, if <i>File1</i> exists and is older than <i>File2</i> .
<i>File1</i> -ef <i>File2</i>	True, if <i>File1</i> and <i>File2</i> exist and refer to the same file.
<i>String1</i> = <i>String2</i>	True, if <i>String1</i> is equal to <i>String2</i> .
<i>String1</i> != <i>String2</i>	True, if <i>String1</i> is not equal to <i>String2</i> .
<i>String</i> = <i>Pattern</i>	True, if the specified string matches the specified pattern.
<i>String</i> != <i>Pattern</i>	True, if the specified string does not match the specified pattern.
<i>String1</i> < <i>String2</i>	True, if <i>String1</i> comes before <i>String2</i> based on the ASCII value of their characters.
<i>String1</i> > <i>String2</i>	True, if <i>String1</i> comes after <i>String2</i> based on the ASCII value of their characters.
<i>Expression1</i> -eq <i>Expression2</i>	True, if <i>Expression1</i> is equal to <i>Expression2</i> .

<i>Expression1</i> -ne <i>Expression2</i>	True, if <i>Expression1</i> is not equal to <i>Expression2</i> .
<i>Expression1</i> -lt <i>Expression2</i>	True, if <i>Expression1</i> is less than <i>Expression2</i> .
<i>Expression1</i> -gt <i>Expression2</i>	True, if <i>Expression1</i> is greater than <i>Expression2</i> .
<i>Expression1</i> -le <i>Expression2</i>	True, if <i>Expression1</i> is less than or equal to <i>Expression2</i> .
<i>Expression1</i> -ge <i>Expression2</i>	True, if <i>Expression1</i> is greater than or equal to <i>Expression2</i> .

Note: In each of the previous expressions, if the *File* variable is similar to **/dev/fd/*n***, where *n* is an integer, then the test is applied to the open file whose descriptor number is *n*.

You can construct a compound expression from these primitives, or smaller parts, by using any of the following expressions, listed in decreasing order of precedence:

<i>(Expression)</i>	True, if the specified expression is true. Used to group expressions.
! <i>Expression</i>	True, if the specified expression is false.
<i>Expression1</i> && <i>Expression2</i>	True, if <i>Expression1</i> and <i>Expression2</i> are both true.
<i>Expression1</i> <i>Expression2</i>	True, if either <i>Expression1</i> or <i>Expression2</i> is true.

Job control in the Korn shell or POSIX shell

The Korn shell, or POSIX shell, provides a facility to control command sequences, or *jobs*.

When you execute the **set -m** special command, the Korn shell associates a job with each pipeline. It keeps a table of current jobs, printed by the **jobs** command, and assigns them small integer numbers.

When a job is started in the background with an ampersand (&), the shell prints a line that looks like the following:

```
[1] 1234
```

This output indicates that the job, which was started in the background, was job number 1. It also shows that the job had one (top-level) process with a process ID of 1234.

If you are running a job and want to do something else, use the Ctrl-Z key sequence. This key sequence sends a **STOP** signal to the current job. The shell normally indicates that the job has been stopped and then displays a shell prompt. You can then manipulate the state of this job (putting it in the background with the **bg** command), run other commands, and then eventually return the job to the foreground with the **fg** command. The Ctrl-Z key sequence takes effect immediately, and is like an interrupt in that the shell discards pending output and unread input when you type the sequence.

A job being run in the background stops if it tries to read from the terminal. Background jobs are normally allowed to produce output. You can disable this option by issuing the **stty tostop** command. If you set this terminal option, then background jobs stop when they try to produce output or read input.

You can refer to jobs in the Korn shell in several ways. A job is referenced by the process ID of any of its processes or in one of the following ways:

%Number	Specifies the job with the given number.
%String	Specifies any job whose command line begins with the <i>String</i> variable.
??String	Specifies any job whose command line contains the <i>String</i> variable.
%%	Specifies the current job.

%+ Equivalent to `%%`.
%- Specifies the previous job.

This shell immediately recognizes changes in the process state. It normally informs you whenever a job becomes blocked so that no further progress is possible. The shell does this just before it prints a prompt so that it does not otherwise disturb your work.

When the monitor mode is on, each completed background job triggers traps set for the **CHLD** signal.

If you try to leave the shell (either by typing `exit` or using the Ctrl-D key sequence) while jobs are stopped or running, the system warns you with the message `There are stopped (running) jobs`. Use the **jobs** command to see which jobs are affected. If you immediately try to exit again, the shell terminates the stopped and running jobs without warning.

Signal handling

The **SIGINT** and **SIGQUIT** signals for an invoked command are ignored if the command is followed by an ampersand (&) and the job **monitor** option is not active. Otherwise, signals have the values that the shell inherits from its parent.

When a signal for which a trap has been set is received while the shell is waiting for the completion of a foreground command, the trap associated with that signal will not be executed until after the foreground command has completed. Therefore, a trap on a **CHILD** signal is not performed until the foreground job terminates.

Inline editing in the Korn shell or POSIX shell

Normally, you type each command line from a terminal device and follow it by a newline character (**RETURN** or **LINE FEED**). When you activate the emacs, gmacs, or vi inline editing option, you can edit the command line.

The following commands enter edit modes:

set -o emacs	Enters emacs editing mode and initiates an emacs-style inline editor. For more information, see “emacs editing mode” on page 474.
set -o gmacs	Enters emacs editing mode and initiates a gmacs-style inline editor. For more information, see “emacs editing mode” on page 474.
set -o vi	Enters vi editing mode and initiates a vi-style inline editor. For more information, see “vi editing mode” on page 476.

An editing option is automatically selected each time the *VISUAL* or *EDITOR* variable is assigned a value that ends in any of these option names.

Note: To use the editing features, your terminal must accept **RETURN** as a carriage return without line feed. A space must overwrite the current character on the screen.

Each editing mode opens a window at the current line. The window width is the value of the *COLUMNS* variable if it is defined; otherwise, the width is 80 character spaces. If the line is longer than the window width minus two, the system notifies you by displaying a mark at the end of the window. As the cursor moves and reaches the window boundaries, the window is centered about the cursor. The marks displayed are as follows:

>	Indicates that the line extends on the right side of the window.
<	Indicates that the line extends on the left side of the window.
*	Indicates that the line extends on both sides of the window.

The search commands in each edit mode provide access to the Korn shell history file. Only strings are matched. If the leading character in the string is a carat (^), the match must begin at the first character in the line.

emacs editing mode

The emacs editing mode is entered when you enable either the **emacs** or **gmacs** option. The only difference between these two modes is the way each handles the Ctrl-T edit command.

To edit, move the cursor to the point needing correction and insert or delete characters or words, as needed. All of the editing commands are control characters or escape sequences.

Edit commands operate from any place on a line (not only at the beginning). Do not press the Enter key or line-feed (Down Arrow) key after edit commands, except as noted.

Ctrl-F	Moves the cursor forward (right) one character.
Esc-F	Moves the cursor forward one word (a string of characters consisting of only letters, digits, and underscores).
Ctrl-B	Moves the cursor backward (left) one character.
Esc-B	Moves the cursor backward one word.
Ctrl-A	Moves the cursor to the beginning of the line.
Ctrl-E	Moves the cursor to the end of the line.
Ctrl-] c	Moves the cursor forward on the current line to the indicated character.
Esc-Ctrl-] c	Moves the cursor backward on the current line to the indicated character.
Ctrl-X Ctrl-X	Interchanges the cursor and the mark.
ERASE	Deletes the previous character. (User-defined erase character as defined by the stty command, usually the Ctrl-H key sequence.)
Ctrl-D	Deletes the current character.
Esc-D	Deletes the current word.
Esc-Backspace	Deletes the previous word.
Esc-H	Deletes the previous word.
Esc-Delete	Deletes the previous word. If your interrupt character is the Delete key, this command does not work.
Ctrl-T	Transposes the current character with the next character in emacs mode. Transposes the two previous characters in gmacs mode.
Ctrl-C	Capitalizes the current character.
Esc-C	Capitalizes the current word.
Esc-L	Changes the current word to lowercase.
Ctrl-K	Deletes from the cursor to the end of the line. If preceded by a numeric parameter whose value is less than the current cursor position, this editing command deletes from the given position up to the cursor. If preceded by a numeric parameter whose value is greater than the current cursor position, this editing command deletes from the cursor up to the given cursor position.
Ctrl-W	Deletes from the cursor to the mark.

Esc-P	Pushes the region from the cursor to the mark on the stack.
KILL	User-defined kill character as defined by the stty command, usually the Ctrl-G key sequence or @. Kills the entire current line. If two kill characters are entered in succession, all subsequent kill characters cause a line feed (useful when using paper terminals).
Ctrl-Y	Restores the last item removed from the line. (Yanks the item back to the line.)
Ctrl-L	Line feeds and prints the current line.
Ctrl-@	(Null character) Sets a mark.
Esc-space	Sets a mark.
Ctrl-J	(New line) Executes the current line.
Ctrl-M	(Return) Executes the current line.
EOF	Processes the end-of-file character, normally the Ctrl-D key sequence, as an end-of-file only if the current line is null.
Ctrl-P	Fetches the previous command. Each time the Ctrl-P key sequence is entered, the previous command back in time is accessed. Moves back one line when not on the first line of a multiple-line command.
Esc-<	Fetches the least recent (oldest) history line.
Esc->	Fetches the most recent (youngest) history line.
Ctrl-N	Fetches the next command line. Each time the Ctrl-N key sequence is entered, the next command line forward in time is accessed.
Ctrl-R <i>String</i>	Reverses search history for a previous command line containing the string specified by the String parameter. If a value of 0 is given, the search is forward. The specified string is terminated by an Enter or newline character. If the string is preceded by a carat (^), the matched line must begin with the String parameter. If the String parameter is omitted, then the next command line containing the most recent String parameter is accessed. In this case, a value of 0 reverses the direction of the search.
Ctrl-O	(Operate) Executes the current line and fetches the next line relative to the current line from the history file.
Esc <i>Digits</i>	(Escape) Defines the numeric parameter. The digits are taken as a parameter to the next command. The commands that accept a parameter are Ctrl-F , Ctrl-B , ERASE , Ctrl-C , Ctrl-D , Ctrl-K , Ctrl-R , Ctrl-P , Ctrl-N , Ctrl-] , Esc-. , Esc-Ctrl-] , Esc-<u> </u> , Esc-B , Esc-C , Esc-D , Esc-F , Esc-H , Esc-L , and Esc-Ctrl-H .
Esc <i>Letter</i>	(Soft-key) Searches the alias list for an alias named _Letter . If an alias of this name is defined, its value is placed into the input queue. The Letter parameter must not specify one of the escape functions.
Esc-[<i>Letter</i>	(Soft-key) Searches the alias list for an alias named double underscore Letter (__Letter). If an alias of this name is defined, its value is placed into the input queue. This command can be used to program function keys on many terminals.
Esc-.	Inserts on the line the last word of the previous command. If preceded by a numeric parameter, the value of this parameter determines which word to insert rather than the last word.
Esc-<u> </u>	Same as the Esc-. key sequence.
Esc-*	Attempts file name substitution on the current word. An asterisk (*) is appended if the word does not match any file or contain any special pattern characters.

Esc-Esc	File name completion. Replaces the current word with the longest common prefix of all file names that match the current word with an asterisk appended. If the match is unique, a slash (/) is appended if the file is a directory and a space is appended if the file is not a directory.
Esc==	Lists the files that match the current word pattern as if an asterisk (*) were appended.
Ctrl-U	Multiplies the parameter of the next command by 4.
\	Escapes the next character. Editing characters and the ERASE , KILL and INTERRUPT (normally the Delete key) characters can be entered in a command line or in a search string if preceded by a backslash (\). The backslash removes the next character's editing features, if any.
Ctrl-V	Displays the version of the shell.
Esc-#	Inserts a pound sign (#) at the beginning of the line and then executes the line. This causes a comment to be inserted in the history file.

vi editing mode

The vi editing mode has two typing modes.

The modes are:

- **Input mode.** When you enter a command, the vi editor is in input mode.
- **Control mode.** Press the Esc key to enter control mode.

Most control commands accept an optional repeat **Count** parameter prior to the command. When in vi mode on most systems, canonical processing is initially enabled. The command is echoed again if one or more of the following are true:

- The speed is 1200 baud or greater.
- The command contains any control characters.
- Less than one second has elapsed since the prompt was printed.

The Esc character terminates canonical processing for the remainder of the command, and you can then modify the command line. This scheme has the advantages of canonical processing with the type-ahead echoing of raw mode. If the **viraw** option is also set, canonical processing is always disabled. This mode is implicit for systems that do not support two alternate end-of-line delimiters and might be helpful for certain terminals.

Available vi edit commands are grouped into categories. The categories are as follows:

- "Input edit commands"
- "Motion edit commands" on page 477
- "Search edit commands" on page 477
- "Text modification edit commands" on page 478
- "Miscellaneous edit commands" on page 479

Input edit commands

The input edit commands for the Korn shell are described below.

Note: By default, the editor is in input mode.

ERASE	Deletes the previous character. (User-defined erase character as defined by the stty command, usually Ctrl-H or #.)
Ctrl-W	Deletes the previous blank separated word.

Ctrl-D	Terminates the shell.
Ctrl-V	Escapes the next character. Editing characters, such as the ERASE or KILL characters, can be entered in a command line or in a search string if preceded by a Ctrl-V key sequence. The Ctrl-V key sequence removes the next character's editing features (if any).
\	Escapes the next ERASE or KILL character.

Motion edit commands

The motion edit commands for the Korn shell are described below.

Motion edit commands move the cursor as follows:

[Count]l	Moves the cursor forward (right) one character.
[Count]w	Moves the cursor forward one alphanumeric word.
[Count]W	Moves the cursor to the beginning of the next word that follows a blank.
[Count]e	Moves the cursor to the end of the current word.
[Count]E	Moves the cursor to the end of the current blank-separated word.
[Count]h	Moves the cursor backward (left) one character.
[Count]b	Moves the cursor backward one word.
[Count]B	Moves the cursor to the previous blank-separated word.
[Count]l	Moves the cursor to the column specified by the <i>Count</i> parameter.
[Count]fc	Finds the next character <i>c</i> in the current line.
[Count]Fc	Finds the previous character <i>c</i> in the current line.
[Count]tc	Equivalent to f followed by h .
[Count]Tc	Equivalent to F followed by l .
[Count];	Repeats for the number of times specified by the <i>Count</i> parameter the last single-character find command: f , F , t , or T .
[Count],	Reverses the last single-character find command the number of times specified by the <i>Count</i> parameter.
0	Moves the cursor to the start of a line.
^	Moves the cursor to the first nonblank character in a line.
\$	Moves the cursor to the end of a line.

Search edit commands

Search edit commands access your command history as follows:

[Count]k	Fetches the previous command.
[Count]-	Equivalent to the k command.
[Count]j	Fetches the next command. Each time the j command is entered, the next command is accessed.
[Count]+	Equivalent to the j command.
[Count]G	Fetches the command whose number is specified by the <i>Count</i> parameter. The default is the least recent history command.

<i>/String</i>	Searches backward through history for a previous command containing the specified string. The string is terminated by a RETURN or newline character. If the specified string is preceded by a carat (^), the matched line must begin with the <i>String</i> parameter. If the value of the <i>String</i> parameter is null, the previous string is used.
<i>?String</i>	Same as <i>/String</i> except that the search is in the forward direction.
n	Searches for the next match of the last pattern to <i>/String</i> or <i>?String</i> commands.
N	Searches for the next match of the last pattern to <i>/String</i> or <i>?String</i> commands, but in the opposite direction. Searches history for the string entered by the previous <i>/String</i> command.

Text modification edit commands

Text-modification edit commands modify the line as follows:

a	Enters the input mode and enters text after the current character.
A	Appends text to the end of the line. Equivalent to the \$a command.
<i>[Count]cMotion</i> c <i>[Count]Motion</i>	Deletes the current character through the character to which the <i>Motion</i> parameter specifies to move the cursor, and enters input mode. If the value of the <i>Motion</i> parameter is c , the entire line is deleted and the input mode is entered.
C	Deletes the current character through the end of the line and enters input mode. Equivalent to the c\$ command.
S	Equivalent to the cc command.
D	Deletes the current character through the end of line. Equivalent to the d\$ command.
<i>[Count]dMotion</i> d <i>[Count]Motion</i>	Deletes the current character up to and including the character specified by the <i>Motion</i> parameter. If <i>Motion</i> is d , the entire line is deleted.
i	Enters the input mode and inserts text before the current character.
I	Inserts text before the beginning of the line. Equivalent to the Oi command.
<i>[Count]P</i>	Places the previous text modification before the cursor.
<i>[Count]p</i>	Places the previous text modification after the cursor.
R	Enters the input mode and types over the characters on the screen.
<i>[Count]rc</i>	Replaces the number of characters specified by the <i>Count</i> parameter, starting at the current cursor position, with the characters specified by the <i>c</i> parameter. This command also advances the cursor after the characters are replaced.
<i>[Count]x</i>	Deletes the current character.
<i>[Count]X</i>	Deletes the preceding character.
<i>[Count].</i>	Repeats the previous text-modification command.
<i>[Count]~</i>	Inverts the case of the number of characters specified by the <i>Count</i> parameter, starting at the current cursor position, and advances the cursor.
<i>[Count]_</i>	Appends the word specified by the <i>Count</i> parameter of the previous command and enters input mode. The last word is used if the <i>Count</i> parameter is omitted.
*	Appends an asterisk (*) to the current word and attempts file name substitution. If no match is found, it rings the bell. Otherwise, the word is replaced by the matching pattern and input mode is entered.

\	File name completion. Replaces the current word with the longest common prefix of all file names matching the current word with an asterisk (*) appended. If the match is unique, a slash / is appended if the file is a directory. A space is appended if the file is not a directory.
---	---

Miscellaneous edit commands

The most commonly used edit commands include the following:

[<i>Count</i>]y <i>Motion</i>	
y[<i>Count</i>]y <i>Motion</i>	Yanks the current character up to and including the character marked by the cursor position specified by the <i>Motion</i> parameter and puts all of these characters into the delete buffer. The text and cursor are unchanged.
Y	Yanks from the current position to the end of the line. Equivalent to the y\$ command.
u	Undoes the last text-modifying command.
U	Undoes all the text-modifying commands performed on the line.
[<i>Count</i>]v	Returns the command <code>fc -e \${VISUAL:-\${EDITOR:-vi}}</code> <i>Count</i> in the input buffer. If the <i>Count</i> parameter is omitted, then the current line is used.
Ctrl-L	Line feeds and prints the current line. This command is effective only in control mode.
Ctrl-J	(New line) Executes the current line regardless of the mode.
Ctrl-M	(Return) Executes the current line regardless of the mode.
#	Sends the line after inserting a pound sign (#) in front of the line. Useful if you want to insert the current line in the history without executing it. If the command line contains a pipe or semicolon or newline character, then additional pound signs (#) will be inserted in front of each of these symbols. To delete all pound signs, retrieve the command line from history and enter another pound sign (#).
=	Lists the file names that match the current word as if an asterisk were appended to it.
@ <i>Letter</i>	Searches the alias list for an alias named <i>_Letter</i> . If an alias of this name is defined, its value is placed into the input queue for processing.

Enhanced Korn shell (ksh93)

In addition to the default system Korn shell (**/usr/bin/ksh**), AIX provides an enhanced version available as Korn shell **/usr/bin/ksh93**. This enhanced version is mostly upwardly compatible with the current default version, and includes a few additional features that are not available in Korn shell **/usr/bin/ksh**.

Some scripts might perform differently under Korn shell **ksh93** than under the default shell because variable handling is somewhat different under the two shells.

Note: There is also a restricted version of the enhanced Korn shell available, called **rksh93**.

The following features are not available in Korn shell **/usr/bin/ksh**, but are available in Korn shell **/usr/bin/ksh93**:

Arithmetic enhancements	<p>You can use libm functions (math functions typically found in the C programming language), within arithmetic expressions, such as <code>\$ value=\$((sqrt(9)))</code>. More arithmetic operators are available, including the unary <code>+</code>, <code>++</code>, <code>--</code>, and the <code>?:</code> construct (for example, "<code>x ? y : z</code>"), as well as the <code>,</code> (comma) operator. Arithmetic bases are supported up to base 64. Floating point arithmetic is also supported. "typeset -E" (exponential) can be used to specify the number of significant digits and "typeset -F" (float) can be used to specify the number of decimal places for an arithmetic variable. The SECONDS variable now displays to the nearest hundredth of a second, rather than to the nearest second.</p>
Compound variables	<p>Compound variables are supported. A compound variable allows a user to specify multiple values within a single variable name. The values are each assigned with a subscript variable, separated from the parent variable with a period (<code>.</code>). For example:</p> <pre>\$ myvar=(x=1 y=2) \$ print "\${myvar.x}" 1</pre>
Compound assignments	<p>Compound assignments are supported when initializing arrays, both for indexed arrays and associative arrays. The assignment values are placed in parentheses, as shown in the following example:</p> <pre>\$ numbers=(zero one two three) \$ print \${numbers[0]} \${numbers[3]} zero three</pre>
Associative arrays	<p>An associative array is an array with a string as an index.</p> <p>The typeset command used with the -A flag allows you to specify associative arrays within ksh93. For example:</p> <pre>\$ typeset -A teammates \$ teammates=([john]=smith [mary]=jones) \$ print \${teammates[mary]} jones</pre>
Variable name references	<p>The typeset command used with the -n flag allows you to assign one variable name as a reference to another. In this way, modifying the value of a variable will in turn modify the value of the variable that is referenced. For example:</p> <pre>\$ greeting="hello" \$ typeset -n welcome=greeting # establishes the reference \$ welcome="hi there" # overrides previous value \$ print \$greeting hi there</pre>
Parameter expansions	<p>The following parameter-expansion constructs are available:</p> <ul style="list-style-type: none"> • <code>\${!varname}</code> is the name of the variable itself. • <code>\${!varname[@]}</code> names the indexes for the <i>varname</i> array. • <code>\${param:offset}</code> is a substring of <i>param</i>, starting at <i>offset</i>. • <code>\${param:offset:num}</code> is a substring of <i>param</i>, starting at <i>offset</i>, for <i>num</i> number of characters. • <code>\${@:offset}</code> indicates all positional parameters starting at <i>offset</i>. • <code>\${@:offset:num}</code> indicates <i>num</i> positional parameters starting at <i>offset</i>. • <code>\${param/pattern/repl}</code> evaluates to <i>param</i>, with the first occurrence of <i>pattern</i> replaced by <i>repl</i>. • <code>\${param//pattern/repl}</code> evaluates to <i>param</i>, with every occurrence of <i>pattern</i> replaced by <i>repl</i>. • <code>\${param/#pattern/repl}</code> if <i>param</i> begins with <i>pattern</i>, then <i>param</i> is replaced by <i>repl</i>. • <code>\${param/%pattern/repl}</code> if <i>param</i> ends with <i>pattern</i>, then <i>param</i> is replaced by <i>repl</i>.

Discipline functions

A discipline function is a function that is associated with a specific variable. This allows you to define and call a function every time that variable is referenced, set, or unset. These functions take the form of *varname.function*, where *varname* is the name of the variable and *function* is the discipline function. The predefined discipline functions are **get**, **set**, and **unset**.

- The **varname.get** function is invoked every time *varname* is referenced. If the special variable **.sh.value** is set within this function, then the value of *varname* is changed to this value. A simple example is the time of day:

```
$ function time.get
> {
>   .sh.value=$(date +%r)
> }
$ print $time
09:15:58 AM
$ print $time # it will change in a few seconds
09:16:04 AM
```

- The **varname.set** function is invoked every time *varname* is set. The **.sh.value** variable is given the value that was assigned. The value assigned to *varname* is the value of **.sh.value** when the function completes. For example:

```
$ function adder.set
> {
>   let .sh.value="
$ { .sh.value } + 1"
> }
$ adder=0
$ echo $adder
1
$ adder=$adder
$ echo $adder
2
```

- The **varname.unset** function is executed every time *varname* is unset. The variable is not actually unset unless it is unset within the function itself; otherwise it retains its value.

Within all discipline functions, the special variable **.sh.name** is set to the name of the variable, while **.sh.subscript** is set to the value of the variables subscript, if applicable.

Function environments	Functions declared with the <i>function myfunc</i> format are run in a separate function environment and support local variables. Functions declared as <i>myfunc()</i> run with the same environment as the parent shell.
Variables	Variables beginning with .sh. are reserved by the shell and have special meaning. See the description of Discipline Functions in this table for an explanation of .sh.name , .sh.value , and .sh.subscript . Also available is .sh.version , which represents the version of the shell.
Command return values	Return values of commands are as follows: <ul style="list-style-type: none">• If the command to be executed is not found, the return value is set to 127.• If the command to be executed is found, but not executable, the return value is 126.• If the command is executed, but is terminated by a signal, the return value is 256 plus the signal number.
PATH search rules	Special built-in commands are searched for first, followed by all functions (including those in FPATH directories), followed by other built-ins.
Shell history	The hist command allows you to display and edit the shells command history. In the ksh shell, the fc command was used. The fc command is an alias to hist . Variables are <i>HISTCMD</i> , which increments once for each command executed in the shells current history, and <i>HISTEDIT</i> , which specifies which editor to use when using the hist command.

Built-in commands The enhanced Korn shell contains the following built-in commands:

- The **builtin** command lists all available built-in commands.
- The **printf** command works in a similar manner as the **printf()** C library routine. See the **printf** command.
- The **disown** blocks the shell from sending a SIGHUP to the specified command.
- The **getconf** command works in the same way as the stand-alone command **/usr/bin/getconf**. See the **getconf** command.
- The **read** built-in command has the following flags:
 - **read -d {char}** allows you to specify a character delimiter instead of the default newline.
 - **read -t {seconds}** allows you to specify a time limit, in seconds, after which the **read** command will time out. If **read** times out, it will return FALSE.
- The **exec** built-in command has the following flags:
 - **exec -a {name} {cmd}** specifies that argument 0 of *cmd* be replaced with *name*.
 - **exec -c {cmd}** tells **exec** to clear the environment before executing *cmd*.
- The **kill** built-in command has the following flags:
 - **kill -n {signum}** is used for specifying a signal number to send to a process, while **kill -s {signame}** is used to specify a signal name.
 - **kill -l**, with no arguments, lists all signal names but not their numbers.
- The **whence** built-in command has the following flags:
 - The **-a** flag displays all matches, not only the first one found.
 - The **-f** flag tells **whence** not to search for any functions.
- An escape character sequence is used for use by the **print** and **echo** commands. The Esc (Escape) key can be represented by the sequence **\E**.
- All regular built-in commands recognize the **-?** flag, which shows the syntax for the specified command.

Other miscellaneous differences between Korn shell ksh and Korn shell ksh93

Other differences are:

- With Korn shell ksh93, you cannot export functions using the **typeset -fx** built-in command.
- With Korn shell ksh93, you cannot export an alias using the **alias -x** built-in command.
- With Korn shell ksh93, a dollar sign followed by a single quote (**\$'**) is interpreted as an ANSI C string. You must quote the dollar sign (**\"\$'**) to get the old (ksh) behavior.
- Argument parsing logic for Korn shell ksh93 built-in commands has been changed. The undocumented combinations of argument parsing to Korn shell ksh built-in commands do not work in Korn shell ksh93. For example, **typeset -4i** works similar to **typeset -i4** in Korn shell ksh, but does not work in Korn shell ksh93.
- With Korn shell ksh93, command substitution and arithmetic expansion is performed on special environment variables PS1, PS3, and ENV while expanding. Therefore, you must escape the grave symbol (**`**) and the dollar sign and opening parenthesis symbols (**\$()**) using a backslash (****) to retain the old behavior. For example, Korn shell ksh literally assigns **x=\${name}\operator'** as **\$name\operator**; Korn shell ksh93 expands **\t** and assigns it as **name<\t expanded>operator**. To preserve the Korn shell ksh behavior, you must quote **\$**. For example, **x="`\${name}\operator'**.
- The **ERRNO** variable has been removed in Korn shell ksh93.
- In Korn shell ksh93, file names are not expanded for non-interactive shells after the redirection symbol.
- With Korn shell ksh93, you must use the **-t** option of the **alias** command to display tracked aliases.
- With Korn shell ksh93, in emacs mode, **Ctrl+T** swaps the current and previous character. With ksh, **Ctrl+T** swaps the current and next character.
- Korn shell ksh93 does not allow unbalanced parentheses within **\${name operator value}**. For example, **\${name-({}** needs an escape such as **\${name-\({}** to work in both versions.
- With Korn shell ksh93, the **kill -l** command lists only the signal names, not their numerical values.

Bourne shell

The Bourne shell is an interactive command interpreter and command programming language.

The **bsh** command runs the Bourne shell.

The Bourne shell can be run either as a login shell or as a subshell under the login shell. Only the **login** command can call the Bourne shell as a login shell. It does this by using a special form of the **bsh** command name: `-bsh`. When called with an initial hyphen (-), the shell first reads and runs commands found in the system `/etc/profile` file and your `$HOME/.profile`, if one exists. The `/etc/profile` file sets variables needed by all users. Finally, the shell is ready to read commands from your standard input.

If the **File** [*Parameter*] parameter is specified when the Bourne shell is started, the shell runs the script file identified by the **File** parameter, including any parameters specified. The script file specified must have read permission; any `setuid` and `setgid` settings are ignored. The shell then reads the commands. If either the **-c** or **-s** flag is used, do not specify a script.

Bourne shell environment

All variables (with their associated values) known to a command at the beginning of its execution constitute its *environment*. This environment includes variables that a command inherits from its parent process and variables specified as keyword parameters on the command line that calls the command.

The shell passes to its child processes the variables named as arguments to the built-in **export** command. This command places the named variables in the environments of both the shell and all its future child processes.

Keyword parameters are variable-value pairs that appear in the form of assignments, normally before the procedure name on a command line (but see also the flag for the **set** command). These variables are placed in the environment of the procedure being called.

See the following examples:

- Consider the following procedure, which displays the values of two variables (saved in a command file named **key_command**):

```
# key_command
echo $a $b
```

The following command lines produce the output shown:

Input	Output
<code>a=key1 b=key2 key_command</code>	<code>key1 key2</code>
<code>a=tom b=john key_command</code>	<code>tom john</code>

A procedure's keyword parameters are not included in the parameter count stored in `$#`.

A procedure can access the values of any variables in its environment. If it changes any of these values, however, the changes are not reflected in the shell environment. The changes are local to the procedure in question. To place the changes in the environment that the procedure passes to its child processes, you must export the new values within that procedure.

See the following examples:

- To obtain a list of variables that are exportable from the current shell, type the following:

```
export
```

- To obtain a list of read-only variables from the current shell, type the following:

```
readonly
```

- To obtain a list of variable-value pairs in the current environment, type the following:

```
env
```

For more information about user environments, see “/etc/environment file” on page 425.

Restricted shell

The restricted shell is used to set up login names and execution environments whose capabilities are more controlled than those of the regular Bourne shell.

The **Rsh** or **bsh -r** command opens the restricted shell. The behavior of these commands is identical to those of the **bsh** command, except that the following actions are not allowed:

- Changing the directory (with the **cd** command)
- Setting the value of *PATH* or *SHELL* variables
- Specifying path or command names containing a slash (/)
- Redirecting output

If the restricted shell determines that a command to be run is a shell procedure, it uses the Bourne shell to run the command. In this way, it is possible to provide a user with shell procedures that access the full power of the Bourne shell while imposing a limited menu of commands. This situation assumes that the user does not have write and execute permissions in the same directory.

If the *File* [*Parameter*] parameter is specified when the Bourne shell is started, the shell runs the script file identified by the *File* parameter, including any parameters specified. The script file specified must have read permission. Any **setuid** and **setgid** settings for script files are ignored. The shell then reads the commands. If either the **-c** or **-s** flag is used, do not specify a script file.

When started with the **Rsh** command, the shell enforces restrictions after interpreting the **.profile** and **/etc/environment** files. Therefore, the writer of the **.profile** file has complete control over user actions by performing setup actions and leaving the user in an appropriate directory (probably not the login directory). An administrator can create a directory of commands in the **/usr/rbin** directory that the **Rsh** command can use by changing the *PATH* variable to contain the directory. If it is started with the **bsh -r** command, the shell applies restrictions when interpreting the **.profile** files.

When called with the name **Rsh**, the restricted shell reads the user's **.profile** file (**\$HOME/.profile**). It acts as the regular Bourne shell while doing this, except that an interrupt causes an immediate exit instead of a return to command level.

Restricted Korn shell

The Restricted Korn Shell is used to set up login names and execution environments whose capabilities are more controlled than those of the regular Korn shell.

The **rksh** or **ksh -r** command opens the Restricted Korn Shell. The behavior of these commands is identical to those of the **ksh** command, except that the following actions are not allowed:

- Change the current working directory
- Set the value of the *SHELL*, *ENV*, or *PATH* variables
- Specify the pathname of a command containing a / (slash)
- Redirect output of a command with > (right caret), >| (right caret, pipe symbol), <> (left caret, right caret), or >> (two right carets).

If the Restricted Korn Shell determines that a command to be run is a shell procedure, it uses the Korn shell to run the command. In this way, it is possible to provide an end user with shell procedures that

access the full power of the Korn shell while imposing a limited menu of commands. This situation assumes that the user does not have write and execute permissions in the same directory.

If the **File** [*Parameter*] parameter is specified when the Korn shell is started, the shell runs the script file identified by the **File** parameter, including any parameters specified. The script file specified must have read permission. Any **setuid** and **setgid** settings for script files are ignored. The shell then reads the commands. If either the **-c** or **-s** flag is used, do not specify a script file.

When started with the **rksh** command, the shell enforces restrictions after interpreting the **.profile** and **/etc/environment** files. Therefore, the writer of the **.profile** file has complete control over user actions by performing setup actions and leaving the user in an appropriate directory (probably not the login directory). An administrator can create a directory of commands in the **/usr/rbin** directory that the **rksh** command can use by changing the **PATH** variable to contain the directory. If it is started with the **ksh -r** command, the shell applies restrictions when interpreting the **.profile** files.

When called with the **rksh** command, the Restricted Korn Shell reads the user's **.profile** file (**\$HOME/.profile**). It acts as the regular Korn shell while doing this, except that an interrupt causes an immediate exit instead of a return to command level.

Bourne shell commands

You can issue commands in the Bourne shell.

When you issue a command in the Bourne shell, it first evaluates the command and makes all indicated substitutions. It then runs the command provided that:

- The command name is a Bourne shell special built-in command.
OR
- The command name matches the name of a defined function. If this is the case, the shell sets the positional parameters to the parameters of the function.

If the command name matches neither a built-in command nor the name of a defined function and the command names an executable file that is a compiled (binary) program, the shell (as *parent*) spawns a new (*child*) process that immediately runs the program. If the file is marked executable but is not a compiled program, the shell assumes that it is a shell procedure. In this case, the shell spawns another instance of itself (a *subshell*), to read the file and execute the commands included in it. The shell also runs a parenthesized command in a subshell. To the user, a compiled program is run in exactly the same way as a shell procedure. The shell normally searches for commands in file system directories in this order:

1. **/usr/bin**
2. **/etc**
3. **/usr/sbin**
4. **/usr/ucb**
5. **\$HOME/bin**
6. **/usr/bin/X11**
7. **/sbin**
8. Current directory

The shell searches each directory, in turn, continuing with the next directory if it fails to find the command.

Note: The **PATH** variable determines the order in which the shell searches directories. You can change the particular sequence of directories searched by resetting the **PATH** variable.

If you give a specific path name when you run a command (for example, `/usr/bin/sort`), the shell does not search any directories other than the one you specify. If the command name contains a slash (`/`), the shell does not use the search path.

You can give a full path name that begins with the root directory (such as `/usr/bin/sort`). You can also specify a path name relative to the current directory. If you specify, for example:

```
bin/myfile
```

the shell looks in the current directory for a directory named **bin** and in that directory for the file **myfile**.

Note: The restricted shell does not run commands containing a slash (`/`).

The shell remembers the location in the search path of each executed command (to avoid unnecessary **exec** commands later). If it finds the command in a relative directory (one whose name does not begin with `/`), the shell must redetermine the command's location whenever the current directory changes. The shell forgets all remembered locations each time you change the *PATH* variable or run the **hash -r** command.

Character quotation

Many characters have a special meaning to the shell. Sometimes you want to conceal that meaning. Single (`'`) and double (`"`) quotation marks surrounding a string, or a backslash (`\`) before a single character allow you to conceal the character's meaning.

All characters (except the enclosing single quotation marks) are taken literally, with any special meaning removed. Thus, the command:

```
stuff='echo $? $*; ls * | wc'
```

assigns the literal string `echo $? $*; ls * | wc` to the variable `stuff`. The shell does not execute the **echo**, **ls**, and **wc** commands or expand the `?` and `$*` variables and the asterisk (`*`) special character.

Within double quotation marks, the special meaning of the dollar sign (`$`), backquote (```), and double quotation (`"`) characters remains in effect, while all other characters are taken literally. Thus, within double quotation marks, command and variable substitution takes place. In addition, the quotation marks do not affect the commands within a command substitution that is part of the quoted string, so characters there retain their special meanings.

Consider the following sequence:

```
ls *
file1 file2 file3
message="This directory contains `ls * ` "
echo $message
This directory contains file1 file2 file3
```

This shows that the asterisk (`*`) special character inside the command substitution was expanded.

To hide the special meaning of the dollar sign (`$`), backquote (```), and double quotation (`"`) characters within double quotation marks, precede these characters with a backslash (`\`). When you do not use double quotation marks, preceding a character with a backslash is equivalent to placing it within single quotation marks. Therefore, a backslash immediately preceding a newline character (that is, a backslash at the end of the line) hides the newline character and allows you to continue the command line on the next physical line.

Signal handling

The shell ignores **INTERRUPT** and **QUIT** signals for an invoked command if the command is terminated with an ampersand (&); that is, if it is running in the background. Otherwise, signals have the values inherited by the shell from its parent, with the exception of the **SEGMENTATION VIOLATION** signal.

For more information, see the Bourne shell built-in **trap** command.

Bourne shell compound commands

A compound command is one of the following.

- Pipeline (one or more simple commands separated by the pipe (|) symbol)
- List of simple commands
- Command beginning with a reserved word
- Command beginning with the control operator left parenthesis (())

Unless otherwise stated, the value returned by a compound command is that of the last simple command executed.

Reserved words

The following reserved words for the Bourne shell are recognized only when they appear without quotation marks as the first word of a command.

for	do	done
case	esac	
if	then	fi
elif	else	
while	until	
{	}	
()	

for <i>Identifier</i> [in <i>Word</i> . . .]	Sets the <i>Identifier</i> parameter to the word or words specified by the <i>Word</i> parameter (one at a time) and runs the commands specified in the <i>List</i> parameter. If you omit in <i>Word</i> . . ., then the for command runs the <i>List</i> parameter for each positional parameter that is set, and processing ends when all positional parameters have been used.
---	--

case <i>Word</i> in <i>Pattern</i> [<i>Pattern</i> . . .) <i>List</i> ; [<i>Pattern</i> [<i>Pattern</i> . . .) <i>List</i> ;] . . . esac	Runs the commands specified in the <i>List</i> parameter that are associated with the first <i>Pattern</i> parameter that matches the value of the <i>Word</i> parameter. Uses the same character-matching notation in patterns that are used for file name substitution, except that a slash (/), leading dot (.), or a dot immediately following a slash (/.) do not need to match explicitly.
---	--

if <i>List</i> then <i>List</i> [elif <i>List</i> then <i>List</i>] . . . [else <i>List</i>] fi	Runs the commands specified in the <i>List</i> parameter following the if command. If the command returns a zero exit value, the shell runs the <i>List</i> parameter following the first then command. Otherwise, it runs the <i>List</i> parameter following the elif command (if it exists). If this exit value is zero, the shell runs the <i>List</i> parameter following the next then command. If the command returns a nonzero exit value, the shell runs the <i>List</i> parameter following the else command (if it exists). If no else <i>List</i> or then <i>List</i> is performed, the if command returns a zero exit value.
--	---

while <i>List</i> do <i>List</i> done	Runs the commands specified in the <i>List</i> parameter following the while command. If the exit value of the last command in the while <i>List</i> is zero, the shell runs the <i>List</i> parameter following the do command. It continues looping through the lists until the exit value of the last command in the while <i>List</i> is nonzero. If no commands in the do <i>List</i> are performed, the while command returns a zero exit value.
--	--

until <i>List</i> do <i>List</i> done	Runs the commands specified in the <i>List</i> parameter following the until command. If the exit value of the last command in the until <i>List</i> is nonzero, runs the <i>List</i> following the do command. Continues looping through the lists until the exit value of the last command in the until <i>List</i> is zero. If no commands in the do <i>List</i> are performed, the until command returns a zero exit value.
(<i>List</i>)	Runs the commands in the <i>List</i> parameter in a subshell.
{ <i>List</i> ; }	Runs the commands in the <i>List</i> parameter in the current shell process and does not start a subshell.
<i>Name</i> () { <i>List</i> }	Defines a function that is referenced by the <i>Name</i> parameter. The body of the function is the list of commands between the braces specified by the <i>List</i> parameter.

Bourne shell built-in commands

Special commands are built into the Bourne shell and run in the shell process.

Unless otherwise indicated, output is written to file descriptor 1 (standard output) and the exit status is 0 (zero) if the command does not contain any syntax errors. Input and output redirection is permitted.

See “List of Bourne shell built-in commands” on page 498 for an alphabetical listing of these commands.

The following special commands are treated somewhat differently from other special built-in commands:

:	(colon)	exec	shift
.	(dot)	exit	times
break		export	trap
continue		readonly	wait
eval		return	

The Bourne shell processes these commands as follows:

- Keyword parameter assignment lists preceding the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors in a shell script cause the script to stop processing.

Special command descriptions

The Bourne shell provides the following special built-in commands.

:	Returns a zero exit value.
. <i>File</i>	Reads and runs commands from the <i>File</i> parameter and returns. Does not start a subshell. The shell uses the search path specified by the <i>PATH</i> variable to find the directory containing the specified file.
break [<i>n</i>]	Exits from the enclosing for , while , or until command loops, if any. If you specify the <i>n</i> variable, the break command breaks the number of levels specified by the <i>n</i> variable.
continue [<i>n</i>]	Resumes the next iteration of the enclosing for , while , or until command loops. If you specify the <i>n</i> variable, the command resumes at the <i>n</i> th enclosing loop.

cd <i>Directory</i>]	Changes the current directory to <i>Directory</i> . If you do not specify <i>Directory</i> , the value of the <i>HOME</i> shell variable is used. The <i>CDPATH</i> shell variable defines the search path for <i>Directory</i> . <i>CDPATH</i> is a colon-separated list of alternative directory names. A null path name specifies the current directory (which is the default path). This null path name appears immediately after the equal sign in the assignment or between the colon delimiters anywhere else in the path list. If <i>Directory</i> begins with a slash (/), the shell does not use the search path. Otherwise, the shell searches each directory in the <i>CDPATH</i> shell variable. Note: The restricted shell cannot run the cd shell command.
echo <i>String</i> . . .]	Writes character strings to standard output. See the echo command for usage and parameter information. The -n flag is not supported.
eval [<i>Argument</i> . . .]	Reads arguments as input to the shell and runs the resulting command or commands.
exec [<i>Argument</i> . . .]	Runs the command specified by the <i>Argument</i> parameter in place of this shell without creating a new process. Input and output arguments can appear, and if no other arguments appear, cause the shell input or output to be modified. This is not recommended for your login shell.
exit [<i>n</i>]	Causes a shell to exit with the exit value specified by the <i>n</i> parameter. If you omit this parameter, the exit value is that of the last command executed (the Ctrl-D key sequence also causes a shell to exit). The value of the <i>n</i> parameter can be from 0 to 255, inclusive.
export [<i>Name</i> . . .]	Marks the specified names for automatic export to the environments of subsequently executed commands. If you do not specify the <i>Name</i> parameter, the export command displays a list of all names that are exported in this shell. You cannot export function names.
hash [-r] [<i>Command</i> . . .]	Finds and remembers the location in the search path of each <i>Command</i> specified. The -r flag causes the shell to forget all locations. If you do not specify the flag or any commands, the shell displays information about the remembered commands in the following format: Hits Cost Command Hits indicates the number of times a command has been run by the shell process. Cost is a measure of the work required to locate a command in the search path. Command shows the path names of each specified command. Certain situations require that the stored location of a command be recalculated; for example, the location of a relative path name when the current directory changes. Commands for which that might be done are indicated by an asterisk (*) next to the Hits information. Cost is incremented when the recalculation is done.
pwd	Displays the current directory. See the pwd command for a discussion of command options.
read [<i>Name</i> . . .]	Reads one line from standard input. Assigns the first word in the line to the first <i>Name</i> parameter, the second word to the second <i>Name</i> parameter, and so on, with leftover words assigned to the last <i>Name</i> parameter. This command returns a value of 0 unless it encounters an end-of-file character.
readonly [<i>Name</i> . . .]	Marks the name specified by the <i>Name</i> parameter as read-only. The value of the name cannot be reset. If you do not specify any <i>Name</i> , the readonly command displays a list of all read-only names.
return [<i>n</i>]	Causes a function to exit with a return value of <i>n</i> . If you do not specify the <i>n</i> variable, the function returns the status of the last command performed in that function. This command is valid only when run within a shell function.

set [<i>Flag</i> [<i>Argument</i>] . . .]	Sets one or more of the following flags:
-a	Marks for export all variables to which an assignment is performed. If the assignment precedes a command name, the export attribute is effective only for that command execution environment, except when the assignment precedes one of the special built-in commands. In this case, the export attribute persists after the built-in command has completed. If the assignment does not precede a command name, or if the assignment is a result of the operation of the getopts or read commands, the export attribute persists until the variable is unset.
-e	Exits immediately if all of the following conditions exist for a command: <ul style="list-style-type: none"> • It exits with a return value greater than 0 (zero). • It is not part of the compound list of a while, until, or if command. • It is not being tested using AND or OR lists. • It is not a pipeline preceded by the ! (exclamation point) reserved word.
-f	Disables file name substitution.
-h	Locates and remembers the commands called within functions as the functions are defined. (Normally, these commands are located when the function is performed; see the hash command.)
-k	Places all keyword parameters in the environment for a command, not just those preceding the command name.
-n	Reads commands but does not run them. To check for shell script syntax errors, use the -n flag.
-t	Exits after reading and executing one command.
-u	Treats an unset variable as an error and immediately exits when performing variable substitution. An interactive shell does not exit.
-v	Displays shell input lines as they are read.
-x	Displays commands and their arguments before they are run.
—	Does not change any of the flags. This is useful in setting the \$1 positional parameter to a string beginning with a hyphen (-).

Using a plus sign (+) rather than a hyphen (-) unsets flags. You can also specify these flags on the shell command line. The **\$-** special variable contains the current set of flags.

Any *Argument* to the **set** command becomes a positional parameter and is assigned, in order, to **\$1**, **\$2**, . . . , and so on. If you do not specify a *flag* or *Argument*, the **set** command displays all the names and values of the current shell variables.

shift [<i>n</i>]	Shifts command line arguments to the left; that is, reassigns the value of the positional parameters by discarding the current value of \$1 and assigning the value of \$2 to \$1 , of \$3 to \$2 , and so on. If there are more than 9 command line arguments, the 10 th is assigned to \$9 and any that remain are still unassigned (until after another shift). If there are 9 or fewer arguments, the shift command unsets the highest-numbered positional parameter that has a value.
	The \$0 positional parameter is never shifted. The shift <i>n</i> command is a shorthand notation specifying <i>n</i> number of consecutive shifts. The default value of the <i>n</i> parameter is 1.
test <i>Expression</i> [<i>Expression</i>]	Evaluates conditional expressions. See the test command for a discussion of command flags and parameters. The -h flag is not supported by the built-in test command in bsh .
times	Displays the accumulated user and system times for processes run from the shell.

trap [<i>Command</i>] [<i>n</i>] . . .	<p>Runs the command specified by the <i>Command</i> parameter when the shell receives the signal or signals specified by the <i>n</i> parameter. The trap commands are run in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective.</p> <p>Note: The shell scans the <i>Command</i> parameter once when the trap is set and again when the trap is taken. If you do not specify a command, then all traps specified by the <i>n</i> parameter are reset to their current values. If you specify a null string, this signal is ignored by the shell and by the commands it invokes. If the <i>n</i> parameter is zero (0), the specified command is run when you exit from the shell. If you do not specify either a command or a signal, the trap command displays a list of commands associated with each signal number.</p>
type [<i>Name</i> . . .]	Indicates how the shell would interpret it as a command name for each <i>Name</i> specified.
ulimit [-HS] [-c -d -f -m -s -t] [<i>limit</i>]	<p>Displays or adjusts allocated shell resources. The shell resource settings can be displayed either individually or as a group. The default mode is to display resources set to the soft setting, or the lower bound, as a group.</p> <p>The setting of shell resources depends on the effective user ID of the current shell. The hard level of a resource can be set only if the effective user ID of the current shell is root. You will get an error if you are not root user and you are attempting to set the hard level of a resource. By default, the root user sets both the hard and soft limits of a particular resource. The root user should therefore be careful in using the -S, -H, or default flag usage of limit settings. Unless you are a root user, you can set only the soft limit of a resource. After a limit has been decreased by a nonroot user, it cannot be increased, even back to the original system limit.</p> <p>To set a resource limit, select the appropriate flag and the limit value of the new resource, which should be an integer. You can set only one resource limit at a time. If more than one resource flag is specified, you receive undefined results. By default, ulimit with only a new value on the command line sets the file size of the shell. Use of the -f flag is optional.</p> <p>You can specify the following ulimit command flags:</p> <ul style="list-style-type: none"> -c Sets or displays core segment for shell. -d Sets or displays data segment for shell. -f Sets or displays file size for shell. -H Sets or displays hard resource limit (root user only). -m Sets or displays memory for shell. -s Sets or displays stack segment for shell. -S Sets or displays soft resource limit. -t Sets or displays CPU time maximum for shell.
umask [<i>nnn</i>]	Determines file permissions. This value, along with the permissions of the creating process, determines a file's permissions when the file is created. The default is 022. When no value is entered, umask displays the current value.
unset [<i>Name</i> . . .]	Removes the corresponding variable or function for each name specified by the <i>Name</i> parameter. The <i>PATH</i> , <i>PS1</i> , <i>PS2</i> , <i>MAILCHECK</i> , and <i>IFS</i> shell variables cannot be unset.
wait [<i>n</i>]	Waits for the child process whose process number is specified by the <i>n</i> parameter to exit and then returns the exit status of that process. If you do not specify the <i>n</i> parameter, the shell waits for all currently active child processes, and the return value is 0.

Command substitution in the Bourne shell

Command substitution allows you to capture the output of any command as an argument to another command.

When you place a command line within backquotes (```), the shell first runs the command or commands and then replaces the entire expression, including the backquotes, with the output. This feature is often used to give values to shell variables. For example, the statement:

```
today=`date`
```

assigns the string representing the current date to the *today* variable. The following assignment saves, in the *files* variable, the number of files in the current directory:

```
files=`ls | wc -l`
```

You can perform command substitution on any command that writes to standard output.

To nest command substitutions, precede each of the nested backquotes with a backslash (`\`), as in:

```
logmsg=`echo Your login directory is \`pwd\``
```

You can also give values to shell variables indirectly by using the **read** special command. This command takes a line from standard input (usually your keyboard) and assigns consecutive words on that line to any variables named. For example:

```
read first init last
```

takes an input line of the form:

```
J. Q. Public
```

and has the same effect as if you had typed:

```
first=J. init=Q. last=Public
```

The **read** special command assigns any excess words to the last variable.

Variable substitution in the Bourne shell

The Bourne shell permits you to perform variable substitutions.

The Bourne shell has several mechanisms for creating variables (assigning a string value to a name). Certain variables, positional parameters and keyword parameters are normally set only on a command line. Other variables are simply names to which you or the shell can assign string values.

User-defined variables in the Bourne shell

The Bourne shell recognizes alphanumeric variables to which string values can be assigned.

To assign a string value to a name, type the following:

```
Name=String
```

A name is a sequence of letters, digits, and underscores that begins with an underscore or a letter. To use the value that you have assigned to a variable, add a dollar sign (`$`) to the beginning of its name. Thus, the *\$Name* variable yields the value specified by the *String* variable. Note that no spaces are on either side of the equal sign (`=`) in an assignment statement. (Positional parameters cannot appear in an assignment statement. They can be set only as described in “Positional parameters in the Bourne shell” on page 496.) You can put more than one assignment on a command line, but remember that the shell performs the assignments from right to left.

If you enclose the *String* variable with double or single quotation marks (`"` or `'`), the shell does not treat blanks, tabs, semicolons, and newline characters within the string as word delimiters, but it imbeds them literally in the string.

If you enclose the *String* variable with double quotation marks ("), the shell still recognizes variable names in the string and performs variable substitution; that is, it replaces references to positional parameters and other variable names that are prefaced by dollar sign (\$) with their corresponding values, if any. The shell also performs command substitution within strings that are enclosed in double quotation marks.

If you enclose the *String* variable with single quotation marks ('), the shell does not substitute variables or commands within the string. The following sequence illustrates this difference:

```
You:          num=875
              number1="Add $num"
              number2='Add $num'
              echo $number1
System:       Add 875
You:          echo $number2
System:       Add $num
```

The shell does not reinterpret blanks in assignments after variable substitution. Thus, the following assignments result in \$first and \$second having the same value:

```
first='a string with embedded blanks'
second=$first
```

When you reference a variable, you can enclose the variable name (or the digit designating a positional parameter) in braces { } to delimit the variable name from any string following. In particular, if the character immediately following the name is a letter, digit, or underscore, and the variable is not a positional parameter, then the braces are required:

```
You:          a='This is a'
              echo "${a}n example"
System:       This is an example
You:          echo "$a test"
System:       This is a test
```

See “Conditional substitution in the Bourne shell” on page 495 for a different use of braces in variable substitutions.

Variables used by the Bourne shell

The shell uses the following variables. Although the shell sets some of them, you can set or reset all of them.

<i>CDPATH</i>	Specifies the search path for the cd (change directory) command.
<i>HOME</i>	Indicates the name of your <i>login directory</i> , which is the directory that becomes the current directory upon completion of a login. The login program initializes this variable. The cd command uses the value of the <i>\$HOME</i> variable as its default value. Using this variable rather than an explicit path name in a shell procedure allows the procedure to be run from a different directory without alterations.
<i>IFS</i>	The characters that are IFS (internal field separators), which are the characters that the shell uses during blank interpretation. See “Blank interpretation” on page 495. The shell initially sets the <i>IFS</i> variable to include the blank, tab, and newline characters.
<i>LANG</i>	Determines the locale to use for the locale categories when both the <i>LC_ALL</i> variable and the corresponding environment variable (beginning with <i>LC_</i>) do not specify a locale. For more information about locales, see <i>Locale Overview in AIX 5L Version 5.3 National Language Support Guide and Reference</i> .
<i>LC_ALL</i>	Determines the locale to be used to override any values for locale categories specified by the settings of the <i>LANG</i> environment variable or any environment variables beginning with <i>LC_</i> .
<i>LC_COLLATE</i>	Defines the collating sequence to use when sorting names and when character ranges occur in patterns.

<i>LC_CTYPE</i>	Determines the locale for the interpretation of sequences of bytes of text data as characters (that is, single versus multibyte characters in arguments and input files), which characters are defined as letters (alpha character class), and the behavior of character classes within pattern matching.
<i>LC_MESSAGES</i>	Determines the language in which messages should be written.
<i>LIBPATH</i>	Specifies the search path for shared libraries.
<i>LOGNAME</i>	Specifies your login name, marked readonly in the /etc/profile file.
<i>MAIL</i>	Indicates the path name of the file used by the mail system to detect the arrival of new mail. If this variable is set, the shell periodically checks the modification time of this file and displays the value of \$MAILMSG if the time changes and the length of the file is greater than 0. Set the <i>MAIL</i> variable in the .profile file. The value normally assigned to it by users of the mail command is /usr/spool/mail/\$LOGNAME .
<i>MAILCHECK</i>	The number of seconds that the shell lets elapse before checking again for the arrival of mail in the files specified by the <i>MAILPATH</i> or <i>MAIL</i> variables. The default value is 600 seconds (10 minutes). If you set the <i>MAILCHECK</i> variable to 0, the shell checks before each prompt.
<i>MAILMSG</i>	The mail notification message. If you explicitly set the <i>MAILMSG</i> variable to a null string (<i>MAILMSG=""</i>), no message is displayed.
<i>MAILPATH</i>	A list of file names separated by colons. If this variable is set, the shell informs you of the arrival of mail in any of the files specified in the list. You can follow each file name by a % and a message to be displayed when mail arrives. Otherwise, the shell uses the value of the <i>MAILMSG</i> variable or, by default, the message [YOU HAVE NEW MAIL]. Note: When the <i>MAILPATH</i> variable is set, these files are checked instead of the file set by the <i>MAIL</i> variable. To check the files set by the <i>MAILPATH</i> variable and the file set by the <i>MAIL</i> variable, specify the MAIL file in your list of MAILPATH files.
<i>PATH</i>	The search path for commands, which is an ordered list of directory path names separated by colons. The shell searches these directories in the specified order when it looks for commands. A null string anywhere in the list represents the current directory. The <i>PATH</i> variable is normally initialized in the /etc/environment file, usually to /usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin . You can reset this variable to suit your own needs. The <i>PATH</i> variable provided in your .profile file also includes \$HOME/bin and your current directory. If you have a project-specific directory of commands, for example, /project/bin , that you want searched before the standard system directories, set your <i>PATH</i> variable as follows: <pre>PATH=/project/bin:\$PATH</pre> The best place to set your <i>PATH</i> variable to a value other than the default value is in your \$HOME/.profile file. You cannot reset the <i>PATH</i> variable if you are executing commands under the restricted shell.
<i>PS1</i>	The string to be used as the primary system prompt. An interactive shell displays this prompt string when it expects input. The default value of the <i>PS1</i> variable is \$ followed by a blank space for nonroot users.
<i>PS2</i>	The value of the secondary prompt string. If the shell expects more input when it encounters a newline character in its input, it prompts with the value of the <i>PS2</i> variable. The default value of the <i>PS2</i> variable is > followed by a blank space.
<i>SHACCT</i>	The name of a file that you own. If this variable is set, the shell writes an accounting record in the file for each shell script executed. You can use accounting programs such as acctcom and acctcms to analyze the data collected.
<i>SHELL</i>	The path name of the shell, which is kept in the environment. This variable should be set and exported by the \$HOME/.profile file of each restricted login.

TIMEOUT	The number of minutes a shell remains inactive before it exits. If this variable is set to a value greater than zero (0), the shell exits if a command is not entered within the prescribed number of seconds after issuing the PS1 prompt. (Note that the shell can be compiled with a maximum boundary that cannot be exceeded for this value.) A value of zero indicates no time limit.
----------------	--

Predefined special variables in the Bourne shell

Several variables have special meanings. The following variables are set only by the Bourne shell:

\$@	Expands the positional parameters, beginning with \$1 . Each parameter is separated by a space. If you place double quotation marks (" ") around \$@ , the shell considers each positional parameter a separate string. If no positional parameters exist, the Bourne shell expands the statement to an unquoted null string.
\$*	Expands the positional parameters, beginning with \$1 . The shell separates each parameter with the first character of the <i>IFS</i> variable value. If you place double quotation marks (" ") around \$* , the shell includes the positional parameter values, in double quotation marks. Each value is separated by the first character of the <i>IFS</i> variable.
 \$#	Specifies the number of positional parameters passed to the shell, not counting the name of the shell procedure itself. The \$# variable thus yields the number of the highest-numbered positional parameter that is set. One of the primary uses of this variable is to check for the presence of the required number of arguments. Only positional parameters \$0 through \$9 are accessible through the shell. See "Positional parameters in the Bourne shell" on page 496 for more information.
 \$?	Specifies the exit value of the last command executed. Its value is a decimal string. Most commands return a value of 0 to indicate successful completion. The shell itself returns the current value of the \$? variable as its exit value.
 \$\$	Identifies the process number of the current process. Because process numbers are unique among all existing processes, this string is often used to generate unique names for temporary files. The following example illustrates the recommended practice of creating temporary files in a directory used only for that purpose: <pre>temp=/tmp/\$\$ ls >\$temp . . . rm \$temp</pre>
 \$!	Specifies the process number of the last process run in the background using the & terminator.
 \$-	A string consisting of the names of the execution flags currently set in the shell.

Blank interpretation

After the shell performs variable and command substitution, it scans the results for internal field separators (those defined in the *IFS* shell variable).

The shell splits the line into distinct words at each place it finds one or more of these characters separating each distinct word with a single space. It then retains explicit null arguments (" " or ' ') and discards implicit null arguments (those resulting from parameters that have no values).

Conditional substitution in the Bourne shell

Normally, the shell replaces the expression **\$Variable** with the string value assigned to the *Variable* variable, if there is one. However, there is a special notation that allows *conditional substitution*, depending on whether the variable is set or not null, or both.

By definition, a variable is set if it has ever been assigned a value. The value of a variable can be the null string, which you can assign to a variable in any one of the following ways:

```
A=
bcd=""
Efg=' '           Assigns the null string to the A, bcd, and Efg.
set ' ' ""       Sets the first and second positional parameters to the null string and unsets all other positional
                  parameters.
```

The following is a list of the available expressions you can use to perform conditional substitution:

<code>\${Variable-String}</code>	If the variable is set, substitute the <i>Variable</i> value in place of this expression. Otherwise, replace this expression with the <i>String</i> value.
<code>\${Variable:-String}</code>	If the variable is set and not null, substitute the <i>Variable</i> value in place of this expression. Otherwise, replace this expression with the <i>String</i> value.
<code>\${Variable=String}</code>	If the variable is set, substitute the <i>Variable</i> value in place of this expression. Otherwise, set the <i>Variable</i> value to the <i>String</i> value and then substitute the <i>Variable</i> value in place of this expression. You cannot assign values to positional parameters in this fashion.
<code>\${Variable:=String}</code>	If the variable is set and not null, substitute the <i>Variable</i> value in place of this expression. Otherwise, set the <i>Variable</i> value to the <i>String</i> value and then substitute the <i>Variable</i> value in place of this expression. You cannot assign values to positional parameters in this fashion.
<code>\${Variable?String}</code>	If the variable is set, substitute the <i>Variable</i> value in place of this expression. Otherwise, display a message of the following form: Variable: String and exit from the current shell (unless the shell is the login shell). If you do not specify a value for the <i>String</i> variable, the shell displays the following message: Variable: parameter null or not set
<code>\${Variable:?String}</code>	If the variable is set and not null, substitute the <i>Variable</i> value in place of this expression. Otherwise, display a message of the following form: Variable: String and exit from the current shell (unless the shell is the login shell). If you do not specify the <i>String</i> value, the shell displays the following message: Variable: parameter null or not set
<code>\${Variable+String}</code>	If the variable is set, substitute the <i>String</i> value in place of this expression. Otherwise, substitute the null string.
<code>\${Variable:+String}</code>	If the variable is set and not null, substitute the <i>String</i> value in place of this expression. Otherwise, substitute the null string.

In conditional substitution, the shell does not evaluate the *String* variable until the shell uses this variable as a substituted string. Thus, in the following example, the shell executes the **pwd** command only if *d* is not set or is null:

```
echo ${d:-`pwd`}
```

Positional parameters in the Bourne shell

When you run a shell procedure, the shell implicitly creates positional parameters that reference each word on the command line by its position on the command line.

The word in position 0 (the procedure name) is called \$0, the next word (the first parameter) is called \$1, and so on, up to \$9. To refer to command line parameters numbered higher than 9, use the built-in **shift** command.

You can reset the values of the positional parameters explicitly by using the built-in **set** command.

Note: When an argument for a position is not specified, its positional parameter is set to null. Positional parameters are global and can be passed to nested shell procedures.

File name substitution in the Bourne shell

The Bourne shell permits you to perform file name substitutions.

Command parameters are often file names. You can automatically produce a list of file names as parameters on a command line. To do this, specify a character that the shell recognizes as a pattern-matching character. When a command includes such a character, the shell replaces it with the file names in a directory.

Note: The Bourne shell does not support file name expansion based on equivalence classification of characters.

Most characters in such a pattern match themselves, but you can also use some special pattern-matching characters in your pattern. These special characters are as follows:

*	Matches any string, including the null string
?	Matches any one character
[. . .]	Matches any one of the characters enclosed in square brackets
[! . . .]	Matches any character within square brackets <i>other than</i> one of the characters that follow the exclamation mark

Within square brackets, a pair of characters separated by a hyphen (-) specifies the set of all characters lexicographically within the inclusive range of that pair, according to the binary ordering of character values.

Pattern matching has some restrictions. If the first character of a file name is a dot (.), it can be matched only by a pattern that also begins with a dot. For example, * matches the file names *myfile* and *yourfile* but not the file names *.myfile* and *.yourfile*. To match these files, use a pattern such as the following:

```
.*file
```

If a pattern does not match any file names, then the pattern itself is returned as the result of the attempted match.

File and directory names should not contain the characters *, ?, [, or] because they can cause infinite recursion (that is, infinite loops) during pattern-matching attempts.

Character classes

You can use character classes to match file names.

You can use character classes to match file names, as follows:

```
[:charclass:]
```

This format instructs the system to match any single character belonging to the specified class. The defined classes correspond to **ctype** subroutines, as follows:

Character Class	Definition
alnum	Alphanumeric characters
alpha	Uppercase and lowercase letters
blank	Space or horizontal tab
cntrl	Control characters
digit	Digits
graph	Graphic characters
lower	Lowercase letters
print	Printable characters
punct	Punctuation characters
space	Space, horizontal tab, carriage return, newline, vertical tab, or form-feed character
upper	Uppercase characters
xdigit	Hexadecimal digits

Input and output redirection in the Bourne shell

There are redirection options that can be used in commands.

In general, most commands do not know whether their input or output is associated with the keyboard, the display screen, or a file. Thus, a command can be used conveniently either at the keyboard or in a pipeline.

The following redirection options can appear anywhere in a simple command. They can also precede or follow a command, but are not passed to the command.

<code><File</code>	Uses the specified file as standard input.
<code>>File</code>	Uses the specified file as standard output. Creates the file if it does not exist; otherwise, truncates it to zero length.
<code>> >File</code>	Uses the specified file as standard output. Creates the file if it does not exist; otherwise, adds the output to the end of the file.
<code><<[-]eofstr</code>	<p>Reads as standard input all lines from the <i>eofstr</i> variable up to a line containing only <i>eofstr</i> or up to an end-of-file character. If any character in the <i>eofstr</i> variable is quoted, the shell does not expand or interpret any characters in the input lines. Otherwise, it performs variable and command substitution and ignores a quoted newline character (<code>\newline</code>). Use a backslash (<code>\</code>) to quote characters within the <i>eofstr</i> variable or within the input lines.</p> <p>If you add a hyphen (-) to the <code><<</code> redirection option, then all leading tabs are stripped from the <i>eofstr</i> variable and from the input lines.</p>
<code><&Digit</code>	Associates standard input with the file descriptor specified by the <i>Digit</i> variable.
<code>>&Digit</code>	Associates standard output with the file descriptor specified by the <i>Digit</i> variable.
<code><&-</code>	Closes standard input.
<code>>&-</code>	Closes standard output.

Note: The restricted shell does not allow output redirection.

For more information about redirection, see Chapter 29, “Input and output redirection,” on page 371.

List of Bourne shell built-in commands

The following is a list of Bourne shell built-in commands.

:	Returns a zero exit value
.	Reads and executes commands from a file parameter and then returns.
break	Exits from the enclosing for , while , or until command loops, if any.
cd	Changes the current directory to the specified directory.
continue	Resumes the next iteration of the enclosing for , while , or until command loops.
echo	Writes character strings to standard output.
eval	Reads the arguments as input to the shell and executes the resulting command or commands.
exec	Executes the command specified by the Argument parameter, instead of this shell, without creating a new process.
exit	Exits the shell whose exit status is specified by the n parameter.
export	Marks names for automatic export to the environment of subsequently executed commands.
hash	Finds and remembers the location in the search path of specified commands.
pwd	Displays the current directory.
read	Reads one line from standard input.
readonly	Marks name specified by Name parameter as read-only.
return	Causes a function to exit with a specified return value.
set	Controls the display of various parameters to standard output.
shift	Shifts command-line arguments to the left.
test	Evaluates conditional expressions.
times	Displays the accumulated user and system times for processes run from the shell.
trap	Runs a specified command when the shell receives a specified signal or signals.
type	Interprets how the shell would interpret a specified name as a command name.
ulimit	Displays or adjusts allocated shell resources.
umask	Determines file permissions.
unset	Removes the variable or function corresponding to a specified name.
wait	Waits for the specified child process to end and reports its termination status.

C shell

The C shell is an interactive command interpreter and a command programming language. It uses syntax that is similar to the C programming language.

The **cs**h command starts the C shell.

When you log in, the **cs**h command first searches the system-wide setup file **/etc/csh.cshrc**. If the setup file exists, the C shell executes the commands stored in that file. Next, the C shell executes the system-wide setup file **/etc/csh.login** if it is available. Then, it searches your home directory for the **.cshrc** and **.login** files. If they exist, they contain any customized user information pertinent to running the C shell. All variables set in the **/etc/csh.cshrc** and **/etc/csh.login** files might be overridden by your **.cshrc** and **.login** files in your **\$HOME** directory. Only the root user can modify the **/etc/csh.cshrc** and **/etc/csh.login** files.

The **/etc/csh.login** and **\$HOME/.login** files are executed only once at login time. These files are generally used to hold environment variable definitions, commands that you want executed once at login, or commands that set up terminal characteristics.

The **/etc/csh.cshrc** and **\$HOME/.cshrc** files are executed at login time and every time the **cs**h command or a C shell script is invoked. They are generally used to define C shell characteristics, such as aliases and C shell variables (for example, *history*, *noclobber*, or *ignoreeof*). It is recommended that you only use the C shell built-in commands (see “C shell built-in commands” on page 519) in the **/etc/csh.cshrc** and **\$HOME/.cshrc** files because using other commands increases the startup time for shell scripts.

C shell limitations

The following are limitations of the C shell.

- Words can be no longer than 1024 bytes.
- Argument lists are limited to *ARG_MAX* bytes. Values for the *ARG_MAX* variable are found in the */usr/include/sys/limits.h* file.
- The number of arguments to a command that involves file name expansion is limited to 1/6th the number of bytes allowed in an argument list.
- Command substitutions can substitute no more bytes than are allowed in an argument list.
- To detect looping, the shell restricts the number of alias substitutions on a single line to 20.
- The **cs**h command does not support file name expansion based on equivalence classification of characters.
- File descriptors (other than standard in, standard out, and standard error) opened before **cs**h executes any application are not available to that application.

Signal handling in the C shell

The C shell normally ignores quit signals. Jobs running detached are not affected by signals generated from the keyboard (**INTERRUPT**, **QUIT**, and **HANGUP**).

Other signals have the values the shell inherits from its parent. You can control the shell's handling of **INTERRUPT** and **TERMINATE** signals in shell procedures with **onintr**. Login shells catch or ignore **TERMINATE** signals depending on how they are set up. Shells other than login shells pass **TERMINATE** signals on to the child processes. In no cases are **INTERRUPT** signals allowed when a login shell is reading the *.logout* file.

C shell commands

A simple command is a sequence of words separated by blanks or tabs. A *word* is a sequence of characters or numerals, or both, that does not contain blanks without quotation marks.

In addition, the following characters and doubled characters also form single words when used as command separators or terminators:

&		;	
&&		<<	>>
<	>	()

These special characters can be parts of other words. Preceding them with a backslash (\), however, prevents the shell from interpreting them as special characters. Strings enclosed in ' ' or " " (matched pairs of quotation characters) or backquotes can also form parts of words. Blanks, tab characters, and special characters do not form separate words when they are enclosed in these marks. In addition, you can enclose a newline character within these marks by preceding it with a backslash (\).

The first word in the simple command sequence (numbered 0) usually specifies the name of a command. Any remaining words, with a few exceptions, are passed to that command. If the command specifies an executable file that is a compiled program, the shell immediately runs that program. If the file is marked executable but is not a compiled program, the shell assumes that it is a shell script. In this case, the shell starts another instance of itself (a subshell) to read the file and execute the commands included in it.

C shell built-in commands

Built-in commands are run within the shell. If a built-in command occurs as any component of a pipeline, except the last, the command runs in a subshell.

Note: If you enter a command from the C shell prompt, the system searches for a built-in command first. If a built-in command does not exist, the system searches the directories specified by the *path* shell

variable for a system-level command. Some C shell built-in commands and operating system commands have the same name. However, these commands do not necessarily work the same way. For more information on how the command works, check the appropriate command description.

If you run a shell script from the shell, and the first line of the shell script begins with `#!/ShellPathname`, the C shell runs the shell specified in the comment to process the script. Otherwise, it runs the default shell (the shell linked to `/usr/bin/sh`). If run by the default shell, C shell built-in commands might not be recognized. To run C shell commands, make the first line of the script `#!/usr/bin/csh`.

See “C shell built-in commands” on page 519 for an alphabetic listing of the built-in commands.

C shell command descriptions

The C shell provides the following built-in commands.

alias [<i>Name</i> [<i>WordList</i>]]	Displays all aliases if you do not specify any parameters. Otherwise, the command displays the alias for the specified <i>Name</i> . If <i>WordList</i> is specified, this command assigns the value of <i>WordList</i> to the alias <i>Name</i> . The specified alias <i>Name</i> cannot be <code>alias</code> or <code>unalias</code> .
bg [% <i>Job</i> ...]	Puts the current job or job specified by <i>Job</i> into the background, continuing the job if it was stopped.
break	Resumes running after the end of the nearest enclosing foreach or while command.
breaksw	Breaks from a switch command; resumes after the endsw command.
case <i>Label</i> :	Defines a <i>Label</i> in a switch command.
cd [<i>Name</i>]	Equivalent to the chdir command (see following description).
chdir [<i>Name</i>]	Changes the current directory to that specified by the <i>Name</i> variable. If you do not specify <i>Name</i> , the command changes to your home directory. If the value of the <i>Name</i> variable is not a subdirectory of the current directory and does not begin with <code>/</code> , <code>./</code> , or <code>../</code> , the shell checks each component of the cdpath shell variable to see if it has a subdirectory matching the <i>Name</i> variable. If the <i>Name</i> variable is a shell variable with a value that begins with a slash (<code>/</code>), the shell tries this to see if it is a directory. The chdir command is equivalent to the cd command.
continue	Continues execution at the end of the nearest enclosing while or foreach command.
default :	Labels the default case in a switch statement. The default should come after all other case labels.
dirs	Displays the directory stack.
echo	Writes character strings to the standard output of the shell.
else	Runs the commands that follow the second else in an if (<i>Expression</i>) then ... else if (<i>Expression2</i>) then ... else ... endif command sequence. Note: The else statement is the csh built-in command when using the if (<i>expr</i>) then .. else ... endif . If the (<i>expr</i>) is true, then the commands up to the else statement is executed. If the (<i>expr</i>) is false, then the commands between the else and endif statement are executed. Anything in single quotes is taken literally and not interpreted.

end	<p>Successively sets the <i>Name</i> variable to each member specified by the <i>List</i> variable and runs the sequence of <i>Commands</i> between the foreach and the matching end statements. The foreach and end statements must appear alone on separate lines.</p> <p>Uses the <code>continue</code> statement to continue the loop and the <code>break</code> statement to end the loop prematurely. When the foreach command is read from the terminal, the C shell prompts with a <code>?</code> to allow <i>Commands</i> to be entered. Commands within loops, prompted for by <code>?</code>, are not placed in the history list.</p>
endif	<p>If the <i>Expression</i> variable is true, runs the <i>Commands</i> that follow the first then statement. If the <code>else if</code> <i>Expression2</i> is true, runs the <i>Commands</i> that follow the second then statement. If the <code>else if</code> <i>Expression2</i> is false, runs the <i>Commands</i> that follow the <code>else</code>. Any number of <code>else if</code> pairs are possible. Only one endif statement is needed. The <code>else</code> segment is optional. The words <code>else</code> and endif can be used only at the beginning of input lines. The <code>if</code> segment must appear alone on its input line or after an <code>else</code> command.</p>
endsw	<p>Successively matches each case label against the value of the <i>string</i> variable. The <i>string</i> is command and file name expanded first. Use the pattern-matching characters <code>*</code>, <code>?</code>, and <code>[. . .]</code> in the case labels, which are variable-expanded. If none of the labels match before a default label is found, the execution begins after the default label. The case label and the default label must appear at the beginning of the line. The breaksw command causes execution to continue after the endsw command. Otherwise, control might fall through the case and default labels, as in the C programming language. If no label matches and there is no default, execution continues after the endsw command.</p>
eval <i>Parameter</i> . . .	<p>Reads the value of the <i>Parameter</i> variable as input to the shell and runs the resulting command or commands in the context of the current shell. Use this command to run commands generated as the result of command or variable substitution because parsing occurs before these substitutions.</p>
exec <i>Command</i>	<p>Runs the specified <i>Command</i> in place of the current shell.</p>
exit (<i>Expression</i>)	<p>Exits the shell with either the value of the status shell variable (if no <i>Expression</i> is specified) or with the value of the specified <i>Expression</i>.</p>
fg [% <i>Job</i> ...]	<p>Brings the current job or job specified by <i>Job</i> into the foreground, continuing the job if it was stopped.</p>
foreach <i>Name</i> (<i>List</i>) <i>Command</i> . . .	<p>Successively sets a <i>Name</i> variable for each member specified by the <i>List</i> variable and a sequence of commands, until reaching an end command.</p>
glob <i>List</i>	<p>Displays <i>List</i> using history, variable, and file name expansion. Puts a null character between words and does not include a carriage return at the end.</p>
goto <i>Word</i>	<p>Continues to run after the line specified by the <i>Word</i> variable. The specified <i>Word</i> is file name and command expanded to yield a string of the form specified by the <i>Label:</i> variable. The shell rewinds its input as much as possible and searches for a line of the form <i>Label:</i>, possibly preceded by blanks or tabs.</p>
hashstat	<p>Displays statistics indicating how successful the hash table has been at locating commands.</p>
history [-r -h] [<i>n</i>]	<p>Displays the history event list. The oldest events are displayed first. If you specify a number <i>n</i>, only the specified number of the most recent events are displayed. The -r flag reverses the order in which the events are displayed so the most recent is displayed first. The -h flag displays the history list without leading numbers. Use this flag to produce files suitable for use with the -h flag of the source command.</p>

if (<i>Expression</i>) <i>Command</i>	<p>Runs the specified <i>Command</i> (including its arguments) if the specified <i>Expression</i> is true. Variable substitution on the <i>Command</i> variable happens early, at the same time as the rest of the <i>if</i> statement. The specified <i>Command</i> must be a simple command (rather than a pipeline, command list, or parenthesized command list).</p> <p>Note: Input and output redirection occurs even if the <i>Expression</i> variable is false and the <i>Command</i> is not executed.</p>
jobs [-l]	<p>Lists the active jobs. With the -l (lowercase <i>l</i>) flag, the jobs command lists process IDs in addition to the job number and name.</p>
kill -l [- <i>Signal</i>] % <i>Job</i> ...! <i>PID</i> ...	<p>Sends either the TERM (terminate) signal or the signal specified by <i>Signal</i> to the specified <i>Job</i> or <i>PID</i> (process). Specify signals either by number or by name (as given in the /usr/include/sys/signal.h file, stripped of the SIG prefix). The -l (lowercase <i>l</i>) flag lists the signal names.</p>
limit [-h] [<i>Resource</i> [<i>Max-Use</i>]]	<p>Limits the usage of the specified resource by the current process and each process it creates. Process resource limits are defined in the /etc/security/limits file. Controllable resources are the central processing unit (CPU) time, file size, data size, core dump size, and memory use. Maximum allowable values for these resources are set with the mkuser command when the user is added to the system. They are changed with the chuser command.</p> <p>Limits are categorized as either soft or hard. Users may increase their soft limits up to the ceiling imposed by the hard limits. You must have root user authority to increase a soft limit above the hard limit, or to change hard limits. The -h flag displays hard limits instead of the soft limits.</p> <p>If a <i>Max-Use</i> parameter is not specified, the limit command displays the current limit of the specified resource. If the <i>Resource</i> parameter is not specified, the limit command displays the current limits of all resources. For more information about the resources controlled by the limit subcommand, see the getrlimit, setrlimit, or vlimit subroutine in the <i>AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 1</i>.</p> <p>The <i>Max-Use</i> parameter for CPU time is specified in the <i>hh:mm:ss</i> format. The <i>Max-Use</i> parameter for other resources is specified as a floating-point number or an integer optionally followed by a scale factor. The scale factor is k or kilobytes (1024 bytes), m or megabytes, or b or blocks (the units used by the ulimit subroutine as explained in the <i>AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 2</i>). If you do not specify a scale factor, k is assumed for all resources. For both resource names and scale factors, unambiguous prefixes of the names suffice.</p> <p>Note: This command limits the physical memory (memory use) available for a process only if there is contention for system memory by other active processes.</p>
login	<p>Ends a login shell and replaces it with an instance of the /usr/bin/login command. This is one way to log out (included for compatibility with the ksh and bsh commands).</p>
logout	<p>Ends a login shell. This command must be used if the ignoreeof option is set.</p>
nice [+ <i>n</i>] [<i>Command</i>]	<p>If no values are specified, sets the priority of commands run in this shell to 24. If the +<i>n</i> flag is specified, sets the priority plus the specified number. If the +<i>n</i> flag and <i>Command</i> are specified, runs <i>Command</i> at priority 24 plus the specified number. If you have root user authority, you can run the nice statement with a negative number. The <i>Command</i> always runs in a subshell, and the restrictions placed on commands in simple <i>if</i> statements apply.</p>

nohup [<i>Command</i>]	Causes hangups to be ignored for the remainder of the script when no <i>Command</i> is specified. If <i>Command</i> is specified, causes the specified <i>Command</i> to be run with hangups ignored. To run a pipeline or list of commands, put the pipeline or list in a shell script, give the script execute permission, and use the shell script as the value of the <i>Command</i> variable. All processes run in the background with an ampersand (&) are effectively protected from being sent a hangup signal when you log out. However, these processes are still subject to explicitly sent hangups unless the nohup statement is used.
notify [% <i>Job</i> ...]	Causes the shell to notify you asynchronously when the status of the current job or specified <i>Job</i> changes. Normally, the shell provides notification just before it presents the shell prompt. This feature is automatic if the notify shell variable is set.
onintr [- <i>Label</i>]	Controls the action of the shell on interrupts. If no arguments are specified, restores the default action of the shell on interrupts, which ends shell scripts or returns to the command input level. If a - flag is specified, causes all interrupts to be ignored. If <i>Label</i> is specified, causes the shell to run a <i>goto Label</i> statement when the shell receives an interrupt or when a child process ends due to an interruption. In any case, if the shell is running detached and interrupts are being ignored, all forms of the onintr statement have no meaning. Interrupts continue to be ignored by the shell and all invoked commands.
popd [+ <i>n</i>]	Pops the directory stack and changes to the new top directory. If you specify a +<i>n</i> variable, the command discards the <i>n</i> th entry in the stack. The elements of the directory stack are numbered from the top, starting at 0.
pushd [+ <i>n</i> / <i>Name</i>]	With no arguments, exchanges the top two elements of the directory stack. With the <i>Name</i> variable, the command changes to the new directory and pushes the old current directory (as given in the cwd shell variable) onto the directory stack. If you specify a +<i>n</i> variable, the command rotates the <i>n</i> th component of the directory stack around to be the top element and changes to it. The members of the directory stack are numbered from the top, starting at 0.
rehash	Causes recomputation of the internal hash table of the contents of the directories in the path shell variable. This action is needed if new commands are added to directories in the path shell variable while you are logged in. The rehash command is necessary only if commands are added to one of the user's own directories or if someone changes the contents of one of the system directories.
repeat <i>Count Command</i>	Runs the specified <i>Command</i> , subject to the same restrictions as commands in simple <i>if</i> statements, the number of times specified by <i>Count</i> . Note: I/O redirections occur exactly once, even if the <i>Count</i> variable equals 0.
set [[<i>Name</i> [<i>n</i>]] [= <i>Word</i>]] [<i>Name</i> = (<i>List</i>)]	Shows the value of all shell variables when used with no arguments. Variables that have more than a single word as their value are displayed as a parenthesized word list. If only <i>Name</i> is specified, the C shell sets the <i>Name</i> variable to the null string. Otherwise, sets <i>Name</i> to the value of the <i>Word</i> variable, or sets the <i>Name</i> variable to the list of words specified by the <i>List</i> variable. When <i>n</i> is specified, the <i>n</i> th component of the <i>Name</i> variable is set to the value of the <i>Word</i> variable; the <i>n</i> th component must already exist. In all cases, the value is command and file name expanded. These arguments may be repeated to set multiple values in a single set command. However, variable expansion happens for all arguments before any setting occurs.

setenv <i>Name Value</i>	Sets the value of the environment variable specified by the <i>Name</i> variable to <i>Value</i> , a single string. The most commonly used environment variables, USER , TERM , HOME , and PATH , are automatically imported to and exported from the C shell variables user , term , home , and path . There is no need to use the setenv statement for these.
shift [<i>Variable</i>]	Shifts the members of the argv shell variable or the specified <i>Variable</i> to the left. An error occurs if the argv shell variable or specified <i>Variable</i> is not set or has less than one word as its value.
source [-h] <i>Name</i>	Reads commands written in the <i>Name</i> file. You can nest the source commands. However, if they are nested too deeply, the shell might run out of file descriptors. An error in a source command at any level ends all nested source commands. Normally, input during source commands is not placed on the history list. The -h flag causes the commands to be placed in the history list without executing them.
stop [% <i>Job</i> ...]	Stops the current job or specified <i>Job</i> running in the background.
suspend	Stops the shell as if a STOP signal had been received.
switch (<i>string</i>)	Starts a switch (<i>String</i>) case <i>String</i> : ... breaksw default: ... breaksw endsw command sequence. This command sequence successively matches each case label against the value of the <i>String</i> variable. If none of the labels match before a default label is found, the execution begins after the default label.
time [<i>Command</i>]	<p>The time command controls automatic timing of commands. If you do not specify the <i>Command</i> variable, the time command displays a summary of time used by this shell and its children. If you specify a command with the <i>Command</i> variable, it is timed. The shell then displays a time summary, as described under the time shell variable. If necessary, an extra shell is created to display the time statistic when the command completes.</p> <p>The following example uses time with the sleep command:</p> <pre>time sleep</pre> <p>The output from this command looks similar to the following:</p> <pre>0.0u 0.0s 0:00 100% 44+4k 0+0io 0pf+0w</pre> <p>The output fields are as follows:</p> <ul style="list-style-type: none"> First Number of seconds of CPU time devoted to the user process Second Number of seconds of CPU time consumed by the kernel on behalf of the user process Third Elapsed (wall clock) time for the command Fourth Total user CPU Time plus system time, as a percentage of elapsed time Fifth Average amount of shared memory used, plus average amount of unshared data space used, in kilobytes Sixth Number of block input and output operations Seventh Page faults plus number of swaps
umask [<i>Value</i>]	Determines file permissions. This <i>Value</i> , along with the permissions of the creating process, determines a file's permissions when the file is created. The default is 022. The current setting will be displayed if no <i>Value</i> is specified.

unalias * <i>Pattern</i>	Discards all aliases with names that match the <i>Pattern</i> variable. All aliases are removed by the unalias * command. The absence of aliases does not cause an error.
unhash	Disables the use of the internal hash table to locate running programs.
unlimit [-h][<i>Resource</i>]	Removes the limitation on the <i>Resource</i> variable. If no <i>Resource</i> variable is specified, all resource limitations are removed. See the description of the limit command for the list of <i>Resource</i> names. The -h flag removes corresponding hard limits. Only a user with root user authority can change hard limits.
unset * <i>Pattern</i>	Removes all variables with names that match the <i>Pattern</i> variable. Use unset * to remove all variables. If no variables are set, it does not cause an error.
unsetenv <i>Pattern</i>	Removes all variables from the environment whose name matches the specified <i>Pattern</i> . (See the setenv built-in command.)
wait	Waits for all background jobs. If the shell is interactive, an INTERRUPT (usually the Ctrl-C key sequence) disrupts the wait. The shell then displays the names and job numbers of all jobs known to be outstanding.
while (<i>Expression</i>) <i>Command</i> . . . end	Evaluates the <i>Commands</i> between the while and the matching end statements while the expression specified by the <i>Expression</i> variable evaluates nonzero. You can use the break statement to end and the continue statement to continue the loop prematurely. The while and end statements must appear alone on their input lines. If the input is from a terminal, prompts occur after the while (<i>Expression</i>) similar to the foreach statement.
@ [<i>Name</i> [<i>n</i>] = <i>Expression</i>]	Displays the values of all the shell variables when used with no arguments. Otherwise, sets the name specified by the <i>Name</i> variable to the value of the <i>Expression</i> variable. If the expression contains <, >, &, or characters, this part of the expression must be placed within parentheses. When <i>n</i> is specified, the <i>n</i> th component of the <i>Name</i> variable is set to the <i>Expression</i> variable. Both the <i>Name</i> variable and its <i>n</i> th component must already exist. C language operators, such as *= and +=, are available. The space separating the <i>Name</i> variable from the assignment operator is optional. Spaces are, however, required in separating components of the <i>Expression</i> variable, which would otherwise be read as a single word. Special suffix operators, double plus sign (++) and double hyphen (--) increase and decrease, respectively, the value of the <i>Name</i> variable.

C shell expressions and operators

The @ built-in command and the **exit**, **if**, and **while** statements accept expressions that include operators similar to those of C language, with the same precedence.

The following operators are available:

Operator	What it means
()	change precedence
~	complement
!	negation
*/ %	multiply, divide, modulo
+ -	add, subtract
<< > >	left shift, right shift
<= >= < >	relational operators
== != =~ !~	string comparison/pattern matching
&	bitwise AND

Operator	What it means
<code>^</code>	bitwise exclusive OR
<code> </code>	bitwise inclusive OR
<code>&&</code>	logical AND
<code> </code>	logical OR

In the previous list, precedence of the operators decreases down the list (left to right, top to bottom).

Note: The operators `+` and `-` are right-associative. For example, evaluation of `a + b - c` is performed as follows:

`a + (b - c)`

and not as follows:

`(a + b) - c`

The `==`, `!=`, `=~`, and `!~` operators compare their arguments as strings; all others operate on numbers. The `=~` and `!~` operators are similar to `==` and `!=`, except that the rightmost side is a *pattern* against which the leftmost operand is matched. This reduces the need for use of the **switch** statement in shell procedures.

The logical operators **or** (`||`) and **and** (`&&`) are also available. They can be used to check for a range of numbers, as in the following example:

```
if ($#argv > 2 && $#argv < 7) then
```

In the preceding example, the number of arguments must be greater than 2 and less than 7.

Strings beginning with zero (0) are considered octal numbers. Null or missing arguments are considered 0. All expressions result in strings representing decimal numbers. Note that two components of an expression can appear in the same word. Except when next to components of expressions that are syntactically significant to the parser (`&` `|` `<` `>` `(` `)`), expression components should be surrounded by spaces.

Also available in expressions as primitive operands are command executions enclosed in parentheses (`()`) and file inquiries of the form (**-operator** *Filename*), where **operator** is one of the following:

r	Read access
w	Write access
x	Execute access
e	Existence
o	Ownership
z	Zero size
f	Plain file
d	Directory

The specified *Filename* is command and file name expanded and then tested to see if it has the specified relationship to the real user. If *Filename* does not exist or is inaccessible, all inquiries return `false(0)`. If the command runs successfully, the inquiry returns a value of `true(1)`. Otherwise, if the command fails, the inquiry returns a value of `false(0)`. If more detailed status information is required, run the command outside an expression and then examine the *status* shell variable.

Command substitution in the C shell

In *command substitution*, the shell executes a specified command and replaces that command with its output.

To perform command substitution in the C shell, enclose the command or command string in backquotes (` `). The shell normally breaks the output from the command into separate words at blanks, tabs, and newline characters. It then replaces the original command with this output.

In the following example, the backquotes (` `) around the **date** command indicate that the output of the command will be substituted:

```
echo The current date and time is: `date`
```

The output from this command might look like the following:

```
The current date and time is: Wed Apr 8 13:52:14 CDT 1992
```

The C shell performs command substitution selectively on the arguments of built-in shell commands. This means that it does not expand those parts of expressions that are not evaluated. For commands that are not built-in, the shell substitutes the command name separately from the argument list. The substitution occurs in a child of the main shell, but only after the shell performs input or output redirection.

If a command string is surrounded by " ", the shell treats only newline characters as word separators, thus preserving blanks and tabs within the word. In all cases, the single final newline character does not force a new word.

Nonbuilt-in C shell command execution

When the C shell determines that a command is not a built-in shell command, it attempts to run the command with the **execv** subroutine.

Each word in the *path* shell variable names a directory from which the shell attempts to run the command. If given neither the **-c** nor **-t** flag, the shell hashes the names in these directories into an internal table. The shell tries to call the **exec** subroutine on a directory only if there is a possibility that the command resides there. If you turn off this mechanism with the **unhash** command or give the shell the **-c** or **-t** flag, the shell concatenates with the given command name to form a path name of a file. The shell also does this in any case for each directory component of the *path* variable that does not begin with a slash (/). The shell then attempts to run the command.

Parenthesized commands always run in a subshell. For example:

```
(cd ; pwd) ; pwd
```

displays the home directory without changing the current directory location. However, the command:

```
cd ; pwd
```

changes the current directory location to the home directory. Parenthesized commands are most often used to prevent the **chdir** command from affecting the current shell.

If the file has execute permission, but is not an executable binary to the system, then the shell assumes it is a file containing shell commands and runs a new shell to read it.

If there is an alias for the shell, then the words of the alias are prefixed to the argument list to form the shell command. The first word of the alias should be the full path name of the shell.

For more information about the **execv** subroutine, see **execv**.

History substitution in the C shell

History substitution lets you modify individual words from previous commands to create new commands. History substitution makes it easy to repeat commands, repeat the arguments of a previous command in the current command, or fix spelling mistakes in the previous command with little typing.

History substitutions begin with the exclamation mark (!) character and can appear anywhere on the command line, provided they do not nest (in other words, a history substitution cannot contain another history substitution). You can precede the ! with a \ to cancel the exclamation point's special meaning. In addition, if you place the ! before a blank, tab, newline character, =, or (, history substitution does not occur.

History substitutions also occur when you begin an input line with a caret (^). The shell echoes any input line containing history substitutions at the workstation before it executes that line.

History lists for the C shell

The history list saves commands that the shell reads from the command line that consist of one or more words. History substitution reintroduces sequences of words from these saved commands into the input stream.

The *history* shell variable controls the size of the history list. You must set the *history* shell variable either in the `.cshrc` file or on the command line with the built-in `set` command. The previous command is always retained regardless of the value of the *history* variable. Commands in the history list are numbered sequentially, beginning with 1. The built-in **history** command produces output similar to the following:

```
9 write michael
10 ed write.c
11 cat oldwrite.c
12 diff *write.c
```

The shell displays the command strings with their event numbers. The event number appears to the left of the command and represent when the command was entered in relation to the other commands in the history. It is not usually necessary to use event numbers to refer to events, but you can have the current event number displayed as part of your system prompt by placing an exclamation mark (!) in the prompt string assigned to the *PROMPT* environment variable.

A full history reference contains an event specification, a word designator, and one or more modifiers in the following general format:

```
Event[.]Word:Modifier[:Modifier] . . .
```

Note: Only one word can be modified. A string that contains blanks is not allowed.

In the previous sample of **history** command output, the current event number is 13. Using this example, the following refer to previous events:

```
!10      Event number 10.
!-2      Event number 11 (the current event minus 2).
!d       Command word beginning with d (event number 12).
!?mic?   Command word containing the string mic (event number 9).
```

These forms, without further modification, simply reintroduce the words of the specified events, each separated by a single blank. As a special case, !! refers to the previous command; the command !! alone on an input line reruns the previous command.

Event specification for the C shell

To select words from an event, follow the event specification with a colon (:) and one of the following word designators (the words of an input line are numbered sequentially starting from 0)

```
0      First word (the command name)
n      nth argument
^      First argument
```

\$	Last argument
%	Word matched by an immediately preceding <i>?string?</i> search
x-y	Range of words from the x th word to the y th word
-y	Range of words from the first word (0) to the y th word
*	First through the last argument, or nothing if there is only one word (the command name) in the event
x*	x th argument through the last argument
x-	Same as x* but omitting the last argument

If the word designator begins with a ^, \$, *, -, or %, you can omit the colon that separates the event specification from the word designator. You can also place a sequence of the following modifiers after the optional word designator, each preceded by a colon:

h	Removes a trailing path name extension, leaving the head.
r	Removes a trailing .xxx component, leaving the root name.
e	Removes all but the .xxx trailing extension.
s/OldWord/NewWord/	Substitutes the value of the <i>NewWord</i> variable for the value of the <i>OldWord</i> variable.

The left side of a substitution is not a pattern in the sense of a string recognized by an editor; rather, it is a word, a single unit without blanks. Normally, a slash (/) delimits the original word (*OldWord*) and its replacement (*NewWord*). However, you can use any character as the delimiter. In the following example, using the % as a delimiter allows a / to be included in the words:

```
s%/home/myfile%/home/yourfile%
```

The shell replaces an ampersand (&) with the *OldWord* text in the *NewWord* variable. In the following example, **/home/myfile** becomes **/temp/home/myfile**.

```
s%/home/myfile%/temp&%
```

The shell replaces a null word in a substitution with either the last substitution or with the last string used in the contextual scan *!?String?*. You can omit the trailing delimiter (/) if a newline character follows immediately. Use the following modifiers to delimit the history list:

t	Removes all leading path name components, leaving the tail
&	Repeats the previous substitution
g	Applies the change globally; that is, all occurrences for each line
p	Displays the new command, but does not run it
q	Quotes the substituted words, thus preventing further substitutions
x	Acts like the q modifier, but breaks into words at blanks, tabs, and newline characters

When using the preceding modifiers, the change applies only to the first modifiable word unless the **g** modifier precedes the selected modifier.

If you give a history reference without an event specification (for example, !\$), the shell uses the previous command as the event. If a previous history reference occurs on the same line, the shell repeats the previous reference. Thus, the following sequence gives the first and last arguments of the command that matches *?foo?*.

```
!?foo?^ !$
```

A special abbreviation of a history reference occurs when the first nonblank character of an input line is a carat (^). This is equivalent to *!:s^*, thus providing a convenient shorthand for substitutions on the text of the previous line. The command *^ 1b^ lib* corrects the spelling of *1 b* in the command.

If necessary, you can enclose a history substitution in braces { } to insulate it from the characters that follow. For example, if you want to use a reference to the command:

```
ls -ld ~paul
```

to perform the command:

```
ls -ld ~paula
```

use the following construction:

```
!{1}a
```

In this example, `!{1}a` looks for a command starting with `l` and appends `a` to the end.

Quotation with single and double quotes

To prevent further interpretation of all or some of the substitutions, enclose strings in single and double quotation marks.

Enclosing strings in `' '` prevents further interpretation, while enclosing strings in `" "` allows further expansion. In both cases, the text that results becomes all or part of a single word.

Alias substitution in the C shell

An *alias* is a name assigned to a command or command string. The C shell allows you to assign aliases and use them as you would commands. The shell maintains a list of the aliases that you define.

After the shell scans the command line, it divides the commands into distinct words and checks the first word of each command, left to right, to see if there is an alias. If an alias is found, the shell uses the history mechanism to replace the text of the alias with the text of the command referenced by the alias. The resulting words replace the command and argument list. If no reference is made to the history list, the argument list is left unchanged.

For information about the C shell history mechanism, see “History substitution in the C shell” on page 508.

The **alias** and **unalias** built-in commands establish, display, and modify the alias list. Use the alias command in the following format:

```
alias [Name [WordList]]
```

The optional *Name* variable specifies the alias for the specified name. If you specify a word list with the *WordList* variable, the command assigns it as the alias of the *Name* variable. If you run the **alias** command without either optional variable, it displays all C shell aliases.

If the alias for the **ls** command is **ls -l**, the following command:

```
ls /usr
```

is replaced by the command:

```
ls -l /usr
```

The argument list is undisturbed because there is no reference to the history list in the command with an alias. Similarly, if the alias for the **lookup** command is as follows:

```
grep \!^ /etc/passwd
```

then the shell replaces `lookup bill` with the following:

```
grep bill /etc/passwd
```

In this example, !^ refers to the history list, and the shell replaces it with the first argument in the input line, in this case `bill`.

You can use special pattern-matching characters in an alias. The following command:

```
alias lprint 'pr &bslash2.!* >
> print'
```

creates a command that formats its arguments to the line printer. The ! character is protected from the shell in the alias by use of single quotation marks so that the alias is not expanded until the `pr` command runs.

If the shell locates an alias, it performs the word transformation of the input text and begins the alias process again on the reformed input line. If the first word of the next text is the same as the previous text, then looping is prevented by flagging the alias to terminate the alias process. Other subsequent loops are detected and result in an error.

Variable substitution in the C shell

The C shell maintains a set of variables, each of which has as its value a list of zero or more words. Some of these variables are set by the shell or referred to by it. For example, the `argv` variable is an image of the shell variable list, and words that comprise the value of this variable are referred to in special ways.

To change and display the values of variables, use the `set` and `unset` commands. Of the variables referred to by the shell, a number are toggles (variables that turn something on and off). The shell does not examine toggles for a value, only for whether they are set or unset. For example, the `verbose` shell variable is a toggle that causes command input to be echoed. The setting of this variable results from issuing the `-v` flag on the command line.

Other operations treat variables numerically. The `@` command performs numeric calculations, and the result is assigned to a variable. Variable values are, however, always represented as (zero or more) strings. For numeric operations, the null string is considered to be zero, and the second and subsequent words of multi-word values are ignored.

When you issue a command, the shell parses the input line and performs alias substitution. Next, before running the command, it performs variable substitution. The `$` character keys the substitution. It is, however, passed unchanged if followed by a blank, tab, or newline character. Preceding the `$` character with a `\` prevents this expansion, except in two cases:

- The command is enclosed in `" "`. In this case, the shell always performs the substitution.
- The command is enclosed in `' '`. In this case, the shell never performs the substitution. Strings enclosed in `' '` are interpreted for command substitution. (See “Command substitution in the C shell” on page 507.)

The shell recognizes input and output redirection before variable expansion and expands each separately. Otherwise, the command name and complete argument list expand together. It is therefore possible for the first (command) word to generate more than one word, the first of which becomes the command name, and the rest of which become parameters.

Unless enclosed in `" "` or given the `:q` modifier, the results of variable substitution might eventually be subject to command and file name substitution. When enclosed by double quotation marks, a variable with a value that consists of multiple words expands to a single word or a portion of a single word, with the words of the variable’s value separated by blanks. When you apply the `:q` modifier to a substitution, the variable expands to multiple words. Each word is separated by a blank and enclosed in double quotation marks to prevent later command or file name substitution.

The following notations allow you to introduce variable values into the shell input. Except as noted, it is an error to reference a variable that is not set with the **set** command.

You can apply the modifiers **:gh**, **:gt**, **:gr**, **:h**, **:r**, **:q**, and **:x** to the following substitutions. If **{ }** appear in the command form, then the modifiers must be placed within the braces. Only one **:** modifier is permitted on each variable expansion.

<i>\$Name</i>	
<i>\${Name}</i>	Replaced by the words assigned to the <i>Name</i> variable, each separated by a blank. Braces insulate the <i>Name</i> variable from any following characters that would otherwise be part of it. Shell variable names start with a letter and consist of up to 20 letters and digits, including the underline (_) character. If the <i>Name</i> variable does not specify a shell variable but is set in the environment, then its value is returned. The modifiers preceded by colons, as well as the other forms described here, are not available in this case.
<i>\$Name[number]</i>	
<i>\${Name[number]}</i>	Selects only some of the words from the value of the <i>Name</i> variable. The number is subjected to variable substitution and might consist of a single number or two numbers separated by a hyphen (-). The first word of a variable's string value is numbered 1. If the first number of a range is omitted, it defaults to 1. If the last number of a range is omitted, it defaults to <i> \$#Name</i> . The asterisk (*) symbol selects all words. It is not an error for a range to be empty if the second argument is omitted or is in a range.
<i> \$#Name</i>	
<i> \$#Name</i>	Gives the number of words in the <i>Name</i> variable. This can be used in a <i>[number]</i> as shown above. For example, <i>\$Name[\$ #Name]</i> .
<i>\$0</i>	Substitutes the name of the file from which command input is being read. An error occurs if the name is not known.
<i>\$number</i>	
<i>\${number}</i>	Equivalent to <i>\$argv[number]</i> .
<i>\$*</i>	Equivalent to <i>\$argv[*]</i> .

The following substitutions may not be changed with **:** modifiers:

<i> \$?name</i>	
<i> \${?name}</i>	Substitutes the string 1 if the <i>name</i> variable is set, zero (0) if this variable is not set.
<i> \$?0</i>	Substitutes 1 if the current input file name is known, zero (0) if the file name is not known.
<i> \$\$</i>	Substitutes the (decimal) process number of the parent shell.
<i> \$<</i>	Substitutes a line from standard input, without further interpretation. Use this substitution to read from the keyboard in a shell procedure.

File name substitution in the C shell

The C Shell permits you to do file name substitutions.

The C shell provides several shortcuts to save time and keystrokes. If a word contains any of the characters *****, **?**, **[]**, or **{ }**, or begins with a tilde (**~**), that word is a candidate for file name substitution. The C shell regards the word as a pattern and replaces the word with an alphabetized list of file names matching the pattern.

The current collating sequence is used, as specified by the *LC_COLLATE* or *LANG* environment variables. In a list of words specifying file name substitution, an error results if no patterns match an existing file name. However, it is not required that every pattern match. Only the character-matching symbols *****, **?**, and **[]** indicate pattern-matching or file name expansion. The tilde (**~**) and **{ }** characters indicate file name abbreviation.

File name expansion in the C shell

The asterisk (*) character matches any string of characters, including the null string.

For example, in a directory containing the files:

```
a aa aax alice b bb c cc
```

the command **echo a*** prints all files names beginning with the character a:

```
a aa aax alice
```

Note: When file names are matched, the characters dot (.) and slash (/) must be matched explicitly.

The question mark (?) character matches any single character. The following command:

```
ls a?x
```

lists every file name beginning with the letter a, followed by a single character, and ending with the letter x:

```
aax
```

To match a single character or a range of characters, enclose the character or characters inside of [].

The following command:

```
ls [abc]
```

lists all file names exactly matching one of the enclosed characters:

```
a b c
```

Within brackets, a lexical range of characters is indicated by [a-z]. The characters matching this pattern are defined by the current collating sequence.

File name abbreviation in the C shell

The tilde (~) and { } characters indicate file name abbreviation. A ~ at the beginning of a file name is used to represent home directories. Standing alone, the ~ character expands to your home directory as reflected in the value of the *home* shell variable.

For example, the following command:

```
ls ~
```

lists all files and directories located in your **\$HOME** directory.

When the command is followed by a name consisting of letters, digits, and hyphen (-) characters, the shell searches for a user with that name and substitutes that user's **\$HOME** directory.

Note: If the ~ character is followed by a character other than a letter or slash (/), or appears anywhere except at the beginning of a word, it does not expand.

To match characters in file names without typing the entire file name, use { } around the file names. The pattern a{b,c,d}e is another way of writing abe ace ade. The shell preserves the left-to-right order and separately stores the results of matches at a low level to preserve this order. This construct might be nested. Thus, the following:

```
~source/s1/{oldls,ls}.c
```

expands to:

```
/usr/source/s1/oldls.c /usr/source/s1/ls.c
```


if the home directory for **source** is **/usr/source**. Similarly, the following:

```
../{memo,*box}
```

might expand to:

```
../memo ../box ../mbox
```

Note: memo is not sorted with the results of matching *box. As a special case, the {, }, and { } characters are passed undisturbed.

Character classes in the C shell

You can use character classes to match file names within a range indication.

The following format instructs the system to match any single character belonging to the specified class:

```
[:charclass:]
```

The following classes correspond to **ctype** subroutines:

Character Class	Definition
alnum	Alphanumeric characters
alpha	Uppercase and lowercase letters
cntrl	Control characters
digit	Digits
graph	Graphic characters
lower	Lowercase letters
print	Printable characters
punct	Punctuation character
space	Space, horizontal tab, carriage return, newline, vertical tab, or form-feed character
upper	Uppercase characters
xdigit	Hexadecimal digits

Suppose that you are in a directory containing the following files:

```
a aa aax Alice b bb c cc
```

Type the following command at a C shell prompt:

```
ls [:lower:]
```

The C shell lists all file names that begin with lowercase characters:

```
a aa aax b bb c cc
```

For more information about character class expressions, see the **ed** command.

Environment variables in the C shell

Certain variables have special meaning to the C shell. Of these, *argv*, *cwd*, *home*, *path*, *prompt*, *shell*, and *status* are always set by the shell.

Except for the *cwd* and *status* variables, the action of being set by the shell occurs only at initialization. All of the above variables maintain their settings unless you explicitly reset them.

The **cs**h command copies the *USER*, *TERM*, *HOME*, and *PATH* environment variables into the *cs*h variables, *user*, *term*, *home*, and *path*, respectively. The values are copied back into the environment whenever the normal shell variables are reset. The *path* variable cannot be set in other than in the **.cshrc** file because **cs**h subprocesses import the path definition from the environment and reexport it if changed.

The following variables have special meanings:

<i>argv</i>	Contains the arguments passed to shell scripts. Positional parameters are substituted from this variable.
<i>cdpath</i>	Specifies a list of alternate directories to be searched by the chdir or cd command to find subdirectories.
<i>cwd</i>	Specifies the full path name of the current directory.
<i>echo</i>	Set when the -x command line flag is used; when set, causes each command and its arguments to echo just before being run. For commands that are not built-in, all expansions occur before echoing. Built-in commands are echoed before command and file name substitution because these substitutions are then done selectively.
<i>histchars</i>	Specifies a string value to change the characters used in history substitution. Use the first character of its value as the history substitution character, this replaces the default character, !. The second character of its value replaces the ^ character in quick substitutions. Note: Setting the histchars value to a character used in command or file names might cause unintentional history substitution.
<i>history</i>	Contains a numeric value to control the size of the history list. Any command that is referenced within the number of events permitted is not discarded. Very large values of the <i>history</i> variable might cause the shell to run out of memory. Regardless of whether this variable is set, the C shell always saves the last command that ran on the history list.
<i>home</i>	Indicates your home directory initialized from the environment. The file name expansion of the tilde (~) character refers to this variable.
<i>ignoreeof</i>	Specifies that the shell ignore an end-of-file character from input devices that are workstations. This prevents shells from accidentally being killed when the shell reads an end-of-file character (Ctrl-D).
<i>mail</i>	Specifies the files where the shell checks for mail. This is done after each command completion which results in a prompt if a specified time interval has elapsed. The shell displays the message Mail in file if the file exists with an access time less than its change time. If the first word of the value of the <i>mail</i> variable is numeric, it specifies a different mail-checking time interval (in seconds); the default is 600 (10 minutes). If you specify multiple mail files, the shell displays the message New mail in file, when there is mail in the specified file.
<i>noclobber</i>	Places restrictions on output redirection to ensure that files are not accidentally destroyed and that redirections append to existing files.
<i>noglob</i>	Inhibits file name expansion. This is most useful in shell scripts that do not deal with file names or when a list of file names has been obtained and further expansions are not desirable.
<i>nomatch</i>	Specifies that no error results if a file name expansion does not match any existing files; rather, the primitive pattern returns. It is still an error for the primitive pattern to be malformed.
<i>notify</i>	Specifies that the shell send asynchronous notification of changes in job status. The default presents status changes just before displaying the shell prompt.
<i>path</i>	Specifies directories in which commands are sought for execution. A null word specifies the current directory. If there is no <i>path</i> variable set, then only full path names can run. The default search path (from the /etc/environment file used during login) is as follows: <code>/usr/bin /etc /usr/sbin /usr/ucb /usr/bin/X11 /sbin</code> A shell given neither the -c nor the -t flag normally hashes the contents of the directories in the <i>path</i> variable after reading the .cshrc and each time the <i>path</i> variable is reset. If new commands are added to these directories while the shell is active, you must give the rehash command. Otherwise, the commands might not be found.

<i>prompt</i>	Specifies the string displayed before each command is read from an interactive workstation input. If a ! appears in the string, it is replaced by the current event number. If the ! character is in a quoted string enclosed by single or double quotation marks, the ! character must be preceded by a \. The default prompt for users without root authority is %. The default prompt for the user with root authority is #.
<i>savehist</i>	Specifies a numeric value to control the number of entries of the history list that are saved in the ~/.history file when you log out. Any command referenced in this number of events is saved. During startup, the shell reads ~/.history into the history list, enabling history to be saved across logins. Very large values of the <i>savehist</i> variable slow down the shell startup.
<i>shell</i>	Specifies the file in which the C shell resides. This is used in forking shells to interpret files that have execute bits set, but which are not executable by the system. This is initialized to the home of the C shell.
<i>status</i>	Specifies the status returned by the last command. If the command ends abnormally, 0200 is added to the status. Built-in commands that are unsuccessful return an exit status of 1. Successful built-in commands set status to a value of 0.
<i>time</i>	Controls automatic timing of commands. If this variable is set, any command that takes more than the specified number of CPU seconds will display a line of resources used at the end of execution. For more information about the default outputs, see the built-in time command.
<i>verbose</i>	Set by the -v command line flag, this variable causes the words of each command to display after history substitution.

Input and output redirection in the C shell

Before the C shell executes a command, it scans the command line for redirection characters. These special notations direct the shell to redirect input and output.

You can redirect the standard input and output of a command with the following syntax statements:

< File	Opens the specified <i>File</i> (which is first variable, command, and file name expanded) as the standard input.
<<Word	Reads the shell input up to the line that matches the value of the <i>Word</i> variable. The <i>Word</i> variable is not subjected to variable, file name, or command substitution. Each input line is compared to the <i>Word</i> variable before any substitutions are done on the line. Unless a quoting character (\, ", ' or `) appears in the <i>Word</i> variable, the shell performs variable and command substitution on the intervening lines, allowing the \ character to quote the \$, \, and ` characters. Commands that are substituted have all blanks, tabs, and newline characters preserved, except for the final newline character, which is dropped. The resultant text is placed in an anonymous temporary file, which is given to the command as standard input.
> File	Uses the specified <i>File</i> as standard output. If <i>File</i> does not exist, it is created. If <i>File</i> exists, it is truncated, and its previous contents are lost. If the <i>noclobber</i> shell variable is set, <i>File</i> must not exist or be a character special file, or an error results. This helps prevent accidental destruction of files. In this case, use the forms including a ! to suppress this check. <i>File</i> is expanded in the same way as < input file names. The form >& redirects both standard output and standard error to the specified <i>File</i> . The following example shows how to separately redirect standard output to /dev/tty and standard error to /dev/null . The parentheses are required to allow standard output and standard error to be separate.
>!File	
>& File	
>&! File	
	<pre>% (find / -name vi -print > /dev/tty) >& /dev/null</pre>

> > <i>File</i>	Uses the specified <i>File</i> as standard output like >, but <i>appends</i> output to the end of <i>File</i> . If the <i>noclobber</i> shell variable is set, an error results if <i>File</i> does not exist, unless one of the forms including a ! is given. Otherwise, it is similar to >.
> >! <i>File</i>	
> >& <i>File</i>	
> >&! <i>File</i>	

A command receives the environment in which the shell was invoked, as changed by the input/output parameters and the presence of the command as a pipeline. Thus, unlike some previous shells, commands that run from a shell script do not have access to the text of the commands by default. Instead, they receive the original standard input of the shell. Use the << mechanism to present inline data, which allows shell command files to function as components of pipelines and also lets the shell block read its input. Note that the default standard input for a command run detached is not changed to the empty **/dev/null** file. Instead, the standard input remains the original standard input of the shell.

To redirect the standard error through a pipe with the standard output, use the form |& rather than only the |.

Flow control in the C shell

The shell contains commands that can be used to regulate the flow of control in command files (shell scripts) and (in limited but useful ways) from shell command-line input. These commands all operate by forcing the shell to repeat, or skip, in its input.

The **foreach**, **switch**, and **while** statements, and the **if-then-else** form of the **if** statement, require that the major keywords appear in a single simple command on an input line.

If the shell input is not searchable, the shell buffers input whenever a loop is being read and searches the internal buffer to do the re-reading implied by the loop. To the extent that this is allowed, backward **gotos** succeed on inputs that you cannot search.

Job control in the C shell

The shell associates a job number with each process. The shell keeps a table of current jobs and assigns them small integer numbers.

When you start a job in the background with an ampersand (&), the shell prints a line that looks like the following:

```
[1] 1234
```

This line indicates that the job number is 1 and that the job is composed of a single process with a process ID of 1234. Use the built-in **jobs** command to see the table of current jobs.

A job running in the background competes for input if it tries to read from the workstation. Background jobs can also produce output for the workstation that gets interleaved with the output of other jobs.

You can refer to jobs in the shell in several ways. Use the percent (%) character to introduce a job name. This name can be either the job number or the command name that started the job, if this name is unique. For example, if a **make** process is running as job 1, you can refer to it as %1. You can also refer to it as %make if there is only one suspended job with a name that begins with the string make. You can also use the following:

```
String
```

to specify a job whose name contains the *String* variable, if there is only one such job.

The shell detects immediately whenever a process changes its state. If a job becomes blocked so that further progress is impossible, the shell sends a message to the workstation. This message displays only after you press the Enter key. If, however, the *notify* shell variable is set, the shell immediately issues a message that indicates changes in the status of background jobs. Use the built-in **notify** command to mark a single process so that its status changes are promptly reported. By default, the **notify** command marks the current process.

C shell built-in commands

The following are C shell built-in commands.

@	Displays the value of specified shell variables.
alias	Displays specified aliases or all aliases.
bg	Puts the current or specified jobs into the background.
break	Resumes running after the end of the nearest enclosing foreach or while command.
breaksw	Breaks from a switch command.
case	Defines a label in a switch command.
cd	Changes the current directory to the specified directory.
chdir	Changes the current directory to the specified directory.
continue	Continues execution of the nearest enclosing foreach or while command.
default	Labels the default case in a switch statement.
dirs	Displays the directory stack.
echo	Writes character strings to the standard output of the shell.
else	Runs the commands that follow the second else in an if (<i>Expression</i>) then ... else if (<i>Expression2</i>) then ... else ... endif command sequence.
end	Signifies the end of a sequence of commands preceded by the foreach command.
endif	Runs the commands that follow the second then statement in an if (<i>Expression</i>) then ... else if (<i>Expression2</i>) then ... else ... endif command sequence.
endsw	Marks the end of a switch (<i>String</i>) case <i>String</i> : ... breaksw default : ... breaksw endsw command sequence. This command sequence successively matches each case label against the value of the <i>String</i> variable. Execution continues after the endsw command if a breaksw command is executed or if no label matches and there is no default.
eval	Reads variable values as input to the shell and executes the resulting command or commands in the context of the current shell.
exec	Runs the specified command in place of the current shell.
exit	Exits the shell with either the value of the status shell variable or the value of the specified expression.
fg	Brings the current or specified jobs into the foreground, continuing them if they are stopped.
foreach	Successively sets a <i>Name</i> variable for each member specified by the <i>List</i> variable and a sequence of commands, until reaching an end command.
glob	Displays list using history, variable, and file name expansion.
goto	Continues to run after a specified line.
hashstat	Displays statistics indicating how successful the hash table has been at locating commands.
history	Displays the history event list.
if	Runs a specified command if a specified expression is true.
jobs	Lists the active jobs.
kill	Sends either the TERM (terminate) signal or the signal specified by the <i>Signal</i> variable to the specified job or process.
limit	Limits usage of a specified resource by the current process and each process it creates.
login	Ends a login shell and replaces it with an instance of the /usr/sbin/login command.
logout	Ends a login shell.
nice	Sets the priority of commands run in the shell.
nohup	Causes hangups to be ignored for the remainder of a procedure.
notify	Causes the shell to notify you asynchronously when the status of the current or a specified job changes.
onintr	Controls the action of the shell on interrupts.
popd	Pops the directory stack and returns to the new top directory.

pushd	Exchanges elements of the directory stack.
rehash	Causes recomputation of the internal hash table containing the contents of the directories in the path shell variable.
repeat	Runs the specified command, subject to the same restrictions as the if command, the number of times specified.
set	Shows the value of all shell variables.
setenv	Modifies the value of the specified environment variable.
shift	Shifts the specified variable to the left.
source	Reads command specified by the <i>Name</i> variable.
stop	Stops the current or specified jobs running in the background.
suspend	Stops the shell as if a STOP signal has been received.
switch	Starts a switch (<i>String</i>) case <i>String</i> : ... breaksw default: ... breaksw endsw command sequence. This command sequence successively matches each case label against the value of the <i>String</i> variable. If none of the labels match before a default label is found, the execution begins after the default label.
time	Displays a summary of the time used by the shell and its child processes.
umask	Determines file permissions.
unalias	Discards all aliases with names that match the <i>Pattern</i> variable.
unhash	Disables the use of the internal hash table to locate running programs.
unlimit	Removes resource limitations.
unset	Removes all variables having names that match the <i>Pattern</i> variable.
unsetenv	Removes all variables from the environment whose names match the specified <i>Pattern</i> variable.
wait	Waits for all background jobs.
while	Evaluates the commands between the while and the matching end command sequence while an expression specified by the <i>Expression</i> variable evaluates nonzero.

The following is related information:

Korn shell

The **ksh** and **stty** commands.

The **alias**, **cd**, **export**, **fc**, **getopts**, **read**, **set**, and **typeset** Korn shell commands.

The **/etc/passwd** file.

Bourne shell

The **bsh** or **Rsh** command, **login** command.

The Bourne shell **read** special command.

The **setuid** subroutine, **setgid** subroutine.

The **null** special file.

The **environment** file, **profile** file format.

C shell

The **cs** command, **ed** command.

The **alias**, **unalias**, **jobs**, **notify** and **set** C Shell built-in commands.

Chapter 35. BSD systems reference

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:

Introduction to AIX for BSD system managers

The following are tips to help Berkeley Software Distribution (BSD) system managers get started managing AIX.

- Start by logging in as root at the graphics console.
- Perform system management from the system console until you become experienced with the system. It is easier to work from the system console than a remote terminal. Once you are experienced with the system, you can work remotely from an xterm or an ASCII terminal.
- Take advantage of the several AIX facilities for system management tasks. They include:
 - System Management Interface Tool (SMIT). SMIT provides an interface between system managers and configuration and management commands. SMIT can help system managers perform most system administration tasks. For more information, see the Chapter 13, “System management interface tool,” on page 271.
 - The Object Data Manager (ODM). The ODM provides routines that access objects from the ODM databases. The ODM databases contain device configuration information. For more information about how the ODM databases store device information, see Chapter 17, “Configuration of a large number of devices,” on page 283.
 - 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 Chapter 10, “System Resource Controller,” on page 239.

Major differences between 4.3 BSD and AIX

The following is a summary of the major differences between AIX and 4.3 BSD systems.

On AIX, 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
- Access rights for UNIX domain sockets.

Configuration data storage

4.3 BSD usually stores configuration data in ASCII files. Related pieces of information are kept on the same line and record processing (sorting and searching) can be done on the ASCII file itself. Records can vary in length and are terminated by a line feed. 4.3 BSD provides tools to convert some potentially large ASCII files to a database (dbm) format. Relevant library functions search the pair of dbm files if they exist, but search the original ASCII file if the dbm files are not found.

Some configuration data for AIX 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 files for AIX where the information is stored in traditional form rather than in stanza form.

Other configuration data for AIX are stored in files maintained by the Object Data Manager (ODM). Web-based System Manager or the System Management Interface Tool (SMIT) can manipulate and display information in ODM files. Alternatively, you can use the ODM commands directly to view these files. To query the ODM files, use the following commands:

- **odmget**
- **odmshow.**

The following ODM commands alter ODM files:

- **odmadd**
- **odmcreate**
- **odmdrop**
- **odmchange**
- **odmdelete.**

Attention: Altering ODM files incorrectly can cause the system to fail, and might prevent you from successfully restarting the system. Only use ODM commands directly on ODM files when task-specific commands, such as those generated by Web-based System Manager or SMIT, are unsuccessful.

Configuration management

When a system running AIX starts up, a set of configuration-specific commands are invoked by the Configuration Manager. These configuration-specific commands are called *methods*. Methods identify the devices on the system and update the appropriate ODM files in the **/etc/objrepos** directory.

Device special files in the **/dev** directly are not preinstalled. Some special files, such as those for hard disks, are created automatically during the startup configuration process. Other special files, such as those for ASCII terminals, must be created by the system administrator by using the Web-based System Manager Devices application or the SMIT **Devices** menu. This information is retained in the ODM for later use by the system.

Disk management

In 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 logical 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.

Commands in AIX used for file system and volume management include:

- **crfs**
- **varyonvg**
- **varyoffvg**
- **lsvg**
- **importvg**
- **exportvg.**

The following 4.3 BSD commands are also available:

- **mkfs**
- **fsck**
- **fsdb**

- **mount**
- **umount.**

Differences between these commands for 4.3 BSD and for AIX are discussed in “File systems for BSD 4.3 system managers” on page 530.

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 **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.

New commands

To handle new configuration and disk management systems, AIX has about 150 commands that are new to 4.3 BSD administrators. For more information, see “Commands for system administration for BSD 4.3 system managers” on page 526.

Startup

AIX supports automatic identification and configuration of devices. Consequently, the startup process is very different from 4.3 BSD systems. In addition to the kernel, an image of a boot file system and the previous base device configuration information is loaded to a RAM disk. In the first phase of startup, sufficient configuration information is loaded and checked to permit accessing logical volumes. The paging space device is identified to the kernel and the hard disk root file system is checked. At this time, the operating system changes the root file system from the RAM disk to the hard disk and completes the startup procedure, including configuring other devices.

User authorization

4.3 BSD, and versions of AT&T UNIX operating systems before SVR4, store all user authentication information, including encrypted passwords, in the **/etc/passwd** file. Traditionally, the **/etc/passwd** file could be read by all.

On SVR4 systems, encrypted passwords are removed from the **/etc/passwd** file and stored in the **/etc/shadow** file. Only users with root authority and trusted programs (such as the **/bin/login** program) can read the **/etc/shadow** file.

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 line printing system for AIX 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 can be changed by the administrator.

AIX does not support **setuid** or **setgid** for shell scripts in any shell.

Note:

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

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

BSD 4.3 system managers can back up data.

The **tar** and **cpio** commands can move data between systems. The **tar** command for AIX is not fully compatible with the 4.3 BSD **tar** command. The **tar** command for AIX requires the **-B** flag (blocking input) if it is reading from a pipe. The AT&T **cpio** command is compatible with this version.

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

```
backup -0uf Device Filesystemname
```

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

```
dump 0uf Device Filesystemname
```

Similarly, the **restore** command for AIX 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 in 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 **backup** command for AIX 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 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts* and Chapter 6, "Backup methods," on page 145.

Startup for BSD 4.3 system managers

The following discusses AIX system startup for BSD 4.3 system managers.

On 4.3 BSD systems, the **init** program is the last step in the startup procedure. The main role of the **init** program is to create processes for each available terminal port. The available terminal ports are found by reading the **/etc/tty** 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 **/etc/inittab** file by directly editing the file, using the **telinit** command, or by using the following commands:

chitab(1)	Changes records in the /etc/inittab file
lkitab(1)	Lists records in the /etc/inittab file
mkitab(1)	Makes records in the /etc/inittab file
rmitab(1)	Removes records in the /etc/inittab file

Changes made to the **/etc/inittab** file take effect the next time the system is rebooted, or when the **telinit q** command is run.

Commands for system administration for BSD 4.3 system managers

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

bosboot(1)	Initializes a boot device.
bootlist(1)	Alters the list of boot devices (or the ordering of these devices in the list) available to the system.
cfgmgr(1)	Configures devices by running the programs in /etc/methods directory.
chcons(1)	Redirects the system console to device or file, effective next startup
chdev(1)	Changes the characteristics of a device
chdisp(1)	Changes the display used by the low-function terminal (LFT) subsystem.
checkcw(1)	Prepares constant-width text for the troff command.
checkeq(1)	Checks documents formatted with memorandum macros.
checkmm(1)	Checks documents formatted with memorandum macros.
checknr(1)	Checks nroff and troff files.
chfont(1)	Changes the default font selected at boot time.
chfs(1)	Changes attributes of a file system.
chgroup(1)	Changes attributes for groups.
chgrpmem(1)	Changes the administrators or members of a group.
chhwkbd(1)	Changes the low function terminal (LFT) keyboard attributes stored in the Object Data Manager (ODM) database.
chitab(1)	Changes records in the /etc/inittab file.
chkbd(1)	Changes the default keyboard map used by the low-function terminal (LFT) at system startup.
chkey(1)	Changes your encryption key.
chlang	Sets LANG environment variable in the /etc/environment file for the next login.
chlicense(1)	There are two types of user licensing, fixed and floating. Fixed licensing is always enabled, and the number of licenses can be changed through the -u flag. Floating licensing can be enabled or disabled (on or off) through the -f flag
chlv(1)	Changes the characteristics of a logical volume
chnamsv(1)	Changes TCP/IP-based name service configuration on a host
chprtsv(1)	Changes a print service configuration on a client or server machine
chps(1)	Changes attributes of a paging space.
chpv(1)	Changes the characteristics of a physical volume in a volume group.
chque(1)	Changes the queue name.
chqueuedev(1)	Changes the printer or plotter queue device names.
chssys(1)	Changes a subsystem definition in the subsystem object class.
chtc(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.
rmpps(1)	Removes a paging space from the system.
rmque(1)	Removes a printer queue from the system.
rmqueuedev(1)	Removes a printer or plotter queue device from the system.
rmserver(1)	Removes a subserver definition from the subserver object class.
rmssys(1)	Removes a subsystem definition from the subsystem object class.
rmuser(1)	Removes a user account.
rmvfs(1)	Removes entries in the /etc/vfs file.
rmvirprt(1)	Removes a virtual printer.
savebase(1)	Saves base customized device data in the ODM onto the boot device.
swapoff(1)	Deactivates one or more paging space.
swapon(1)	Specifies additional devices for paging and swapping.
syncvg(1)	Synchronizes logical volume copies that are not current.
usrck(1)	Verifies the correctness of a user definition.
varyoffvg(1)	Deactivates a volume group.
varyonvg(1)	Activates a volume group.

Cron for BSD 4.3 system managers

The **cron** daemon for this operating system is similar to the System V Release 2 **cron** daemon.

An entry in the **/etc/inittab** file starts the **cron** daemon.

Devices for BSD 4.3 system managers

The following discusses 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 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 can mean that the hardware is faulty or it can mean that the configuration database in the **/etc/objrepos** directory is damaged.

The **cfgmgr** command processes the configuration database in the **/etc/objrepos** directory and is processed at startup time by the **cfgmgr** command (the Configuration Manager).

The pseudocode below shows the Configuration Manager logic:

```

/* Main */
While there are rules in the Config_Rules database
    {
        Get the next rule and execute it
        Capture stdout from the last execution
        Parse_Output(stdout)
    }
/* Parse Output Routine */
/* stdout will contain a list of devices found */
Parse_OutPut(stdout)
    {
        While there are devices left in the list
            {
                Lookup the device in the database
                if (!defined)
                    Get define method from database and execute
                if (! configured)
                    {
                        Get config method from database and execute
                        Parse_Output(stdout)
                    }
            }
    }

```

File comparison table for 4.3 BSD, SVR4, and AIX

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

File Comparison Table				
4.3 BSD File	SVR4 File	File for AIX	Database	Type (odm/dbm)
L-Devices	Devices	Devices	no	
L-dialcodes	Dialcodes	Dialcodes	no	
L.cmds	Permissions	Permissions	no	
L.sys	Systems	Systems	no	
USERFILE	Permissions	Permissions	no	
aliases	mail/namefiles	aliases	aliasesDB/DB	dbm
fstab	vfstab	filesystems	no	
ftputers	ftputers	ftputers	no	
gettytab		N/A		
group	group	group	no	

File Comparison Table				
4.3 BSD File	SVR4 File	File for AIX	Database	Type (odm/dbm)
hosts	hosts	hosts	no	
hosts.equiv	hosts.equiv	hosts.equiv	no	
inetd.conf	inetd.conf	inetd.conf	no	
map3270	N/A	map3270	no	
motd	motd	motd	no	
mtab	mnttab	N/A	no	
named.boot	named.boot	named.boot	no	
named.ca		named.ca	no	
named.hosts		named.data (See note)	no	
named.local		named.local	no	
named.pid	named.pid	named.pid	no	
named.rev		named.rev	no	
networks	networks	networks	no	
passwd	passwd	passwd	no	
printcap	qconfig	qconfig		
protocols		protocols	no	
remote	remote	remote	no	
resolv.conf	resolv.conf	resolv.conf	no	
sendmail.cf	sendmail.cf	sendmail.cf	sendmail.cfDB	neither
services		services	no	
shells	shells	N/A		
stab		N/A		
syslog.conf		syslog.conf	no	
syslog.pid		syslog.pid	no	
termcap	terminfo	terminfo		
ttys	ttys	N/A	yes	odm
types		N/A		
utmp	utmp	utmp		
vfont		N/A		
vgrindefs		vgrindefs		
wtmp	wtmp	wtmp		

Note: The file names **named.ca**, **named.hosts**, **named.local**, and **named.rev** are user definable in the **named.boot** file. However, these are the names used for these files in the documentation for AIX.

File systems for BSD 4.3 system managers

Similar commands are used to mount and unmount file systems.

AIX 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 store 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 might be interested in the *check* variable in the **/etc/filesystems** file. The *check* variable can be set to the value `True`, `False`, or to a number. For example, you can specify `check=2` in the **/etc/filesystems** file. The number specifies the pass of the **fsck** command that will check this file system. The *check* parameter corresponds to the fifth field in an **/etc/fstab** file record.

There is no dump frequency parameter in the **/etc/filesystems** file.

File system support on AIX

AIX supports various file systems.

AIX supports disk quotas.

AIX does not allow mounting of diskettes as file systems.

The syntax of the **mount** and **umount** commands for AIX differs from 4.3 BSD and from SVR4 versions of these commands. The commands to mount and unmount all file systems at once are shown for all three systems in the following table:

mount and unmount Commands

Function	Syntax for this operating system	4.3 BSD Syntax	SVR4 Syntax
mount all file systems	mount all	mount -a	mountall
umount all file systems	umount all	umount -a	umountall

See Chapter 5, “File systems,” on page 93 for more information.

Finding and examining files for BSD 4.3 system managers

The following is a list of the BSD file commands that AIX supports.

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 can be used to simulate the functionality:

```
#!/bin/bsh
PATH=/bin
for dir in /bin /etc /lib /usr
do
find $dir -print | egrep $1
done
```

The syntax for the **ffind** script is:

```
ffind Filename
```

Paging space for BSD 4.3 system managers

The following commands assist in managing paging space (also known as swap space).

chps(1)	Changes attributes of a paging space
lsp(1)	List attributes of a paging space
mkps(1)	Add an additional paging space to the system
rm(1)	Removes a paging space from the system
swapoff(1)	Deactivates one or more paging spaces
swapon(1)	Specifies additional devices for paging and swapping

If a large paging space is required, place one paging logical volume for each hard disk. This allows scheduling of paging across multiple disk drives.

Changing the default startup to permit 4.3 BSD ASCII configuration

You can administer network interfaces for AIX through the SMIT and ODM files, or through 4.3 BSD ASCII configuration files.

To administer network interfaces through 4.3 BSD ASCII configuration files, uncomment the commands in the **/etc/rc.net** file below the heading:

```
# Part II - Traditional  
Configuration
```

Then if you want flat file configuration and SRC support, edit the **/etc/rc.net** file and uncomment the **hostname**, **ifconfig**, and **route** commands with the appropriate parameters.

If you want flat file configuration without SRC support, use the **smit setbootup_option** fast path to change the system to BSD-style **rc** configuration. This option configures the system to use the **/etc/rc.bsdnet** file at startup. You also have to edit the **/etc/rc.bsdnet** file and uncomment the **hostname**, **ifconfig**, and **route** commands with the appropriate parameters.

Additional options for ifconfig and netstat commands

The following is a list of additional options for the **ifconfig** and **netstat** commands.

The **ifconfig** command for AIX has the following additional options:

mtu The *mtu* variable specifies the maximum transmission unit (MTU) used on the local network (and local subnets) and the MTU used for remote networks. To maximize compatibility with Ethernet and other networks, set both the Token-Ring and Ethernet default *mtu* value to 1500.

allcast

The **allcast** flag sets the Token-Ring broadcast strategy. Setting the **allcast** flag optimizes connectivity through Token-Ring bridges. Clearing the **allcast** flag (by specifying **-allcast**) minimizes excess traffic on the ring.

The **netstat** command for AIX 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. For more information about the **ifconfig** and **netstat** commands, see **ifconfig** and **netstat**.

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 the running of several unsecured TCP/IP programs, such as the ftpp , 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: <ul style="list-style-type: none"> dogticks Sets timer granularity for ifwatchdog routines subnetsarelocal Determines if packet address is on the local network ipsendredirects Specifies whether the kernel should send redirect signals ipforwarding Specifies whether the kernel should forward packets tcp_ttl Specifies the time-to-live for Transmission Control Protocol (TCP) packets udp_ttl Specifies the time-to-live for User Datagram Protocol (UDP) packets maxttl Specifies the time-to-live for Routing Information Protocol (RIP) packets ipfragttl Specifies the time-to-live for Internet Protocol (IP) fragments lowclust Specifies a low water mark for cluster mbuf pool lowmbuf Specifies a low water mark for the mbuf pool thewall Specifies the maximum amount of memory that is allocated to the mbuf and cluster mbuf pool arpt_killc Specifies the time in minutes before an inactive complete Address Resolution Protocol (ARP) entry is deleted
iptrace	The iptrace command provides interface-level packet tracing for Internet protocols.
ipreport	The ipreport command formats the trace into human-readable form. An example of using this command is the following: <pre>iptrace -i en0 /tmp/iptrace.log # kill iptrace daemon kill `ps ax grep iptrace awk '{ print \$1 }'` ipreport /tmp/iptrace.log more</pre>

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.

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 **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.

Note:

- 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 in **/usr/man/man?** files.
- 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.

For more information about these commands, see **man**, **apropos**, **whatis**, and **catman**.

NFS and NIS (formerly Yellow Pages) for BSD 4.3 system managers

The following describes NFS and NIS for BSD 4.3 system managers.

Network File System (NFS) and Network Information Services (NIS) daemons are started from the **/etc/rc.nfs** file. However, before the NFS and NIS daemons can be started, the **portmap** daemon must be started in the **/etc/rc.tcpip** file. By default, the **/etc/rc.nfs** file is not invoked by the **/etc/inittab** file. If you add a line in the **/etc/inittab** file to invoke the **/etc/rc.nfs** script, it should be invoked after the **/etc/rc.tcpip** script.

If NIS is active, include a root entry prior to the **+::** (plus sign, colon, colon) entry in the **/etc/passwd** file and a system entry prior to the **+::** entry in the **/etc/group** file. This allows a system administrator to log in as root and make changes if the system is unable to communicate with the NIS server.

NFS can be configured by using Web-based System Manager (type **wsm**, then select **Network**), or the SMIT fast path, **smit nfs**. The Web-based System Manager and SMIT menus refer to NIS (formerly Yellow Pages) as NIS. Many of the NFS and NIS commands are found in the **/etc** and **/usr/etc** directory.

Some NFS environments use an **arch** command to identify machine families and types of machines. For example if you are using the IBM RS/6000, specify the **power** identifier for family (CPU), and the **ibm6000** identifier for type (machine).

User passwords for BSD 4.3 system managers

When you use the **/bin/passwd** command for AIX as the root user, you are prompted for the current root user password.

An example of using the **/bin/passwd** command follows:

```
# passwd cslater
Changing password for "cslater"
Enter root's Password or
cslater's Old password:
cslater's New password:
Re-enter cslater's
new password:
#
```

The 4.3 BSD version does not prompt for the current root user password. An example of the 4.3 BSD version follows:

```
# passwd cslater
New password:
Retype new password:
#
```

Importing a BSD 4.3 password file

You can import a BSD 4.3 password file into AIX.

To import a BSD 4.3 password file, perform the following steps:

1. Copy the BSD 4.3 password file to the **/etc/passwd** file and enter:

```
pwdck -y ALL
```

2. Update the **/etc/security/limits** file 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. For more information about the **usrck** command, see **usrck**.

Attention: If the **/etc/security/limits** file is modified, the stack must not exceed 65,536 bytes. If it does, running the **usrck** command can cause problems. Change the stack size to 65,536 and run **usrck** command again.

3. Run the **grpck** and **usrck** commands to verify group and user attributes.

Editing the password file for BSD 4.3 system managers

The following explains how to change entries in the password file and how to administer passwords on AIX in a BSD 4.3 manner.

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.

For more information about these commands, see **lsuser**, **mkuser**, **chuser**, and **rmuser**.

Note: Using an editor to change several user name entries at one time requires editing of several files simultaneously, because passwords are stored in the **/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 in AIX in a BSD 4.3 manner. Use the following procedure:

1. Put a BSD 4.3 password file in the **/etc/shadow** file.
 2. Change the permissions to the file by entering:
- ```
chmod 000 /etc/shadow
```
3. Place the following **vipw** shell script in the **/etc** directory:

```


#!/bin/bsh

vipw. Uses pwdck for now. May use usrck someday

PATH=/bin:/usr/bin:/etc:/usr/ucb # Add to this if your editor is
 # some place else
if [-f /etc/ptmp] ; then
 echo "/etc/ptmp exists. Is someone else using vipw?"
 exit 1
fi
if [! -f /`which "$EDITOR" | awk '{ print $1 }'`] ; then
 EDITOR=vi
fi
cp /etc/shadow /etc/ptmp
if (cmp /etc/shadow /etc/ptmp) ; then
 $EDITOR /etc/ptmp
else
 echo cannot copy shadow to ptmp
```

```

 exit 1
 fi
 if (egrep "^root:" /etc/ptmp >/dev/null) ; then
 cp /etc/ptmp /etc/shadow ; cp /etc/ptmp /etc/passwd
 chmod 000 /etc/passwd /etc/shadow
 pwdck -y ALL 2>1 >/dev/null # return code 114 may change
 rc=$?
 if [$rc -eq 114]; then
 chmod 644 /etc/passwd
 rm -f /etc/passwd.dir /etc/passwd.pag
 mkpasswd /etc/passwd
 # update /etc/security/limits, or ftp
 # will fail
 else
 pwdck -y ALL
 fi
 fi
else
 echo bad entry for root in ptmp
fi
rm /etc/ptmp

```

4. If you use the **vipw** shell script or the **mkpasswd** command, be aware that Web-based System Manager, SMIT, and the **mkuser**, **chuser**, and **rmuser** commands do not use the **mkpasswd** command. You must run:

```
mkpasswd /etc/passwd
```

to update the **/etc/passwd.dir** and **/etc/passwd.pag** files.

**Attention:** Initialization of the *IFS* variable and the trap statements guard against some of the common methods used to exploit security holes inherent in the **setuid** feature. However, the **vipw** and **passwd** shell scripts are intended for relatively open environments where compatibility is an important consideration. If you want a more secure environment, use only the standard commands for AIX.

5. Put the following **passwd** shell script in the **/usr/ucb** directory:

```

#!/bin/ksh
#
matches changes to /etc/security/passwd file with changes to
#/etc/shadow
#
IFS=" "
PATH=/bin
trap "exit 2" 1 2 3 4 5 6 7 8 10 12 13 14 15 16 17 18 21 22 \
 23 24 25 27 28 29 30 31 32 33 34 35 36 60 61 62
if [-n "$1"]; then
 USERNAME=$1
else
 USERNAME=$LOGNAME
fi
if [-f /etc/ptmp]; then
 echo password file busy
 exit 1
fi
trap "rm /etc/ptmp; exit 3" 1 2 3 4 5 6 7 8 10 12 13 \
 14 15 16 17 18 21 22 23 24 25 27 28 29 30 31 \
 32 33 34 35 36 60 61 62
if (cp /etc/security/passwd /etc/ptmp) ; then
 chmod 000 /etc/ptmp else
 rm -f /etc/ptmp exit 1
fi
if (/bin/passwd $USERNAME) ; then
 PW=`awk ' BEGIN { RS = "" }
 $1 == user { print $4 } ' user="$USERNAME:" \
/etc/security/passwd `
else

```

```

 rm -f /etc/ptmp
 exit 1
fi
rm -f /etc/ptmp
awk -F: '{ $1 == user { print $1:"pw":$3 ":"$4:"$5":$6:"$7 }
 $1 != user { print $0 }' user="$USERNAME" pw="$PW" \
 /etc/shadow > /etc/ptmp
chmod 000 /etc/ptmp
mv -f /etc/ptmp /etc/shadow

```

6. Change the permissions to the **passwd** script by entering:
 

```
chmod 4711 /usr/ucb/passwd
```
7. Ensure that each user *PATH* environmental variable specifies that the **/usr/ucb** directory be searched before the **/bin** directory.

---

## Performance measurement and tuning for BSD 4.3 system managers

The following discusses AIX device attributes and performance measurement and tuning.

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 might be too low for users who have many windows open simultaneously. The following command can be used to change the value systemwide:

```
hdev -l sys0 -a maxuproc=100
```

This example changes the maximum number to 100. The new value is set once the system has restarted.

To view the current setting of this and other system attributes, type:

```
lsattr -E -l sys0
```

The **maxmbuf** attribute is not currently supported by the mbuf services.

AIX supports the **vmstat** and **iostat** commands, but not the **sysstat** command or load averages. For more information about these commands, see **vmstat** and **iostat**.

---

## Printers for BSD 4.3 system managers

In AIX 5.1 and later, the operating system supports two printing subsystems: 4.3 BSD and System V.

The System V style of printing subsystem uses System V Release 4 commands, queues, and files and is administered the same way. The following paragraphs describe what you need to know to manage the 4.3 BSD style of printing subsystem. You control which subsystem is made active through SMIT. Only one subsystem can be active at a time.

Printing is managed by programs and configurations in the **/usr/lpd** directory. The design, configuration, queueing mechanism, and daemon processes of the 4.3 BSD and printer subsystems for AIX are different. However, they both use the **lpd** protocol for remote printing. Both systems use **/etc/hosts.lpd**, if it exists,

or **/etc/host.equiv** otherwise. The printer subsystem for AIX offers a gateway to 4.3 BSD printer subsystems, so systems using AIX 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 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 in AIX, 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:

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| <b>cancel(1)</b>     | Cancels requests to a line printer                                         |
| <b>chqueuedev(1)</b> | Changes the printer or plotter queue device names                          |
| <b>chvirprt(1)</b>   | Changes the attribute values of a virtual printer                          |
| <b>disable(1)</b>    | Disables a printer queue                                                   |
| <b>enable(1)</b>     | Enables a printer queue                                                    |
| <b>hplj(1)</b>       | Postprocesses <b>troff</b> output for HP LaserJetII with the K cartridge   |
| <b>ibm3812(1)</b>    | Postprocesses <b>troff</b> output for IBM 3812 Mod 2 Pageprinter           |
| <b>ibm3816(1)</b>    | Postprocesses <b>troff</b> output for IBM 3816 Pageprinter                 |
| <b>ibm5587G(1)</b>   | Postprocesses <b>troff</b> output for IBM 5587G with 32x32/24x24 cartridge |
| <b>lp(1)</b>         | Sends requests to a line printer                                           |
| <b>lpr(1)</b>        | Enqueues print jobs                                                        |
| <b>lprm(1)</b>       | Removes jobs from the line printer spooling queue                          |
| <b>lpstat(1)</b>     | Displays line printer status information                                   |
| <b>lptest(1)</b>     | Generates the line printer ripple pattern                                  |
| <b>lsallqdev(1)</b>  | Lists all configured printer queue device names within a queue             |
| <b>lsvirprt(1)</b>   | Displays the attribute values of a virtual printer                         |
| <b>mkque(1)</b>      | Adds a printer queue to the system                                         |
| <b>mkqueuedev(1)</b> | Adds a printer queue device to the system                                  |
| <b>mkvirprt(1)</b>   | Makes a virtual printer                                                    |
| <b>pac(1)</b>        | Prepares printer/plotter accounting records                                |
| <b>piobe(1)</b>      | Print Job Manager for the printer backend                                  |
| <b>pioburst(1)</b>   | Generates burst pages (header and trailer pages) for printer output        |
| <b>piocmdout(3)</b>  | Subroutine that outputs an attribute string for a printer formatter        |
| <b>pidigest(1)</b>   | Digests attribute values for a virtual printer definition and stores       |
| <b>pioexit(3)</b>    | Subroutine that exits from a printer formatter                             |
| <b>pioformat(1)</b>  | Drives a printer formatter                                                 |
| <b>piofquote(1)</b>  | Converts certain control characters destined for PostScript printers       |



|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| <b>piogetstr(3)</b>  | Subroutine that retrieves an attribute string for a printer formatter                  |
| <b>piogetvals(3)</b> | Subroutine that initializes Printer Attribute database variables for printer formatter |
| <b>piomsgout(3)</b>  | Subroutine that sends a message from a printer formatter                               |
| <b>pioout(1)</b>     | Printer backend's device driver interface program                                      |
| <b>piopredef(1)</b>  | Creates a predefined printer data stream definition                                    |
| <b>proff(1)</b>      | Formats text for printers with personal printer data streams                           |
| <b>qadm(1)</b>       | Performs system administration for the printer spooling system                         |
| <b>qconfig(4)</b>    | Configures a printer queueing system                                                   |
| <b>qstatus(1)</b>    | Provides printer status for the print spooling system                                  |
| <b>restore(3)</b>    | Restores the printer to its default state                                              |
| <b>rmque(1)</b>      | Removes a printer queue from the system                                                |
| <b>rmquedev(1)</b>   | Removes a printer or plotter queue device from the system                              |
| <b>rmvirprt(1)</b>   | Removes a virtual printer                                                              |
| <b>splp(1)</b>       | Changes or displays printer driver settings                                            |
| <b>xpr(1)</b>        | Formats a window dump file for output to a printer                                     |

---

## Terminals for BSD 4.3 system managers

The following discusses 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, 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 is to be configured. When using SMIT, the first planar serial port (physically labeled **s1**) is referred to as location **00-00-S1**, adapter **sa0**, and port **s1** in the SMIT menus. The second planar serial port (physically labeled **s2**) is referred to as location **00-00-S2**, adapter **sa1**, and port **s2**.

Use the **penable** and **pdisable** commands to enable and disable a port.

## termcap and terminfo

Like System V, this operating system uses **terminfo** entries in **/usr/lib/terminfo/?/\*** files.

Users with 4.3 BSD Systems might find the following commands helpful:

### **captainfo(1)**

Converts a **termcap** file to a **terminfo** file

**tic(1)** Translates the **terminfo** files from source to compiled format.

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

---

## UUCP for BSD 4.3 system managers

Web-based System Manager provides System V Basic Networking Utilities (BNU) which are often referred to as the HDB UUCP. The following table lists lists UUCP commands and files.

|                     |                                                                             |
|---------------------|-----------------------------------------------------------------------------|
| <b>Dialers(4)</b>   | Lists modems used for BNU remote communications links                       |
| <b>Maxuuxqts(4)</b> | Limits the number of instances of the BNU <b>uuxqt</b> daemons that can run |

|                          |                                                                        |
|--------------------------|------------------------------------------------------------------------|
| <b>Permissions(4)</b>    | Specifies BNU command permissions for remote systems                   |
| <b>Poll(4)</b>           | Specifies when the BNU program should poll remote systems              |
| <b>Systems(4)</b>        | Lists remote computers with which the local system can communicate     |
| <b>rmail(1)</b>          | Handles remote mail received through BNU                               |
| <b>uuccheck(1)</b>       | Checks for files and directories required by BNU                       |
| <b>uuclean(1)</b>        | Removes files from the BNU spool directory                             |
| <b>uucleanup(1)</b>      | Deletes selected files from the BNU spooling directory                 |
| <b>uucpadmin(1)</b>      | Enters basic BNU configuration information                             |
| <b>uudemon.admin(1)</b>  | Provides periodic information on the status of BNU file transfers      |
| <b>uudemon.cleanu(1)</b> | Cleans up BNU spooling directories and log files                       |
| <b>uudemon.hour(1)</b>   | Initiates file transport calls to remote systems using the BNU program |
| <b>uudemon.poll(1)</b>   | Polls the systems listed in the BNU Poll file                          |
| <b>uulog(1)</b>          | Provides information about BNU file-transfer activities on a system    |
| <b>uupoll(1)</b>         | Forces a poll of a remote BNU system                                   |
| <b>uuq(1)</b>            | Displays the BNU job queue and deletes specified jobs from the queue   |
| <b>uusnap(1)</b>         | Displays the status of BNU contacts with remote systems                |
| <b>uustat(1)</b>         | 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 information about these commands, see **uuencode** and **uudecode**.

For more information, see the lists in BNU file and directory structure.

---

## Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Dept. LRAS/Bldg. 003  
11400 Burnet Road  
Austin, TX 78758-3498  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

---

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX  
AIX 5L  
CICS  
DB2  
DFS  
DirectTalk  
Encina  
Extended Services  
HACMP  
IBM  
ibm.com  
Micro Channel  
MQSeries  
OS/2  
Portmaster  
POWER  
POWER5  
RS/6000  
SecureWay  
TXSeries

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Other company, product, or service names may be the trademarks or service marks of others.



---

# Index

## Special characters

- : built-in command
  - Bourne shell 488
  - Korn or POSIX shell 461
- / (root) file system 96
- /dev/null file 373
- /etc/environment file 425
- /etc/hosts 18
- /etc/inittab file
  - changing 15
- /etc/passwd file 436
- /etc/profile file 173, 426
- /etc/security/passwd file 412
- /export directory 101
- /opt file system 94
- /proc file system 94
- /usr/bin/ksh93 479
- /usr/bin/psh command 436
- /usr/bin/sh command 436
- /usr/share directory 99
- /var file system 100
- . built-in command
  - Bourne shell 488
  - Korn or POSIX shell 461
- .env file 427
- .hushlogin file 335
- .mwmrc file 429
- .profile file 173, 426
- .Xdefaults file 428
- .xinitrc file 428
- \$HOME directory 138
- @ built-in command
  - C shell 501
- ~ (home) directory 138

## Numerics

- 64-bit mode
  - filesystems 181

## A

- absolute path name 139
- access control
  - displaying information 421
  - editing information 423
  - setting information 422
- access control lists 416
  - example 422
  - example for AIXC ACL 419
  - for file system objects 417
  - maintaining 416
- access modes
  - controlling 413
  - directories 413
  - displaying group information 415
  - files 413

- access modes (*continued*)
  - numeric representation of 414
  - symbolic representation of 413
  - user classes 413
- accessing a system that will not boot 7
- accounting system
  - BSD System Managers 524
  - commands
    - overview 263
    - running automatically 263
    - running from the keyboard 264
- connect-time data
  - collecting 257
  - displaying 249
  - reporting 257
- CPU usage
  - displaying 248, 249
- disk-usage data 258
  - displaying 250
  - reporting 259
- failure
  - recovering from 266
- fees
  - charging 259
  - reporting 259
- files
  - data files 265
  - formats 268
  - overview 265
  - report and summary files 265
  - runacct command files 267
- holidays file
  - updating 256
- overview 245
- printer-usage data 259
  - displaying 250
- problems
  - fixing bad times 252
  - fixing incorrect file permissions 252
  - fixing out-of-date holidays file 256
  - fixing runacct errors 253
- process data
  - collecting 257
  - reporting 258
- reporting data
  - overview 260
- reports
  - daily 260, 261
  - fiscal 259
  - monthly 262, 263
- runacct command
  - restarting 266
  - starting 266
- setting up 245
- summarizing records 262
- system activity data
  - displaying 247
  - displaying while running a command 247

- accounting system (*continued*)
  - system activity data (*continued*)
    - reporting 260
  - tacct errors
    - fixing 251
  - wtmp errors
    - fixing 251
- ACL Type
  - AIXC 417
  - NFS4 419
- acledit command 416, 423
- aclget command 416, 421
- aclput command 416, 422
- ACLs 416
  - example 422
  - example for AIXC ACL 419
  - for file system objects 417
  - maintaining 416
- adapter location codes 289
- AIX
  - overview for BSD system managers
    - paging space 532
- aixterm command 346
- AIXwindows
  - starting Window Manager 427
  - startup files 427
- alias built-in command
  - C shell 501
  - Korn or POSIX shell 447, 466
- alias command 360
- alias substitution
  - C shell 511
- aliases
  - creating 447
  - exporting 447
  - listing 447
  - not supported 447
  - r 358, 359
  - removing 447
  - tracked 448
- aliasing
  - Korn or POSIX shell 447
- allocation group size 129
- allocations, file zero (kproc) 131
- API
  - Workload Manager (WLM) 231
- append redirection operator 372
- application programming interface
  - Workload Manager (API) 231
- arguments
  - in commands 354
- arithmetic evaluation
  - Korn or POSIX shell 454
- ASCII terminals
  - adding 348
- ASCII to PostScript
  - automating conversion 408, 409
  - converting files 408, 409
  - printing 408
- assigning
  - values and attributes 449

- at command 367, 368
- atq command 367, 368
- attributes
  - supported by Korn or POSIX shell 449
- authentication 412
- authorization 420
- availability
  - for adapter or power supply failure 49
  - for disk failure 48
- awk command 381

## B

- background processes 362
- backup 150, 151
  - BSD system managers 525
  - commands, list of 145
  - compressing files 159
  - compressing files before 159
  - effect of fragments on 134
  - files 145
  - for security 411
  - implementing with scripts 153
  - media 147
  - methods 145
  - overview 145
  - performing regularly scheduled 153
  - policy 146
  - procedure for system and user data 148
  - procedure for user file systems 149
  - replicating a system (cloning) 149
  - restoring data 162
  - restoring files 166
  - strategy for managing
    - guidelines for 146
    - planning 148
  - user file systems 149
  - user files 149
  - user-defined volume group 151
  - using smit command 159
- backup command 147, 158
- banner command 377
- batch processes 367
- bg built-in command
  - C shell 501
  - Korn or POSIX shell 466
- bidirectional languages 346
- binding a process to a processor 191
- blanks
  - definition 437
  - interpretation of 495
- blocks
  - performance costs of 129
- boot image
  - creating 12
- boot images
  - creating 12
- boot processing
  - phases of 17
- booting
  - BSD System Managers 525



- booting (*continued*)
  - crashed system 7
  - diagnosing problems 11
  - from hard disk for maintenance 6
  - rebooting a running system 5
  - understanding
    - maintenance mode 18
    - overview 16
    - RAM file system 19
    - system boot processing 17
  - uninstalled system 5
- Bourne shell 436
  - built-in commands 488
  - character classes 497
  - command substitution 492
  - commands 485
  - compound commands 487
  - conditional substitution 496
  - environment 483
  - file name substitution 497
  - list of built-in commands 499
  - pattern matching 497
  - positional parameters 497
  - predefined variables 495
  - quoting characters 486
  - redirecting input and output 498
  - reserved words 487
  - signal handling 487
  - starting 483
  - user-defined variables 492
  - variable substitution 492
  - variables 493
- break built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 461
- breaksw built-in command
  - C shell 501
- BSD 524, 533, 534, 537, 539
  - comparison for system managers 521
    - accounting 524
    - backup 525
    - boot and startup 525
    - commands 526
    - cron 528
    - devices 528
    - file comparison 529
    - file systems 530
    - finding and examining files 531
    - networking 521, 532, 533
    - performance 537
    - printers 537
    - UUCP 539
  - comparison to AIX for system managers
    - paging space 532
  - comparison to system managers
    - NFS and NIS (formerly Yellow Pages) 534
    - online documentation and man command 533
- bsh command 436, 483, 484
- built-in commands
  - : 461, 488
- built-in commands (*continued*)
  - . 461, 488
  - @ 501
  - alias 447, 466, 501
  - bg 466, 501
  - Bourne shell 488, 499
  - break 461, 488, 501
  - breaksw 501
  - C shell 500, 501
  - case 501
  - cd 466, 488, 501
  - chdir 501
  - command 466
  - continue 461, 488, 501
  - default 501
  - definition 437
  - dirs 501
  - echo 466, 488, 501
  - else 501
  - end 501
  - endif 501
  - endsw 501
  - eval 461, 488, 501
  - exec 443, 461, 488, 501
  - exit 461, 488, 501
  - export 443, 460, 461, 488
  - fc 359, 445, 466
  - fg 466, 501
  - foreach 501
  - getopts 466
  - glob 501
  - goto 501
  - hangups 501
  - hash 488
  - hashstat 501
  - history 501
  - if 501
  - jobs 466, 501, 518
  - kill 466, 501
  - Korn or POSIX shell 460
  - let 454, 466
  - limit 501
  - login 501
  - logout 501
  - newgrp 461
  - nice 501
  - notify 501
  - onintr 501
  - popd 501
  - print 466
  - pushd 501
  - pwd 466, 488
  - read 466, 488, 492
  - readonly 460, 461, 488
  - regular 460, 466, 488
  - rehash 501
  - repeat 501
  - return 461, 488
  - set 461, 488, 497, 501
  - setenv 501
  - setgroups 466

- built-in commands (*continued*)
  - setsenv 466
  - shift 461, 488, 497, 501
  - source 501
  - special 460, 461, 488
  - stop 501
  - suspend 501
  - switch 501
  - test 466, 488
  - time 501
  - times 461, 488
  - trap 461, 488
  - type 488
  - typeset 443, 449, 454, 460, 461
  - ulimit 466, 488
  - umask 466, 488, 501
  - unalias 447, 466, 501
  - unhash 501
  - unlimit 501
  - unset 461, 488, 501
  - unsetenv 501
  - wait 466, 488, 501
  - whence 466
  - while 501
- bytes
  - counting number of 388

## C

- C shell 436
  - alias substitution 511
  - built-in commands 500, 501
  - command execution 508
  - command substitution 508
  - commands 500
  - environment variables 515
  - expressions 506
  - file name substitution 513
  - history lists 509
  - history substitution 509
  - job control 518
  - limitations 500
  - list of built-in commands 519
  - operators 506
  - redirecting input and output 517
  - signal handling 500
  - starting 499
  - startup files 499
  - variable substitution 512
- cables
  - checking connections 319
- canceling
  - foreground processes 365
  - print jobs 402
- capture command 376
- case built-in command
  - C shell 501
- cat command 372, 376, 386, 392
- cd built-in command
  - Bourne shell 488
  - C shell 501

- cd built-in command (*continued*)
  - Korn or POSIX shell 466
- cd command 138, 140
- CD-ROM
  - file systems 105
- CD-ROM File System (CDRFS) 123
- CDPATH variable 452
- CDRFS file systems 105
- changing
  - control keys 432
  - default font 431
  - defaults 428
  - permissions 416
  - priority of processes 365
  - system prompt 432
  - to another directory 140
  - to become another user 334
- character classes
  - Bourne shell 497
- character-display terminals
  - adding 348
- characters
  - quoting in Korn or POSIX shell 445
- chdev command 317
- chdir built-in command
  - C shell 501
- checking
  - integrity of tapes 158
  - process status 363
  - status of print jobs 405
- checking file systems for inconsistencies 107
- chfont command 431
- chgrp command 422
- chmod command 414, 416
- chown command 412, 413, 422
- chpq command 409
- classes
  - user 413
- clear command 375
- clearing your screen 375
- clock
  - resetting 183
- clock battery 183
- colrm command 391
- COLUMNS variable 452
- combining commands 353
- command aliasing
  - Korn or POSIX shell 447
  - tilde substitution 448
- command built-in command
  - Korn or POSIX shell 466
- command flags 353
- command history
  - editing 359
  - Korn or POSIX shell 445
  - substitution 445
- command substitution
  - Bourne shell 492
  - C shell 508
  - Korn or POSIX shell 453
- command summaries 369

command summaries *(continued)*  
 backup files 169  
 directories 144  
 file security 424  
 file systems 144  
 files 395  
 I/O redirection 377  
 login names 339  
 passwords 339  
 printing 409  
 storage media 169  
 system IDs 339  
 system information 346  
 system security 424  
 user environment 346  
 commands 353  
 /usr/bin/psh 436  
 /usr/bin/sh 436  
 > 371  
 >> 372  
 < 373  
 | 374  
 acledit 416, 423  
 aclget 416, 421  
 aclput 416, 422  
 aixterm 346  
 alias 360  
 at 367, 368  
 atq 367, 368  
 awk 381  
 backup 147, 158  
 banner 377  
 Bourne shell 485  
 Bourne shell built-in 488  
 bsh 436, 483, 484  
 C shell 500  
 C shell built-in 500, 501  
 capture 376  
 cat 372, 376, 386, 392  
 cd 138, 140  
 chdev 317  
 chfont 431  
 chgrp 422  
 chmod 414, 416  
 chown 412, 413, 422  
 chpq 409  
 clear 375  
 colrm 391  
 combining 353  
 compound Korn shell 442  
 compress 159, 160  
 cp 141, 383  
 cpio 147  
 cpio -i 157  
 cpio -o 156  
 creating shortcut names 360  
 csh 436, 499  
 cut 389  
 date 184  
 definition 437  
 del 393

commands *(continued)*  
 df 102  
 diag 183  
 diff 388  
 dircmp 144  
 dosdel 395  
 dosdir 395  
 dosread 394  
 doswrite 394  
 echo 376  
 enq 397  
 env 345  
 exit 335  
 export 431  
 fdformat 155  
 file 385  
 find 158, 384  
 flags 354  
 flcopy 156  
 for BSD System Managers 526  
 format 155  
 fsck 146, 155  
 grep 9, 374, 386  
 groups 412  
 head 389  
 history 357  
 id 334, 337, 412  
 kill 9, 192, 368  
 Korn or POSIX shell 440  
 Korn or POSIX shell built-in 460  
 ksh 157, 436, 443  
 ln 137, 392  
 lock 423  
 login 334, 420  
 logname 336  
 logout 335  
 lp 397  
 lpr 397  
 ls 141, 412, 413  
 lsattr 317  
 lscfg 341  
 lscons 343  
 lsdev 317  
 lsdisp 343  
 lsfont 343  
 lsgroup 415  
 lskbd 344  
 lslpp 344  
 man 356  
 mkdev 317  
 mkdir 139  
 mkvirprt 397  
 more 386  
 mv 383  
 mvdir 140  
 mwm 427  
 names 353  
 nice 365  
 nl 391  
 overview 353  
 pack 159, 161

commands (*continued*)

- page 386
- parameters 354
- passwd 339
- paste 390
- pg 192, 386, 392
- piobe 397
- piping 353
- pr 406
- printenv 346
- ps 9, 192, 363, 466
- psh 436, 443
- pwd 140
- qcan 402
- qchk 405
- qhld 404
- qmov 403
- qpri 403
- qprt 397, 399, 408
- r 358, 359
- renice 192, 365
- repeating entered 358
- restore 158, 159, 162
- rm 382, 393
- rmdir 143
- rsh 436
- Rsh 436, 484
- saving entered 357
- script 376
- setclock 184
- sh 436
- shutdown 335
- smit 159, 162, 356, 399, 402, 403, 404, 405, 409, 431
- smit rmat 368
- sort 387
- stty 344, 432
- su 334, 420
- substituting strings 359
- syntax 353
- tail 389
- tapechk 146, 158
- tar 147, 159, 169
- tcopy 158
- tee 375
- text formatting 360
- tn 9
- touch 335
- tsh 436
- tty 343
- uname 336, 337
- uncompress 159, 160, 161
- unpack 159, 161
- usage statements 355
- wc 388
- whatis 357
- whereis 356
- who 192, 336, 337
- who am i 336
- whoami 336
- wsm 356

commands (*continued*)

- xinit 428
- xlock 423
- zcat 161
- commands and fastpaths 30
- commands list
  - for Bourne shell 499
  - for C shell 519
  - for Korn or POSIX shell 470
- comments
  - definition 437
- Common Desktop Environment 347
  - adding displays and terminals 348
  - customizing display devices 350
  - disabling autostart 347
  - enabling autostart 347
  - modifying profiles 348
  - removing displays and terminals 348
  - starting 347
  - starting manually 347
  - stopping 347
  - stopping manually 347
- comparing files 388
- compound commands 442
  - Bourne shell 487
- compress command 159, 160
- compressing
  - files 159
- concatenating
  - text files 372
- conditional substitution
  - Bourne shell 496
- connect-time accounting 257
- console
  - displaying name 343
- continue built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 461
- control keys
  - changing 432
  - displaying settings 344
- control mode 476
- converting
  - ASCII files to PostScript 408
  - DOS files 394
- coprocess facility
  - Korn or POSIX shell 459
- copying
  - base operating system files 394
  - DOS files 394
  - files 383
  - files from tape or disk 157
  - files to tape or disk 156
  - screen to file 376
  - to or from diskettes 156
  - to or from tape 158
- counting
  - bytes 388
  - lines 388
  - words 388

- cp command 141, 383
- cpio -i command 157
- cpio -o command 156
- cpio command 147
- CPU usage
  - displaying 248, 249
- creating
  - aliases 447
  - command alias 360
  - directories 139
  - shell scripts 439
- cron
  - for BSD System Managers 528
- cron daemon
  - generating data with 256
- csch command 436, 499
- Ctrl-C sequence 9
- Customized Configuration Database 278
- customizing
  - colors and fonts 428
  - display devices 350
  - key bindings 429
  - menu definitions 429
  - mouse button bindings 429
  - system environment 431, 432
- cut command 389
- cutting
  - sections of text files 389

**D**

- daemon processes 362
- data compression 131
  - fragments 125
  - performance costs of 133
- date command 184
- default built-in command
  - C shell 501
- default shell 436
- defaults
  - changing 428
- del command 393
- deleting
  - directories 143
  - DOS files 395
  - files 382
- desktop autostart
  - disabling 347
  - enabling 347
- device
  - configuring a read/write optical drive 281
  - for BSD System Managers 528
  - installation 273
- device configuration database
  - synchronizing with Logical Volume Manager 71
- device drivers
  - effect of using fragments on size of 134
- devices 283
  - changing attributes 317
  - checking attributes 317
  - checking the connections 319

- devices (*continued*)
  - checking the ready state 319
  - checking the software 317
  - checking the state of 317
  - classes 285
  - configuring large numbers 283
  - defining new 317
  - displaying information about 341
  - location codes 289
  - MPIO
    - cabling 304
  - MPIO-capable 304
  - nodes 285
  - printers or plotters 397
  - queue 397
  - running diagnostics 320
  - states 286
- df command 102
- diag command 183
- diagnosing boot problems
  - accessing a system that will not boot 7
  - rebooting a system with planar graphics 11
- diagnosing disk drive problems 63
- diagnostic output 371
- dials/LPFKeys location codes 292
- diff command 388
- dircmp command 144
- directories 137
  - abbreviations 138
  - access modes 413
  - changing 140
  - changing ownership 413
  - changing permissions 416
  - comparing contents 144
  - copying 141
  - creating 139
  - deleting 143
  - displaying 140
  - displaying contents 141
  - home 138
  - linking 392
  - listing DOS files 395
  - listing files 141
  - mounting 118
  - moving 140
  - naming conventions 138
  - organization 138
  - overview 137
  - parent 138
  - path names 139
  - permissions 413
  - removing 143
  - renaming 140
  - root 137
  - specifying with abbreviations 138
  - structure 138
  - subdirectories 138
  - types 137
  - working 138
- directory tree 138

- dirs built-in command
  - C shell 501
- discarding output 373
- disk
  - adding 32
  - removing 43
- disk drives
  - also see physical volumes 75
  - diagnosing 63
  - freeing space on 63
  - mounting space from another disk 64
  - recovering from problems 63
  - recovery of data
    - without reformatting 64
  - removing obsolete files from 63
  - restricting access to directories on 64
  - unmounting file systems on a disk 104
- disk drives (hard drives)
  - direct-attached 291
  - failure of
    - example of recovery from 68
  - listing file systems 104
  - serial-linked
    - location codes 291
- disk overflows, fixing 108
- disk striping 56
- disk utilization
  - effect of fragments on 125
- disk-usage accounting 258
- diskette drive
  - location codes 292
- diskettes
  - copying to or from 156
  - formatting 155
  - handling 147
  - using as backup medium 147
- diskless workstations
  - mount security 119
- disks (hard drives) 60
  - configuring 60
- displaying
  - access control information 421
  - available displays 343
  - available space 102
  - console name 343
  - control key assignments 344
  - DOS directory contents 395
  - environment variables 345
  - file contents 385
  - file directory 140
  - file directory contents 141
  - file types 385
  - first lines of files 389
  - fonts available 343
  - group information 415
  - keyboard maps 344
  - last lines of files 389
  - logged in users 337
  - login name 336
  - one screen at a time 386
  - operating system name 336
- displaying (*continued*)
  - software products 344
  - system devices 341
  - terminal name 343
  - text in large letters on screen 377
  - user ID 337
  - values of environment variables 346
  - your system name 337
- displays
  - listing currently available on system 343
- DOS files
  - converting 394
  - copying 394
  - deleting 395
  - listing contents 395
- dosdel command 395
- dosdir command 395
- dosread command 394
- doswrite command 394
- DVD
  - file systems 105
- Dynamic Processor Deallocation 174, 176, 185

## E

- echo built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 466
- echo command 376
- ed editor 382
- editing
  - access control information 423
  - command history 359
  - inline in Korn or POSIX shell 473
- EDITOR variable 452
- editors
  - ed 382
  - emacs 473, 474
  - gmacs 473, 474
  - inline editing 473
  - vi 382, 473
- else built-in command
  - C shell 501
- emacs editor
  - inline editing 473, 474
- emergency
  - shutting down in an 20
- enabled file systems
  - create 131
  - free space 131
  - large file geometry 131
- enabling file systems
  - zero file allocations 131
- end built-in command
  - C shell 501
- endif built-in command
  - C shell 501
- endsw built-in command
  - C shell 501
- Enhanced Journaled File System (JFS2) 123

- enhanced Korn shell
  - arithmetic enhancements 479
  - associative arrays 479
  - built-in commands 479
  - command return values 479
  - compound assignments 479
  - compound variables 479
  - description 479
  - discipline functions 479
  - function environments 479
  - parameter expansions 479
  - PATH search rules 479
  - shell history 479
  - variable name references 479
  - variables 479
- enq command 397
- encrypt filter 408
- env command 345
- ENV variable 452
- environment
  - displaying current 345
  - file 425
  - setting 425
  - system 341
- environment variables
  - C shell 515
  - displaying values 346
- ERRNO variable 451
- error logging
  - checking for device errors 317
- error output 371
- eval built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 461
- exclusive use RSET
  - exclusive use processor resource set 236
- exec built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 443, 461
- exit built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 461
- exit command 335
- exit status
  - Korn or POSIX shell 460
- expanding
  - files 160, 161
- export built-in command 460
  - Bourne shell 488
  - Korn or POSIX shell 443, 461
- export command 431
- exporting
  - aliases 447
  - shell variables 431
- expressions
  - C shell 506
  - conditional 471
  - finding files with matching 384

## F

- failed disk drive
  - example of recovery from 68
- fc built-in command 359, 445
  - Korn or POSIX shell 466
- FCEDIT variable 452
- fdformat command 155
- fee accounting 259
- fg built-in command
  - C shell 501
  - Korn or POSIX shell 466
- field splitting
  - Korn or POSIX shell 455
- file
  - command 385
  - descriptor 371, 373
  - trees 93
- file name substitution
  - Bourne shell 497
  - C shell 513
  - Korn or POSIX shell 456
- file system
  - bypassing 40
  - images 134
- file system fragment addressability 129
- file system log 34
- file systems 137
  - /opt 94
  - /proc 94
  - backing up user file systems 149
  - backing up with scripts 153
  - CD-ROM File System (CDRFS) 123
  - CDRFS 105
  - checking for consistency 155
  - commands for managing 102, 103
  - conducting interactive repairs 155
  - data compression 131
  - description 333
  - disk overflows 108
  - Enhanced Journaled File System (JFS2) 123
  - example 380
  - file tree
    - / (root) file system 96
    - /export directory 101
    - /usr file system 98
    - /usr/share directory 99
    - /var file system 100
    - overview 94
    - root (/) file system 96
  - fixing damaged 112
  - for BSD System Managers 530
  - fragments 125
  - groups
    - mounting 104
    - unmounting 104
  - home 94
  - i-nodes 125
  - Journaled File System (JFS) 93, 123
  - journaling techniques 93
  - large files 131
  - management tasks 102

- file systems *(continued)*
  - mounting 104, 118
  - Network File System (NFS) 123
  - on read/write optical media 105
  - overview 93
  - reducing size in root volume group 114
  - root 94
  - space available 102
  - sparse files 130
  - structure 94
  - types
    - CD-ROM 123
    - DVD-ROM 123
    - enhanced journaled file system (JFS2) 123
    - journaled file system (JFS) 123
    - network file system (NFS) 123
  - UDFS 105
  - unmounting 104
  - verifying integrity of 107
- file types
  - binary 379
  - directory 379
  - text 379
- files 137
  - /dev/null 373
  - /etc/environment 425
  - /etc/passwd 436
  - /etc/profile 426
  - /etc/security/passwd 412
  - .env file 427
  - .hushlogin 335
  - .mwmrc 429
  - .profile 426
  - .Xdefaults 428
  - .xinitrc 428
  - access modes 413
  - appending single line of text 376
  - archiving 169
  - ASCII 379
  - backing up 158
  - binary 379
  - changing ownership 413
  - changing permissions 416
  - comparing 144, 388
  - compressing 159
  - concatenating 372
  - copying 383
  - copying from DOS 394
  - copying from screen 376
  - copying from tape or disk 157
  - copying to DOS 394
  - creating with redirection from keyboard 372
  - cutting selected fields from 389
  - deleting 382
  - deleting DOS 395
  - displaying contents 385
  - displaying first lines 389
  - displaying last lines 389
  - displaying types 385
  - environment 425
  - executable 379

- files *(continued)*
  - expanding 161
  - for BSD System Managers 529, 531
  - formatting for display 385
  - formatting for printing 406
  - handling 382
  - HISTFILE 445
  - identifying type 385
  - joining 372
  - linking 392
  - locating sections 356
  - matching expressions 384
  - merging the lines of several 390
  - metacharacters 381
  - mounting 118
  - moving 383
  - naming conventions 380
  - numbering lines 391
  - overview 379
  - ownership 412
  - packing 159
  - pasting text 390
  - path names 139, 380
  - permissions 379, 413
  - printing ASCII on a PostScript printer 408
  - regular expressions 381
  - removing 382
  - removing columns 391
  - removing linked 393
  - renaming 383
  - restoring 159, 162, 166
  - retrieving from storage 169
  - searching for a string 386
  - sorting text 387
  - uncompressing 161
  - unpacking 161
  - writing to output 389
- filters 374
- find command 158, 384
- finding
  - files 384
  - text strings within files 386
- fixed-disk drives (hard drives) 108
  - also see disk drives 63
- flags 353
  - for pr command 406
  - for qprt command 399, 408
  - in commands 354
- flat network 18
- flcopy command 156
- fonts
  - changing 431
  - listing available for use 343
- foreach built-in command
  - C shell 501
- foreground processes 362
- format command 155
- formatting
  - diskettes 155
  - files for printing 406
- FPATH variable 452



fragments  
  and variable number of i-nodes 125  
  effect on backup/restore 134  
  effect on disk utilization 125  
  limitation for device drivers 134  
  performance costs of 129  
  size of  
    identifying 127  
    specifying 127  
fsck command 146, 155

## G

getopts built-in command  
  Korn or POSIX shell 466  
glob built-in command  
  C shell 501  
gmacs editor  
  inline editing 473, 474  
goto built-in command  
  C shell 501  
grep command 9, 374, 386  
groups command 412

## H

hangups built-in command  
  C shell 501  
hard disk 60  
hash built-in command  
  Bourne shell 488  
hashstat built-in command  
  C shell 501  
head command 389  
here document 374, 457  
hierarchical network 18  
HISTFILE  
  file 445  
  variable 452  
history  
  editing 359  
  lists in C shell 509  
  substitution in C shell 509  
history built-in command  
  C shell 501  
history command 357  
HISTSIZE variable 445, 452  
holding  
  print jobs 404  
home directory 138  
home file system 94  
HOME variable 452  
hot disk removability 32, 43  
hot plug management  
  PCI 297  
hot removability 43, 44, 69  
hot spots in logical volumes 58  
hot spots in logical volumes, enabling 36

**I**  
i-node number 137, 379, 392  
i-nodes 127  
  and fragments 125  
  number of bytes per (NBPI)  
    identifying 127  
    specifying 127  
  variable number of 127  
i-nodes, number of 129  
I/O redirection  
  Bourne shell 498  
  C shell 517  
  Korn or POSIX shell 457  
  standard 371  
id command 334, 337, 412  
idbgen 174  
IDE devices  
  address for a tape drive 276  
  controls for a tape drive 276  
  customized attributes 279  
  installing 275  
    Customized Configuration Database 278  
identifier  
  definition 437  
IDs  
  user 412  
if built-in command  
  C shell 501  
IFS variable 452  
importing user-defined volume groups 114  
inactive system  
  checking hardware 8  
  checking processes 9  
  restarting the system 10  
index node reference number 137  
inittab file 15  
  srcmstr daemon in 240  
inline editing  
  emacs editing mode 474  
  gmacs editing mode 474  
  Korn or POSIX shell 473  
  vi editing mode 476, 477, 478, 479  
inline input documents 374  
inoperable system  
  checking hardware 8  
  checking processes 9  
  restarting the system 10  
input  
  redirection 371  
  redirection operator 373  
input mode  
  definition 476  
  input edit commands 476  
integer arithmetic 454  
inter-disk allocation strategy 53  
international character support  
  text formatting 360  
interpreting  
  blanks 495  
intra-disk allocation strategy 55

## J

- JFS
  - copy to another physical volume 135
- JFS (journaled file system)
  - data compression 131
  - fragments 125
  - maximum size of 129
  - on read / write optical media 105
  - size limitations 128
  - with variable number of i-nodes 125
- JFS (journaled file system) log
  - size of 129
- JFS log 34
- JFS2 (enhanced journaled file system)
  - size limitations 128, 130
- JFS2 log 34
- job control
  - C shell 518
  - Korn or POSIX shell 472
- jobs
  - listing scheduled 367
  - removing from schedule 368
  - scheduling 367
- jobs built-in command
  - C shell 501, 518
  - Korn or POSIX shell 466
- Journaled File System (JFS) 93, 123

## K

- key bindings 429
- keyboard
  - changing attributes
    - using chhwkbd command 526
- keyboard maps
  - listing currently available 344
- kill built-in command
  - C shell 501
  - Korn or POSIX shell 466
- kill command 9, 192, 368
- Korn shell or POSIX shell 436
  - arithmetic evaluation 454
  - built-in commands 460
  - command aliasing 447
  - command history 445
  - command substitution 453
  - compound commands 442
  - conditional expressions 471
  - coprocess facility 459
  - editing 473
  - enhanced 479
  - environment 443
  - exit status 460
  - field splitting 455
  - file name substitution 456
  - functions 444
  - job control 472
  - list of regular built-in commands 470
  - list of special built-in commands 470
  - parameter substitution 449

- Korn shell or POSIX shell (*continued*)
  - pattern matching 456
  - predefined parameters 450
  - predefined variables 451
  - quote removal 457
  - quoting 445
  - redirecting input and output 457
  - redirecting input and output from coprocesses 459
  - reserved words 447
  - signal handling 473
  - starting 443
  - tilde substitution 448
  - user-defined variables 452
  - using commands 440
- ksh command 157, 436, 443
- ksh93
  - arithmetic enhancements 479
  - associative arrays 479
  - built-in commands 479
  - command return values 479
  - compound assignments 479
  - compound variables 479
  - description 479
  - discipline functions 479
  - function environments 479
  - parameter expansions 479
  - PATH search rules 479
  - shell history 479
  - variable name references 479
  - variables 479

## L

- LANG variable 452
- languages
  - bidirectional 346
- LC\_ALL variable 452
- LC\_COLLATE variable 452
- LC\_CTYPE variable 452
- LC\_MESSAGES variable 452
- let built-in command
  - Korn or POSIX shell 454, 466
- limit built-in command
  - C shell 501
- limitations
  - logical volumes 70
- line of text
  - appending to file 376
- LINENO variable 451
- lines
  - counting number of 388
- LINES variable 452
- linked files
  - removing 393
- linking
  - directories 392
  - files 392
- links
  - creating 392
  - hard 392
  - overview 392

- links *(continued)*
  - removing 393
  - symbolic 392
  - types 392
- listing
  - aliases 447
  - scheduled processes 367
- lists
  - definition 437
- In command 137, 392
- local printers
  - definition 397
- location codes 289
  - adapter 289
  - defined 289
  - dials/LPFKeys 292
  - direct-attached disk 291
  - diskette drive 292
  - multiprotocol port 292
  - printer/plotter 290
  - SCSI device 291
  - serial-linked disk 291
  - tty 290
- lock command 423
- locking
  - your terminal 423
- logging in
  - as another user 334
  - more than one time 334
  - to the operating system 333
- logging out
  - of the operating system 335
- logical partitions
  - defining size of 114
  - definition 28
  - inter-disk allocation strategy 53
- logical volume
  - copy to another physical volume 33
  - raw
    - define 40
- Logical Volume Manager 30
- Logical Volume Manager (LVM) 25
  - definition 60
  - synchronizing with device configuration
    - database 71
- logical volume storage
  - definition 25
  - disk overflows 108
  - file systems 29
  - inter-disk allocation policy 53
  - intra-disk allocation policy 55
  - logical partitions 28
  - logical volumes 28
  - maximum sizes 29
  - nonquorum volume groups 74
  - physical volumes 26
  - quorums 73
  - volume groups 26
  - write scheduling policy 50, 51
- logical volumes
  - adding a file system on new 103
- logical volumes *(continued)*
  - changing name 32
  - definition 28
  - hot spots 36, 58
  - limitations 70
  - map files 56
  - moving contents to another system 38
  - replacing a disk 69
  - size
    - checking 103
    - decreasing 103
    - increasing 103
  - strategy for 49
  - striped 56
  - volume group policy 59
  - write-verify policy 57
- logical-volume control block
  - not protected from raw-logical-volume access 40
- login
  - directory 333
  - displaying name 336
  - name 333
  - overview 333
  - shell 436
  - suppressing messages 335
  - user ID 412
- login built-in command
  - C shell 501
- login command 334, 420
- login files
  - /etc/environment 425
  - /etc/profile 426
  - /etc/profile file 173
  - .env file 427
  - .profile 426
  - .profile file 173
- logname command 336
- logout
  - overview 333
- logout built-in command
  - C shell 501
- logout command 335
- lp command 397
- lpr command 397
- ls command 141, 412, 413
- lsattr command 317
- lscfg command 341
- lscons command 343
- lsdev command 317
- lsdisp command 343
- lsfont command 343
- lsgroup command 415
- lskbd command 344
- lslpp command 344
- lssrc command 242
- LVCB (logical-volume control block)
  - not protected from raw-logical-volume access 40
- LVM 25, 30

## M

- MAIL variable 452
- MAILCHECK variable 452
- MAILPATH variable 452
- maintaining
  - access control lists 416
  - ACLs 416
- maintenance 30
- man command 4, 356
  - BSD System Managers 533
- map files 56
- maps
  - keyboard 344
- menu definitions 429
- message of the day
  - changing 185
- messages
  - displaying on screen 376
  - sending to standard output 376
- messages, screen, responding to 192
- metacharacters 381
  - definition 437
  - quoting in Korn or POSIX shell 445
- Mirror Write Consistency (MWC) 51
- mirrored volume group
  - replacing a physical volume 75
- mirroring
  - root volume group (rootvg) 41
  - splitting a mirrored disk from a volume group 80
  - volume group 41
- mkdev command 317
- mkdir command 139
- mkvirprt command 397
- modifying
  - desktop profiles 348
- monitoring processes 187
- more command 386
- motd file 185
- mount points 117
- mounting
  - /etc/filesystem automatic mounts 118
  - automatic mounts 118
  - diskless workstation mounts
    - description of 121
    - security 119
  - file system mounting 118
  - local
    - definition 118
  - overview 117
  - remote
    - definition 118
  - using multiple mounts 118
- mouse button bindings 429
- moving
  - print jobs 403
- MPIO 303
  - managing 304
- Multi-path I/O 303
- multibyte character support
  - enter characters 361
  - text formatting 361

- multiprotocol port
  - location codes 292
- multiuser systems
  - changing run levels on 14
- mv command 383
- mmdir command 140
- mwm command 427

## N

- named parameters 449
- naming conventions
  - directories 138
  - files 380
- NBPI 127
- network
  - displaying system name 337
  - for BSD System Managers 521, 532, 533
- Network File System (NFS) 123
- network planning
  - TCP/IP 18
- newgrp built-in command
  - Korn or POSIX shell 461
- NFS and NIS
  - BSD System Managers 534
- nice built-in command
  - C shell 501
- nice command 365
- NIS 534
- nl command 391
- NLSPATH variable 452
- nonquorum volume groups 74
- notify built-in command
  - C shell 501
- number of bytes per i-node (NBPI) 127
- numbering
  - lines in text files 391

## O

- OLDPWD variable 451
- onintr built-in command
  - C shell 501
- operands
  - in commands 354
- operating system
  - displaying name 336
  - loading 10
  - logging in 333
  - logging out 335
  - shutting down 335
- operators
  - C shell 506
- OPTARG variable 451
- optical drive
  - configuring 281
- optical media
  - using file systems on read/write 105
- OPTIND variable 451
- options
  - in commands 354

- output
  - discarding with /dev/null file 373
  - redirecting to a file 372
  - redirection 371
  - redirection operator 371
- overriding
  - auto-determination of print file types 409

## P

- pack command 159, 161
- page command 386
- paging space
  - AIX for BSD System Managers 532
  - allocating 88
  - changing characteristics of 84
  - changing size of hd6 84
  - characteristics for creating 91
  - commands for managing 91
  - early allocation mode 88
  - late allocation mode 88
  - moving hd6 84
  - overview 83, 88
  - removing 84
- parameter assignment lists
  - definition 437
- parameter substitution
  - Korn or POSIX shell 449
- parameters
  - in commands 354
  - Korn or POSIX shell 449, 450
  - named 449
  - positional 449
  - predefined 450
  - special 449, 450
- parent directory 138
- passwd command 339
- passwords
  - changing or setting 339
  - description 333
  - guidelines 338
  - setting to null 339
- paste command 390
- pasting
  - sections of text files 390
- path names
  - absolute 139, 380
  - directory 139
  - files 380
  - relative 139
- PATH variable 452
- paths
  - directory 139
- pattern matching
  - Bourne shell 497
  - Korn or POSIX shell 456
- performance
  - BSD System Managers 537
  - improving
    - defining raw logical volumes 40
- permissions
  - directory 416
  - file 416
- pg command 192, 386, 392
- physical partitions
  - definition 27
  - size 27
- physical volume
  - copy JFS to another 135
  - copy logical volume to another 33
- physical volumes
  - configuring a disk 60
  - creating from available disk drive 62
  - definition 26
  - moving contents 38
  - replacing in a mirrored volume group 75
- PID number 362
- piobe command 397
- pipelines
  - definition 374, 437
- pipes 374
- piping 353
- popd built-in command
  - C shell 501
- positional parameters 449
  - Bourne shell 497
- PostScript files
  - converting from ASCII 408, 409
- PostScript printers
  - printing ASCII files 408
- PPID variable 451
- pr command
  - flags 406
- predefined variables
  - Bourne shell 495
  - Korn or POSIX shell 451
- print built-in command
  - Korn or POSIX shell 466
- print jobs
  - canceling 402
  - checking status 405
  - definition 397
  - holding 404
  - moving 403
  - prioritizing 403
  - releasing 404
  - starting 399
- print queue
  - status conditions 406
- print spoolers
  - definition 397
- printenv command 346
- printer
  - for BSD System Managers 537
  - location codes 290
- printer backend
  - definition 397
- printer-usage accounting 259
- printer/plotter device
  - definition 397

- printers
  - local 397
  - overview 397
  - real 397
  - remote 397
  - status conditions 406
  - virtual 397
- printing
  - ASCII files on a PostScript printer 408
  - canceling print jobs 402
  - checking status of print jobs 405
  - devices 397
  - formatting files for 406
  - holding print jobs 404
  - local printers 397
  - moving print jobs 403
  - overriding print file types 409
  - overview 397
  - print jobs 397
  - printer backend 397
  - prioritizing print jobs 403
  - qdaemon 397
  - queue 397
  - queue devices 397
  - real printers 397
  - releasing print jobs 404
  - remote printers 397
  - spoolers 397
  - starting print jobs 399
  - status conditions of printers 406
  - terminology 397
  - virtual printers 397
- prioritizing
  - print jobs 403
- priority of processes 190
- process identification number 362
- process summaries 369
- processes 353
  - background 362
  - batch 367
  - binding of to a processor 191
  - canceling foreground processes 365
  - changing priority 365
  - checking status 363
  - collecting accounting data on 257
  - daemon 362
  - description 362
  - displaying all active 363
  - displaying CPU usage 248, 249
  - foreground 362
  - generating accounting reports 258
  - listing scheduled 367
  - management of 187
  - monitoring of 187
  - priority alteration of 190
  - qdaemon 397
  - removing background processes 368
  - removing from schedule 368
  - restarting stopped 366
  - scheduling for later operation 367
  - setting initial priority 365

- processes (*continued*)
  - srcmstr 397
  - starting 363
  - startsrc 397
  - stopping foreground processes 366
  - termination of 190
  - zombie 362
- profile
  - files 173
  - overview 173
- profile files 425
- program
  - copying output into a file 375
- prompt
  - changing 432
- ps command 9, 192, 363, 466
- PS1 variable 452
- PS2 variable 452
- PS3 variable 452
- PS4 variable 452
- psh command 436, 443
- pushd built-in command
  - C shell 501
- pwd built-in command
  - Bourne shell 488
  - Korn or POSIX shell 466
- pwd command 140
- PWD variable 451

## Q

- qcan command 402
- qchk command 405
- qdaemon
  - definition 397
  - printer backend 397
- qhld command 404
- qmov command 403
- qpri command 403
- qprt command 397, 399
  - flags 399, 408
- queue
  - status conditions 406
- queue devices
  - definition 397
- queues
  - definition 397
- quorums
  - changing to nonquorum status 74
  - definition 73
  - nonquorum volume groups 74
- quote removal
  - Korn or POSIX shell 457
- quoting characters
  - Bourne shell 486
  - Korn or POSIX shell 445

## R

- r alias 358, 359
- r command 358, 359

- RANDOM variable 451
- Range setting 53
- raw logical volume
  - define 40
- read built-in command
  - Bourne shell 488, 492
  - Korn or POSIX shell 466
- reading the three-digit display 333
- readonly built-in command 460
  - Bourne shell 488
  - Korn or POSIX shell 461
- real printers
  - definition 397
- rebooting a system with planar graphics 11
- recovering data from a disk without reformatting 64
- recovery procedures
  - accessing a system that will not boot 7
  - rebooting a system with planar graphics 11
- recovery procedures for failed disk drive
  - example of 68
- redirecting
  - input and output from coprocesses 459
  - input and output in Bourne shell 498
  - input and output in Korn or POSIX shell 457
  - output to a file 372
  - standard error output 373
  - standard input 373
  - standard output 371
- refresh command 242
- regular built-in commands
  - Korn or POSIX shell 466, 470
- regular expressions 381
- rehash built-in command
  - C shell 501
- relative path name 139
- releasing
  - print jobs 404
- remote
  - login 333
  - shell 436
- remote printers
  - definition 397
- removing
  - aliases 447
  - background processes 368
  - columns in text files 391
  - linked files 393
  - local display 348
  - processes from schedule 368
- renaming
  - directories 140
  - files 383
- renice command 192, 365
- repeat built-in command
  - C shell 501
- REPLY variable 451
- reserved words
  - Bourne shell 487
  - Korn or POSIX shell 447
- resource files
  - modifying 428, 429
- restart the system 10
- restarting
  - stopped processes 366
- restore
  - effect of fragments on 134
- restore command 158, 159, 162
- restoring
  - files 159, 162
- Restricted Korn Shell
  - starting 484
- restricted shell
  - starting 484
- Restricted shell 436
- restricting users from specified directories 64
- return built-in command
  - Bourne shell 488
  - Korn or POSIX shell 461
- rm command 382, 393
- rmdir command 143
- root (/) file system 96
- root file system 94
- root volume group (rootvg)
  - mirroring 41
  - reducing size of file systems 114
- rsh command 436
- Rsh command 436, 484
- run level
  - changing 14
  - displaying history 13
  - identifying 13
- runacct command
  - restarting 266
  - starting 266
- running
  - shell scripts 439

## S

- scheduling
  - processes 367
- screen messages, responding to 192
- screens
  - clearing 375
  - copying display to a file 375
  - copying to file 376
  - displaying text in large letters 377
  - displaying text one screen at a time 386
- script command 376
- SCSI devices
  - location codes 291
- SECONDS variable 451
- security
  - /etc/security/passwd file 412
  - authentication 412
  - backups 411
  - file 411
  - identification 412
  - login user ID 412
  - system 411
  - threats 411
  - unattended terminals 412

- set built-in command 497
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 461
- setclock command 184
- setenv built-in command
  - C shell 501
- setgroups built-in command
  - Korn or POSIX shell 466
- setsenv built-in command
  - Korn or POSIX shell 466
- setting
  - access control information 422
  - initial priority of processes 365
- sh command 436
- Shared Product Object Tree (SPOT) directory 101
- shell commands
  - fc 359
  - history 357
  - r alias 358, 359
- shell environments
  - customizing 173
- shell procedures 353
- shell scripts 353
  - creating 439
  - specifying a shell 440
- SHELL variable 452
- shell variables
  - definition 437
  - exporting 431
  - local 431
- shells
  - alias substitution in C shell 511
  - Bourne 436
  - Bourne built-in commands 488
  - Bourne command substitution 492
  - Bourne environment 483
  - Bourne I/O redirection 498
  - Bourne list of built-in commands 499
  - Bourne predefined variables 495
  - Bourne user-defined variables 492
  - Bourne variable substitution 492
  - C 436
  - C built-in commands 500, 501
  - character classes in Bourne 497
  - command execution in C shell 508
  - command substitution in C shell 508
  - conditional substitution in Bourne 496
  - creating shell scripts 439
  - default 436
  - environment variables in C shell 515
  - features 435
  - file name substitution in Bourne 497
  - file name substitution in C shell 513
  - history lists in C shell 509
  - history substitution in C shell 509
  - job control in C shell 518
  - Korn 436
  - Korn or POSIX arithmetic evaluation 454
  - Korn or POSIX built-in commands 460
  - Korn or POSIX command aliasing 447
  - Korn or POSIX command history 445
  - Korn or POSIX command substitution 453
  - Korn or POSIX compound commands 442
  - Korn or POSIX conditional expressions 471
  - Korn or POSIX coprocess facility 459
  - Korn or POSIX environment 443
  - Korn or POSIX exit status 460
  - Korn or POSIX file name substitution 456
  - Korn or POSIX I/O redirection 457
  - Korn or POSIX inline editing 473
  - Korn or POSIX job control 472
  - Korn or POSIX list of regular built-in commands 470
  - Korn or POSIX list of special built-in commands 470
  - Korn or POSIX reserved words 447
  - Korn or POSIX signal handling 473
  - login 436
  - overview 435
  - parameters 449
  - positional parameters in Bourne 497
  - POSIX 436
  - quoting in Korn or POSIX 445
  - redirecting input and output in C shell 517
  - remote 436
  - Restricted 436
  - signal handling in C shell 500
  - standard 436
  - starting Bourne shell 483
  - starting C shell 499
  - starting Korn or POSIX 443
  - starting restricted 484
  - terminology 437
  - trusted 436
  - types 436
  - using Korn or POSIX commands 440
  - variable substitution in C shell 512
  - variables used by Bourne 493
- shift built-in command 497
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 461
- shortcut name for commands
  - creating 360
- shutdown
  - emergency 20
  - to single-user mode 20
  - understanding 19
  - without rebooting 20
- shutdown command 335
- shutting down the operating system 335
- shutting down the system 20
- SIGINT signal 473
- signal handling
  - Bourne shell 487
  - C shell 500
  - Korn or POSIX shell 473
- signals
  - SIGINT 473
  - SIGQUIT 473



- SIGQUIT signal 473
- simple commands
  - definition 437
- single-user mode 20
- single-user systems
  - changing run levels on 14
- skulker command 63
- smit command 159, 356, 431
  - canceling a print job 402
  - checking status of a print job 405
  - converting ASCII to PostScript 409
  - holding a print job 404
  - moving a print job 403
  - prioritizing a print job 403
  - releasing a print job 404
  - restoring files 162
  - starting a print job 399
- smit rmat command 368
- software
  - checking for device problems 317
- software products
  - displaying information about 344
- sort command 387
- sorting
  - text files 387
- source built-in command
  - C shell 501
- space
  - displaying available 102
- special built-in commands
  - Bourne shell 488
  - Korn or POSIX shell 461, 470
- special parameters 449
- splitting a mirrored disk from a volume group 80
- spoolers
  - definition 397
- SPOT directory 101
- srcmstr command 242
- srcmstr daemon 240
- standard error 371
- standard error output
  - redirecting 373
- standard input 371
  - copying to a file 375
  - redirecting 373
- standard output 371
  - appending to a file 372
  - redirecting 371
- standard shell
  - conditional expressions 471
- starting
  - AIXwindows Window Manager 427
  - Bourne shell 483
  - C shell 499
  - Korn or POSIX shell 443
  - print jobs 399
  - processes 363
  - Restricted Korn Shell 484
  - restricted shell 484
  - Web-based System Manager 356
- starting Workload Manager 201
- startsrc command 241
- startup
  - controlling windows and applications 428
- startup files
  - AIXwindows 427
  - C shell 499
  - system 425
  - X Server 427
- status conditions
  - of printers 406
- stderr 371
- stdin 371
- stdout 371
- stop built-in command
  - C shell 501
- stopping
  - foreground processes 366
- stopping Workload Manager 201
- stopsrc command 241
- storage media 145
- strict inter-disk setting 54
- strings
  - finding in text files 386
- stty command 344, 432
- su command 334, 420
- subserver
  - description of 239
  - displaying status 242
  - starting 241
  - stopping 241
  - turning off tracing 242
  - turning on tracing 242
- subshells
  - definition 437
- subsystem
  - displaying status 242
  - properties of 239
  - refreshing 242
  - starting 241
  - stopping 241
  - turning off tracing 242
  - turning on tracing 242
- subsystem group
  - description of 239
  - displaying status 242
  - refreshing 242
  - starting 241
  - stopping 241
  - turning off tracing 242
  - turning on tracing 242
- summaries
  - AIXwindows startup files 341
  - commands 377
  - customizing system environment 341
  - for commands 369
  - for printing 409
  - for processes 369
  - system startup files 341
- super strict inter-disk setting 54
- suspend built-in command
  - C shell 501

- swap space
  - see paging space 83
- switch built-in command
  - C shell 501
- switches
  - in commands 354
- system
  - accounting 333
  - changing prompt 432
  - customizing environment 431, 432
  - default variables 426
  - displaying name 337
  - environment 341
  - management 93
  - powering on 333
  - security 411
  - starting the 5
  - startup files 425
  - stopping the 20
- system accounting
  - commands
    - running automatically 263
    - running from the keyboard 264
  - connect-time data 249, 257
  - CPU usage
    - displaying 248, 249
  - disk-usage data 250, 259
    - collecting 258
  - failure
    - recovering from 266
  - fees
    - charging 259
    - reporting 259
  - files
    - data files 265
    - formats 268
    - overview 265
    - report and summary files 265
    - runnact command files 267
  - holidays file
    - updating 256
  - overview 245
  - printer-usage data 250
    - collecting 259
    - reporting 259
  - problems
    - fixing bad times 252
    - fixing incorrect file permissions 252
    - fixing runacct errors 253
    - fixing-out-of-date holidays file 256
  - process data
    - collecting 257
    - reporting 258
  - reporting data
    - overview 260
  - reports
    - daily 260, 261
    - fiscal 259
    - monthly 262, 263
  - runnact command
    - restarting 266
- system accounting (*continued*)
  - runnact command (*continued*)
    - starting 266
  - setting up 245
  - summarizing records 262
  - system activity
    - data 260
  - system activity data
    - displaying 247
    - displaying while running a command 247
  - tacct errors
    - fixing 251
  - wtmp errors
    - fixing 251
  - system activity
    - tracking 260
  - system battery 183
  - system clock
    - resetting 183
    - testing the battery 183
  - system environment 174, 185
    - 64-bit mode 181
    - Dynamic Processor Deallocation 174, 176, 185
    - message of the day 185
    - profile 173
    - time data manipulation services 174
  - system failure
    - checking hardware 8
    - checking processes 9
    - restarting the system 10
  - System Management Interface Tool (SMIT) 271
  - System Resource Controller
    - commands
      - list of 240
    - functions of 239
    - starting 240
  - system run level 13
    - changing 14

## T

- tacct errors
  - fixing 251
- tail command 389
- tape drives
  - attributes
    - changeable 321, 323, 324, 325, 326, 327, 328, 329, 330
    - managing 321
    - special files for 331
- tapechk command 146, 158
- tapes
  - checking integrity 158
  - copying to or from 158
  - using as backup medium 147
- tar command 147, 159, 169
- tcopy command 158
- TCP/IP
  - /etc/hosts 18
  - naming
    - flat network 18

- TCP/IP (*continued*)
  - naming (*continued*)
    - hierarchical network 18
    - network planning 18
- tee command 375
- terminal problems
  - stopping stalled processes 192
- terminal, locked up 192
- terminals
  - displaying control key assignments 344
  - displaying name 343
  - displaying settings 346
  - for BSD System Managers 539
  - locking 423
  - unattended 412
- terminology
  - for shells 437
- test built-in command
  - Bourne shell 488
  - Korn or POSIX shell 466
- text
  - appending to a file 376
  - displaying in large letters 377
- text files
  - concatenating 372
  - creating from keyboard input 372
  - cutting sections 389
  - finding strings 386
  - numbering lines 391
  - pasting sections 390
  - removing columns 391
  - sorting 387
- text formatting
  - commands 360
  - extended single-byte characters 361
  - international character support 360
  - multibyte character support 361
- Text Formatting System 408
- three-digit display 333
- tilde substitution
  - aliasing commands 448
- time built-in command
  - C shell 501
- times built-in command
  - Bourne shell 488
  - Korn or POSIX shell 461
- TMOU variable 452
- tn3270 command 521
- touch command 335
- tracesoff command 242
- traceson command 242
- tracked aliases 448
- Transmission Control Protocol/Internet Protocol 18
- trap built-in command
  - Bourne shell 488
  - Korn or POSIX shell 461
- trusted shell 436
- tsh command 436
- tty (teletypewriter)
  - location codes 290
- tty command 343

- type built-in command
  - Bourne shell 488
- typeset built-in command 460
  - Korn or POSIX shell 443, 449, 454, 461

## U

- ulimit built-in command
  - Bourne shell 488
  - Korn or POSIX shell 466
- umask built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 466
- unalias built-in command
  - C shell 501
  - Korn or POSIX shell 447, 466
- uname command 336, 337
- uncompress command 159, 160, 161
- uncompressing
  - files 161
- underscore variable 451
- unhash built-in command
  - C shell 501
- unlimit built-in command
  - C shell 501
- unmirroring
  - volume group 43
- unpack command 159, 161
- unpacking
  - files 161
- unset built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 461
- unsetenv built-in command 501
- usage statements
  - for commands 355
- user
  - changing to another 334
  - classes 413
  - displaying group information 415
  - groups 412
- user environments
  - customizing 173
- user ID
  - changing to another 334
  - login 412
- user-defined variables 452
  - Bourne shell 492
- user-defined volume groups
  - importing 114
- users
  - displaying system ID 337
  - displaying who is logged in 337
- UUCP
  - BSD System Managers 539

## V

- variable number of i-nodes 127
  - and fragments 125
- variable substitution
  - Bourne shell 492
  - C shell 512
  - Korn or POSIX shell 451
- variables
  - Bourne shell 493, 495
  - Bourne shell user-defined 492
  - C shell environment 515
  - CDPATH 452
  - COLUMNS 452
  - EDITOR 452
  - ENV 452
  - ERRNO 451
  - exporting 431
  - FCEDIT 452
  - FPATH 452
  - HISTFILE 452
  - HISTSIZE 445, 452
  - HOME 452
  - IFS 452
  - Korn or POSIX shell 451, 452
  - LANG 452
  - LC\_ALL 452
  - LC\_COLLATE 452
  - LC\_CTYPE 452
  - LC\_MESSAGES 452
  - LINENO 451
  - LINES 452
  - MAIL 452
  - MAILCHECK 452
  - MAILPATH 452
  - NLSPATH 452
  - OLDPWD 451
  - OPTARG 451
  - OPTIND 451
  - PATH 452
  - PPID 451
  - predefined 451
  - PS1 452
  - PS2 452
  - PS3 452
  - PS4 452
  - PWD 451
  - RANDOM 451
  - REPLY 451
  - SECONDS 451
  - SHELL 452
  - SHELL PROMPT variable 452
  - TMOUT 452
  - underscore 451
  - user-defined 452
  - variables
    - SHELL PROMPT 452
  - VISUAL 452
- vary-on process 73
  - overriding failure of 71
- verifying file systems 107
- VGDA (volume group descriptor area) 73

- VGSA (volume group status area) 73
- vi editor 382
  - commonly used edit commands 479
  - control mode 476
  - cursor movement 477
  - inline editing 473, 476, 477, 478, 479
  - input edit commands 476
  - input mode 476
  - miscellaneous edit commands 479
  - motion edit commands 477
  - search edit commands 477
  - text-modification edit commands 478
- Virtual Memory Manager 87
- Virtual Memory Manager (VMM)
  - overview 83
- virtual printers
  - definition 397
- VISUAL variable 452
- VMM 87
- volume group
  - mirroring 41
  - root
    - mirroring 41
  - splitting a mirrored disk from 80
  - unmirroring 43
- volume group descriptor area (VGDA) 73
- volume group status area (VGSA) 73
- volume groups
  - changing to nonquorum status 74
  - definition of 26
  - exporting 37
  - high availability 48
  - importing 37
  - mirrored
    - replacing a physical volume 75
  - moving 37
  - nonquorum 74
  - policy implementation 59
  - quorums 73
  - replacing a disk 69
  - strategy for 48
  - user-defined
    - importing 114
  - vary-on process 73
  - when to create separate 48

## W

- wait built-in command
  - Bourne shell 488
  - C shell 501
  - Korn or POSIX shell 466
- wc command 388
- Web-based System Manager 269
  - canceling a print job 402
  - checking status of a print job 405
  - holding a print job 404
  - moving a print job 403
  - prioritizing a print job 403
  - releasing a print job 404
  - starting 356

- whatis command 357
- whence built-in command
  - Korn or POSIX shell 466
- whereis command 356
- while built-in command
  - C shell 501
- who am i command 336
- who command 192, 336, 337
- whoami command 336
- wildcard characters 380
  - asterisk 380
  - definition 437
  - question mark 381
- WLM
  - API 231
- words
  - counting number of 388
  - definition 437
  - reserved in Korn or POSIX shell 447
- working directory 138
- Workload Manager
  - API 231
  - starting and stopping 201
- write scheduling policy 50
- write-verify policy 57
- wsm command 356
- wtmp errors
  - fixing 251

## X

- X Server
  - startup files 427
- X terminal 348
- xinit command 428
- xlock command 423

## Y

- Yellow Pages 534
  - BSD System Managers 534

## Z

- zcat command 161
- zero file allocations 131
- zombie processes 362



---

# Readers' Comments — We'd Like to Hear from You

**AIX 5L Version 5.3**  
**Operating system and device management**

**Publication No. SC23-5204-01**

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: [aix6koub@austin.ibm.com](mailto:aix6koub@austin.ibm.com)

If you would like a response from IBM, please fill in the following information:

\_\_\_\_\_  
Name

\_\_\_\_\_  
Address

\_\_\_\_\_  
Company or Organization

\_\_\_\_\_  
Phone No.

\_\_\_\_\_  
E-mail address



Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
Information Development  
Department 04XA-905-6C006  
11501 Burnet Road  
Austin, TX 78758-3493



Fold and Tape

Please do not staple

Fold and Tape







Printed in U.S.A.

SC23-5204-01

