IBM Advanced Interactive Executive for the PS/2 and System/370 Managing the Operating System Version 1.2.1

Document Number SC23-2293-01

Copyright IBM Corp. 1985, 1991

IBM Advanced Interactive Executive
for the PS/2 and System/370

Managing the Operating System

Version 1.2.1

Document Number SC23-2293-01

Copyright IBM Corp. 1985, 1991

Edition Notice

Third Edition (March 1991)

This edition applies to Version 1.2.1 of the IBM Advanced Interactive Executive for the System/370 (AIX/370), Program Number 5713-AFL, and to Version 1.2.1 of the IBM Advanced Interactive Executive for the Personal System/2 (AIX PS/2) and PS/55, Program Number 5713-AEQ, and to all subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department 52QA MS 911
Neighborhood Road
Kingston, NY 12401
U.S.A.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

¦ Copyright International Business Machines Corporation 1985, 1991. All rights reserved.
¦ Copyright AT&T Technologies 1984, 1987, 1988.
¦ Copyright TITN, Incorporated 1984, 1989.
Note to U.S. Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

FRONT_1 Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

Subtopics

Trademarks and Acknowledgments

¦ Copyright IBM Corp. 1985, 1991

Trademarks and Acknowledgments

The following trademarks apply to this book:

IBM is a registered trademark of International Business Machine Corporation.

AIX is a registered trademark of International Business Machine Corporation.

Personal System/2 and PS/2 are registered trademarks of Internationa Business Machines Corporation.

System/370 is a trademark of International Business Machine Corporation.

The Rand MH Message Handling System was developed by the Ran Corporation and the University of California.

The Network File System (NFS) was developed by Sun Microsystems, Inc NFS is a trademark of Sun Microsystems, Inc. Sun Microsystems is a registered trademark of Sun Microsystems, Inc.

Ethernet is a trademark of Xerox, Inc

DEC, VT100, and VT200 are registered trademarks of Digital Equipmen Corporation.

RT is a registered trademark of International Business Machine Corporation.

UNIX is a registered trademark of UNIX System Laboratories, Inc. i the USA and other countries.

DOS Merge and PCI are trademarks of Locus Computing Corporation

INed is a trademark of INTERACTIVE Systems Corporation

Portions of the code and documentation were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley Campus of the University of California under the auspices of the Regents of the University of California.

¦ Copyright IBM Corp. 1985, 1991

About This Book

This book provides instructions for managing the Advanced Interactive Executive (AIX) Operating System. It shows you how to keep the operating system in good working order, how to customize the operating system, and how to maintain data communication facilities.

Most of the procedures in this book are described from the point of view of a person working at a PS/2 (or PS/55) running AIX PS/2 and functioning as a single-user, standalone unit or one of the hosts in an AIX cluster.

In many cases, the procedure being described applies to the cluster as a whole and can be done from any position in the cluster, such as from the console of a System/370 machine or from the keyboard of an ASCII terminal attached to the cluster.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

Who Should Read This Book

This book is written for anyone who is responsible for managing the AIX Operating System. If more than one person uses your system, system management responsibilities may be given to one person or shared among several. If you are the only person using your system, you still must perform certain system management tasks.

¦ Copyright IBM Corp. 1985, 1991

What You Should Know

This book assumes that you know how to operate your workstation. Before using this book, make sure that your workstation is set up and the AIX Operating System is installed. In addition, you should have a **user name** and possibly a password. If your system is not installed, see the installation manual for your workstation. If you need a user name, see "Creating, Changing, and Removing Accounts - The adduser Command" in topic 1.2.5.1.

Many of the tasks in this book require you to use one of the AIX Operating System text editing programs. The following editing programs are available:

ed (see *AIX Operating System Commands Reference*)

INed (see *INed Guide*)

vi (see *Text Formatting Guide*).

In addition, if you are responsible for managing an AIX cluster, especially a cluster that includes System/370s, you should read the *AIX/370 Administration Guide* for a thorough understanding of the interactions that take place in such an environment. You may also find it helpful to read *AIX/370 Planning Guide* and the *General Information* manual to get a basic understanding of an AIX cluster and *Guide to Multibyte Character Set (MBCS) Support* to get an understanding of the AIX multibyte character set.

¦ Copyright IBM Corp. 1985, 1991

How to Use This Book

You can use this book in one of two ways:

As a training manual. Read it from beginning to end. This will giv you a general understanding of the AIX Operating System.

As a reference manual. Use the "Contents" and "Index" to locat particular topics if you need to refresh your memory or learn more details about the AIX Operating System. You may find it helpful to refer to Chapter 4, "Additional System Management Topics," looking for features or capabilities that you can use to make the AIX system better suit your needs.

This book includes a glossary of terms, which you can refer to if you are unfamiliar with a term used in this book. A complete index aids you in finding references quickly.

In the text, whenever you are told to ***enter*** a command or other information, you should type the information and then press the **Enter** key.

Subtopics
Quick-Reference Boxes
Highlighting

¦ Copyright IBM Corp. 1985, 1991

Quick-Reference Boxes

Where appropriate, the chapters and major sections of this book begin with boxes containing quick reference information. For example:

To Use the Quick Reference Boxes

1. Skip the quick reference boxes the first time you read a section.
2. Use the quick reference boxes as a fast path through the book.
3. Refer to the quick reference boxes to refresh your memory.

You can skip the box the first time you read a section. You can use the boxes for reference after you are generally familiar with the contents of a section or chapter. The boxes make a convenient fast path through the book, but they are not comprehensive nor are they intended to take the place of the explanatory information in each section.

After the quick-reference boxes, each chapter takes the same general approach to the topics it covers--a series of explanations followed by examples. The examples build upon each other; in many instances, an example uses a file created in a previous example. Therefore, if you intend to follow the examples on your system, it is important to work through each chapter from beginning to end.

¦ Copyright IBM Corp. 1985, 1991

Highlighting

This book uses different type styles to distinguish among certain kinds of information. General information is printed in the standard type style (the type style used for this sentence). The following type styles indicate other types of information:

New terms
Each time a new term is introduced, its first occurrence is printed in bold italics (for example, "the AIX Operating System ***file system***").

System parts
The names for keys, commands, files, and other parts of the system are printed in bold (for example, "the **cp** command").

Variable information
The names for unique information that you must provide are printed in italics (for example, "type **yourname**").

Information you are to type
Many examples in this book are designed for you to try on your own system. The information that you should type is printed in monospace type style (for example, type **ls text**. This type style is also used for the names of files that you create as you work through this book (for example, "create a file named **afile**"). In addition, any characters that have a special meaning are printed in monospace type (for example, "the **&** and **&&** operators have different uses").

¦ Copyright IBM Corp. 1985, 1991

Related Publications

For additional information, you may want to refer to the following publications:

AIX Access for DOS Users Administrator's Guide, SC23-2042, explains how to install and administer the AIX Access for DOS Users program on the IBM PS/2, RT, and System/370 computers running the AIX Operating System with the AIX DOS Server. It covers the responsibilities for installation, daily operation, and maintenance of the AIX Access program.

AIX/370 Administration Guide, SC23-2088, describes such administrative tasks as updating the file system, backing up files, and fine-tuning and monitoring the performance of the operating system.

AIX Operating System Commands Reference, SC23-2292 (Vol. 1) and SC23-2184 (Vol. 2), lists and describes the AIX/370 and AIX PS/2 Operating System commands.

AIX/370 Diagnosis Guide, SC23-2090, describes procedures and tools that can be used to define and categorize symptoms of problems that may occur during daily operation.

AIX/370 General Information, SC23-2062, describes the functions and capabilities of AIX/370 and its position in the AIX family of products.

DOS PS/2 DOS Merge User's and Administrator's Guide, SC23-2045, shows how to use DOS in the AIX environment, including running DOS and AIX programs simultaneously and running AIX commands from the DOS environment. It also shows how to install the DOS Merge software and how to perform essential system maintenance activities, such as adding user accounts, backing up the file system, and setting up terminals.

Installing and Customizing the AIX PS/2 Operating System, SC23-2290, provides step-by-step instructions for installing the AIX PS/2 Operating System and related programs. This book also shows how to customize the operating system to suit the user's specific needs and work environment.

Installing and Customizing the AIX/370 Operating System, SC23-2066, provides step-by-step instructions for installing the AIX/370 Operating System and related programs. This book also shows how to customize the operating system to suit the user's specific needs and work environment.

AIX PS/2 Keyboard Description and Character Reference, SC23-2037, describes the characters and keyboards supported by the AIX PS/2 Operating System. This book also provides information on keyboard position codes, keyboard states, control code points, code-sequence processing, and non-spacing character sequences.

AIX Operating System Messages Reference, SC23-2294, lists messages displayed by the AIX Operating System and explains how to respond to them.

AIX/370 Planning Guide, GC23-2065, describes the functions and capabilities of the AIX/370 Operating System and lists the requirements for all supported hardware and software. This book also

¦ Copyright IBM Corp. 1985, 1991

includes information to assist with planning for installation and customization of the operating system.

AIX TCP/IP User's Guide, SC23-2309, describes the features of TCP/IP and shows how to install and customize the program. It includes reference information on TCP/IP commands that are used to transfer files, manage the network, and log into remote systems.

AIX Operating System Technical Reference, SC23-2300 (Vol. 1) and SC23-2301 (Vol. 2), describes the system calls and subroutines a programmer uses to write application programs. This book also provides information about the AIX Operating System file system, special files, miscellaneous files, and the writing of device drivers.

Using the AIX Operating System, SC23-2291, shows the beginning user how to use AIX Operating System commands to do such basic tasks as log in and out of the system, display and print files, and set and change passwords. It includes information for intermediate to advanced users about how to use communication and networking facilities and write shell procedures.

Guide to Multibyte Character Set (MBCS) Support, GC23-2333, explains the MBCS encoding scheme used in AIX Version 1.2.1. The guide acts as a "road map" for the rest of the MBCS changes in the AIX document set.

For more information on the X.25 Host Interface, the following publication is recommended:

Defense Data Network X.25 Host Interface Specification, BBN Communications Corporation, 70 Fawcett Street, Cambridge, MA 02238, 1983.

¦ Copyright IBM Corp. 1985, 1991

Table of Contents

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

Figures

¦ Copyright IBM Corp. 1985, 1991

Tables

¦ Copyright IBM Corp. 1985, 1991

1.0 Part 1. Managing the AIX Operating System

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.1 Chapter 1. Introduction to System Management

Subtopics
1.1.1 Contents
1.1.2 About This Chapter
1.1.3 General System Structure
1.1.4 The File System - Background for System Management
1.1.5 The Base AIX File Systems
1.1.6 Finding and Viewing System Files
1.1.7 MBCS Overview
1.1.8 Support of Calligraphic Languages
1.1.9 Hardware and Software in Kanji-Capable Clusters
1.1.10 Compatibility and Interoperability

¦ Copyright IBM Corp. 1985, 1991

1.1.1 Contents

¦ Copyright IBM Corp. 1985, 1991

1.1.2 About This Chapter

The AIX Operating System is a powerful and versatile tool. Generally, you use the operating system to do your work--for example, to process data, edit text files, run spreadsheet programs, and communicate with other users. There are times, however, when you must work on the operating system itself--for example, to add new users to the system or to create, back up, and repair file systems. The work you do on the operating system is called ***system management***.

This chapter includes the background information necessary for effective system management. While this chapter and the remainder of this book provide much specific information and careful guidance, you should understand that the job of system management is not the same on any two systems, and that system management requirements may change considerably on a particular system over time. Thus, in addition to becoming familiar with the information in the rest of this book, the best way to manage your system effectively is to continue to learn about the AIX system and the requirements placed on your system by its users.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System,* and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

1.1.3 General System Structure

You can think of the AIX system as having four levels, as shown in Figure 1-1, below. At the first level is the computer ***hardware***, especially the system unit. At the second level is the AIX Operating System kernel, a set of programs that manages the resources of the computer (main storage, disk storage, display stations, and printers). The third level, libraries, is a set of routines providing additional functions (such as copy files and display files). At the fourth level are the ***application programs***, such as the shell, text editors, electronic spreadsheets, calendars, and communications programs. Together, the AIX Operating System kernel, support programs, and the application programs comprise the computer's ***software***.

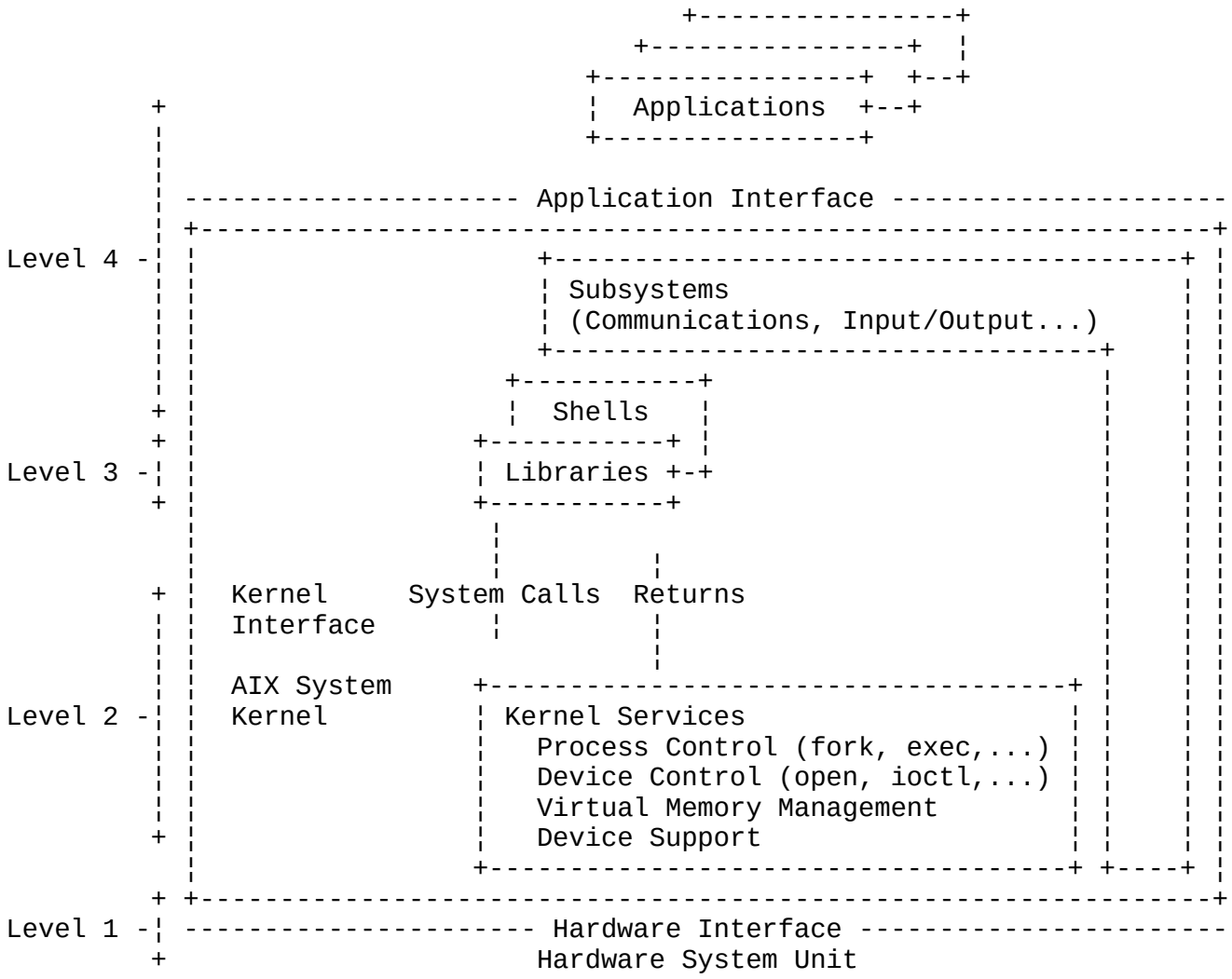


Figure 1-1. Parts of the AIX Operating System

Subtopics
1.1.3.1 The Kernel
1.1.3.2 The Shell

¦ Copyright IBM Corp. 1985, 1991

1.1.3.1 The Kernel

The kernel is a program that controls the computer system's physical components (the hardware) and makes the functions of the hardware available to system utilities and applications.

Through its management of hardware, the kernel provides ***virtual*** resources to the shell and other programs. A virtual resource, such as virtual memory or a virtual terminal, is a simulation of a real resource. Virtual resources are based on the real resources of the system. The use of virtual resources makes the computer system more flexible than it would be if the operating system had to use real resources directly.

Among the tasks done by the kernel are:

Scheduling of processes (programs or commands) so that each proces gets its share of processing unit time

Handling data transfer (input and output) among system devices (fo example, keyboards, displays, printers, and storage media)

Managing the file system so that data can be stored and retrieved i an orderly way.

The shell and other programs use ***system calls*** to access the kernel. You do not need to understand system calls in order to manage your computer system. For those who do need to understand them (programmers, for example), the system calls are described in *AIX Operating System Technical Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.1.3.2 The Shell

The shell, often called an ***interface*** or a ***command interpreter***, is the part of the operating system that makes it possible for you to use the kernel. The shell is actually an ordinary program that is said to run *on top of* the kernel.

You can use different interfaces or different versions of the standard shell. Following is a list of the interfaces available on the AIX Operating System:

AIX (Bourne) shell, the standard interfac

Usability Services, described in *AIX Usability Services User's Guide*

Merge, described in *AIX PS/2 DOS Merge User's and Administrator's Guide*

C shell, described under **csh** in *AIX Operating System Commands Reference*.

The information in this book is about the standard AIX shell.

The shell makes it possible for other programs to run by starting kernel processes. The kernel manages the way its services are shared among programs.

The shell also makes it convenient for you to use certain basic kernel services, such as:

Device-independent **input** and **output** (used by the shell to accomplish I/O redirection)

Pipes.

For more information about redirecting input and output and about pipes, see *Using the AIX Operating System*.

Finally, the shell is a ***command programming language***. That is, with the shell, you can develop your own commands or shell procedures without having to use a conventional programming language. (For more information about the shell, see *Using the AIX Operating System* and **sh** or **csh** in *AIX Operating System Commands Reference*.)

¦ Copyright IBM Corp. 1985, 1991

1.1.4 The File System - Background for System Management

Note: Before you begin this section, you should be familiar with the way the file system appears to an ordinary system user. For information on file systems, see *Using the AIX Operating System*.

A file system is a complete directory structure, including a root directory and any subdirectories and files beneath it. File systems are confined either to a single ***minidisk*** (partition of a fixed disk) or to a diskette (one file system per diskette). Some of the most important system management jobs have to do with file systems, specifically:

Allocating space for file systems (on minidisks

Creating file system

Making file system space available to system user

Monitoring file system space usag

Backing up file systems to guard against data loss in the event o system or disk failures

Maintaining file systems in a consistent state

There are certain system commands designed specifically for system management. Of those commands, the ones you probably will use regularly for working with file systems are:

backup Performs a full or incremental backup of a file system.

clri Clears inodes (used when file system inconsistencies cannot be corrected by **fsck**).

dd Copies data directly from one device to another (for making file system back ups).

df Reports the amount of space used and free on a file system.

fsck Checks file systems and repairs inconsistencies.

fsdb Permits interactive exploration and patching of a file system. It is a tool for the expert or student of file system structure, rather than a maintenance utility.

mkfs Makes a file system of a specified size on a specified minidisk.

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.

umount Removes a file system from the system-wide naming structure, making the files and directories in the file system inaccessible.

primrec Performs user level reconciliation.

¦ Copyright IBM Corp. 1985, 1991

Regardless of the device they reside on (diskette or minidisk), all file systems have the same structure. Thus, you can move a file system from one device to another, provided the destination device is large enough. Internally, disk space is managed for file systems in ***logical blocks*** that can contain 4096 ***bytes*** of data. In the following discussion, a reference to block is a reference to logical block.

A file system has four major parts:

Bootstrap bloc

Superbloc

Inode

Data blocks

¦ Copyright IBM Corp. 1985, 1991

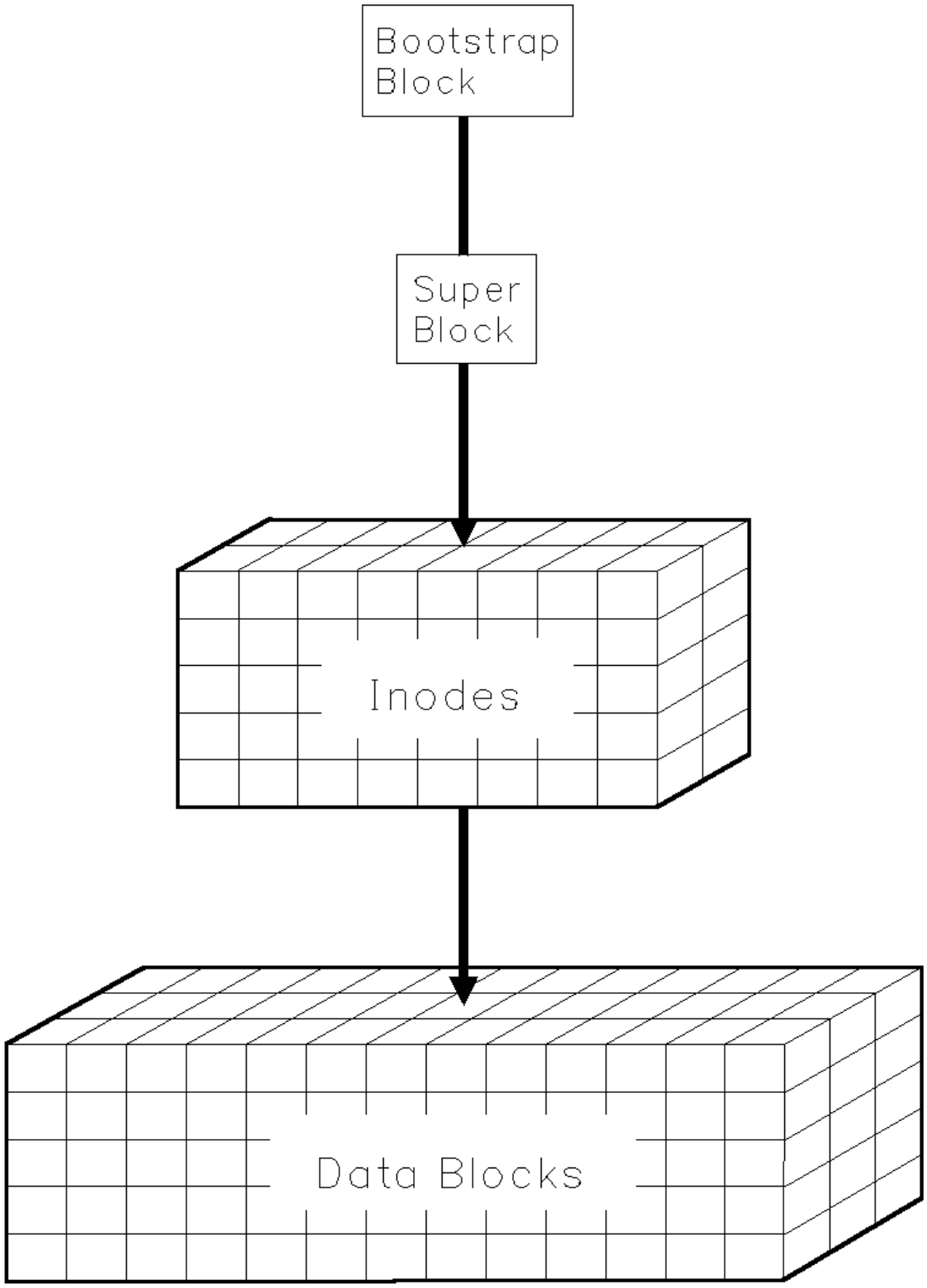


Figure 1-2. Major Parts of an AIX File System

Subtopics
1.1.4.1 Bootstrap Block
1.1.4.2 The Superblock
1.1.4.3 Inodes
1.1.4.4 Data Blocks

¦ Copyright IBM Corp. 1985, 1991

1.1.4.1 Bootstrap Block

The first block of every file system (block 0) is reserved for a ***bootstrap***, or initialization program. The bootstrap block is not actually part of the file system structure. File system data begin on block 1 of the minidisk.

¦ Copyright IBM Corp. 1985, 1991

1.1.4.2 The Superblock

Block 1 of every file system is called the ***superblock***. Among the most important information the superblock contains are:

Total size of the file system (in file system blocks)

Number of file system blocks reserved for inodes (explained below)

Name of the file system

Volume ID of the minidisk or diskette

Date of the last superblock update

Global file system number (GFS #), the unique name for each fil system.

The head of the ***free-block list***, a chain that contains all of the free blocks in the file system (the blocks available for allocation). When new blocks are allocated to a file, they are allocated from this list. When a file is deleted, its blocks are returned to this list.

A list of free inodes. This is a partial list of inodes available t be allocated for newly created files. For more detailed information regarding inodes, see *AIX Operating System Technical Reference*, especially **fs** and **inode**.

¦ Copyright IBM Corp. 1985, 1991

1.1.4.3 Inodes

After the superblock, there is a group of blocks that contain ***inodes*** (descriptions of the individual files in the file system). There is one inode for each possible file in the file system. Each inode is associated with an inode number, or i-number. For any file system, there is a maximum number of inodes and, thus, a maximum number of files that the file system can contain. This maximum number varies with the size of the file system. The first inode (inode 1) on every file system is unnamed and unused. When associated with files, the remaining inodes contain the following information:

File type. Possible file types are ordinary file, directory, block device, character device, symbolic link, hidden directory, multiplexed file, and first-in-first-out (also sometimes called ***FIFO*** or ***named pipe***).

File owner. The inode contains the user and group IDs associated with the file.

Protection information. This is a specification of **r**ead, **w**rite, and e**x**ecute access for the owner, members of the group associated with the file, and others. It also includes other mode information specified by the **chmod** command. (For a description of protections, see *Using the AIX Operating System*.)

Link count (or hard link count). A directory entry (link) consists of a name and the number of the inode in the file system that represents the file (its ***i-number***). The link count specifies the number of directory entries that refer to the file. A file is deleted when the link count becomes zero. That is, its inode is returned to the list of free inodes and its associated data blocks are returned to the free-block list. (For a description of links, see *Using the AIX Operating System*.)

Size of the file (in bytes).

Date the file was last accessed.

Date the file was last modified.

Date the inode was last modified.

Pointers to data blocks. These pointers indicate the location of the data blocks on the physical disk.

Inode 2 must correspond to the root directory for the file system. All other files in the file system are below the root directory in the file system. Beyond inode 2, any inode can be assigned to any file. Similarly, any data block can be assigned to any file. Neither inodes nor blocks have to be allocated in any particular order. For more information on inodes, see *AIX Operating System Technical Reference*, especially **fs** and **inode**.

¦ Copyright IBM Corp. 1985, 1991

1.1.4.4 Data Blocks

Beyond the inodes, the file system consists of ***data blocks***. Each data block can contain 4096 bytes of information. The inode points directly to the first 10 data blocks of the file. Figure 1-3 shows the direct pointers from an inode to the data blocks.

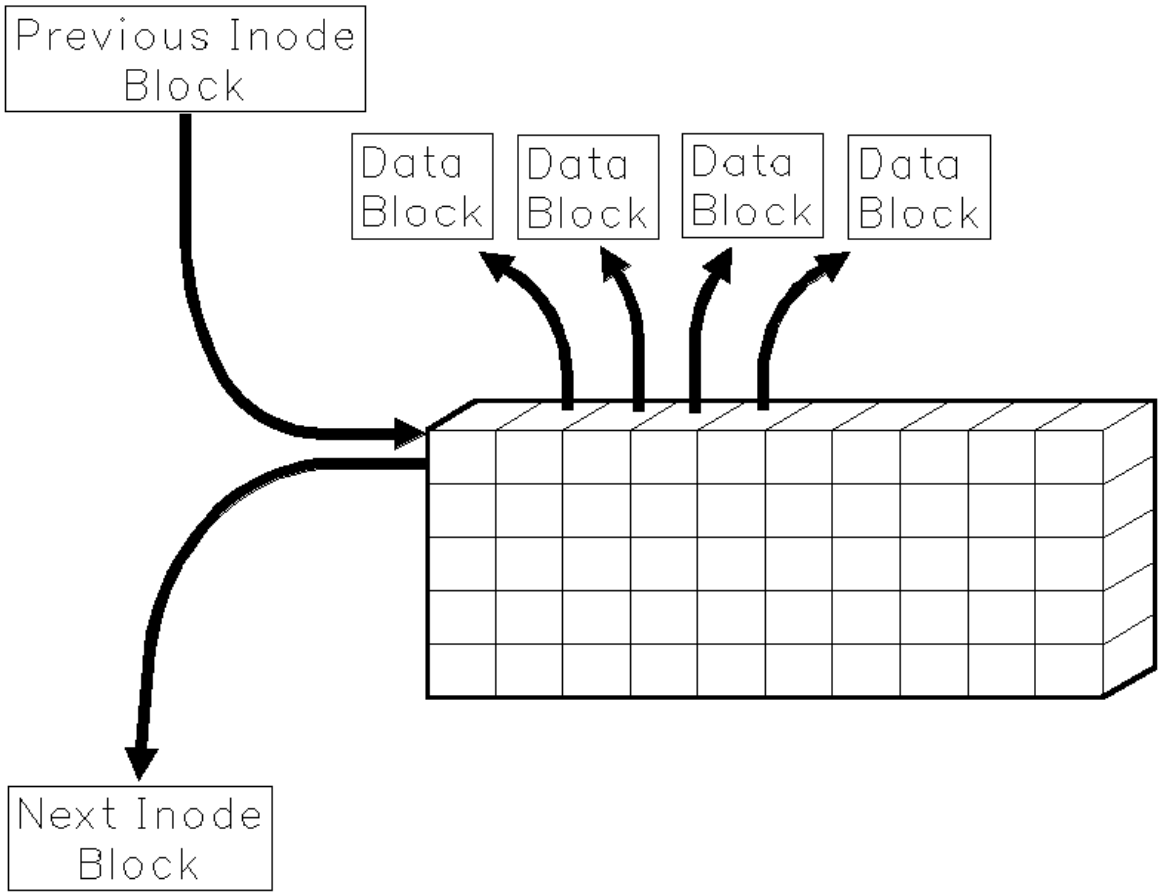


Figure 1-3. Direct Pointers from an Inode to Data Blocks

When a file contains more than 10 blocks of data, block number 11 is an ***indirect block***. The indirect block contains ***pointers*** to 1024 additional data blocks. A file that has one indirect block is called ***single-indirect*** and has a maximum size of 1035 blocks (the first 10 data blocks, plus the indirect block, plus the 1024 data blocks.)

If a file is larger than 1034 blocks, block number 12 is a ***double-indirect*** block. A double-indirect block points to an indirect block that contains 1024 pointers to other indirect blocks. Each of those indirect blocks in turn points to 1024 data blocks. Thus, a double-indirect file has a maximum size of 1034 + (1024 * 1024), or 1,049,610 blocks. If a file is 384 or fewer bytes in size, the file data is stored directly in the inode, and no data blocks are allocated for the file.

Figure 1-4 shows the relationship of data blocks and indirect blocks associated with an inode.

¦ Copyright IBM Corp. 1985, 1991

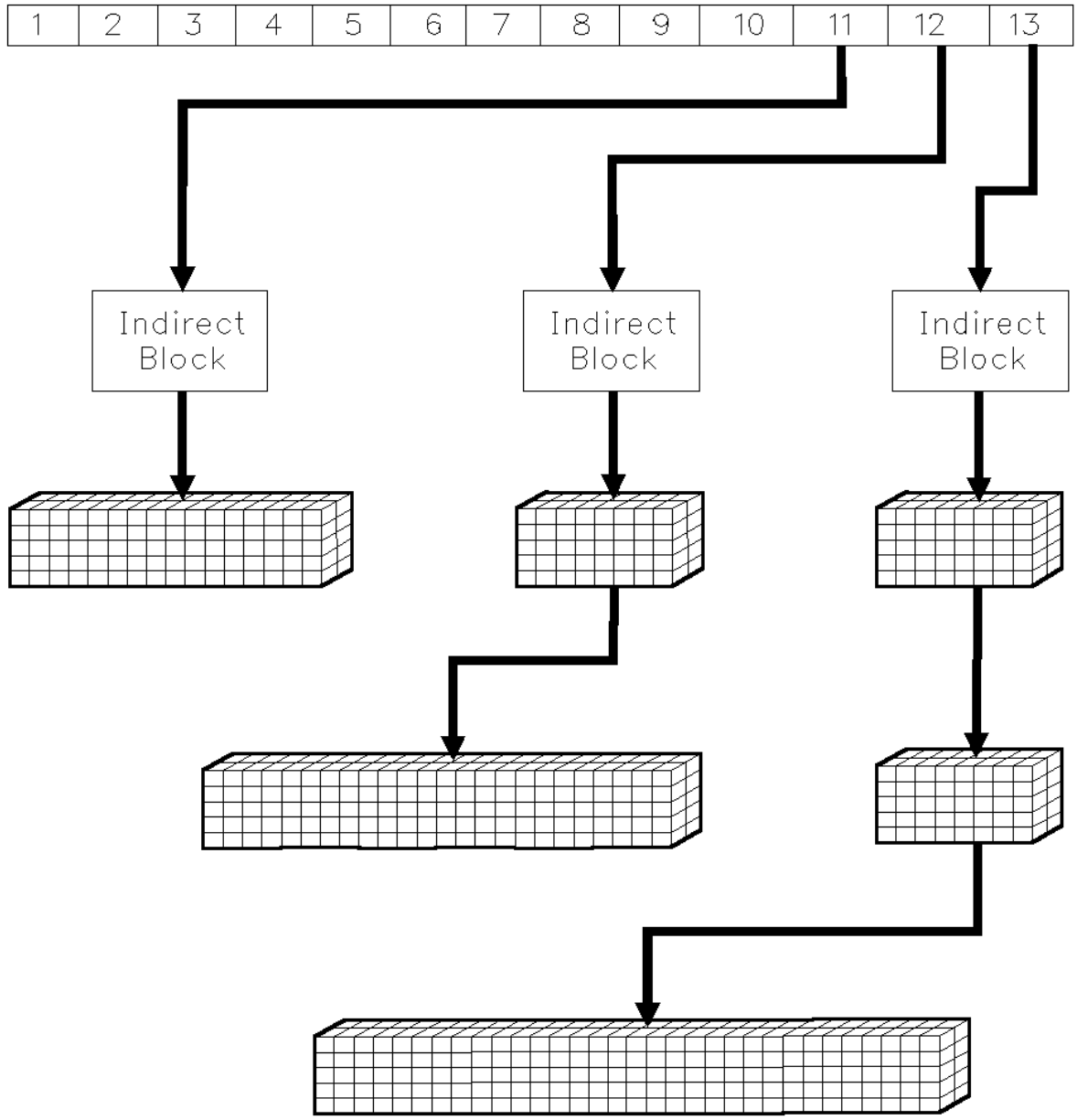


Figure 1-4. The Relationship of Data Blocks and Indirect Blocks

¦ Copyright IBM Corp. 1985, 1991

1.1.5 The Base AIX File Systems

Any machine being used as a host in a Transparent Computing Facility (TCF) cluster is called a cluster site. For more information about clusters, see *AIX/370 Planning Guide, Using the AIX Operating System* (Chapter 5), and *AIX/370 Administration Guide* (Chapter 2). Each cluster site has at least three separate file systems:

/ - the replicated root file system. (This is the root and certain files and directories described under the root file system.)

/machinename or **<LOCAL>** file system.

/u - the user file system.

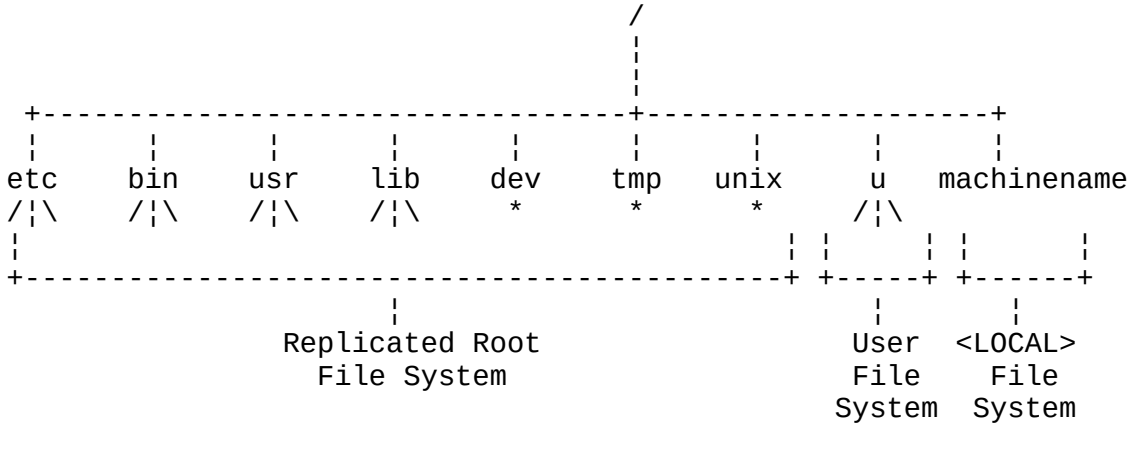
A single cluster site has one replicated root file system. When multiple computer sites are connected together, the replicated root file system is duplicated by the system on each cluster site.

The **<LOCAL>** file system is the file system named for that particular cluster site. It mainly contains site specific information or temporary files.

The **/u** file system contains the home directories of all system users.

Each of these file systems (the replicated root, **<LOCAL>**, and user) reside on a separate minidisk. The system gains access to the replicated root file system when it initializes itself. The **<LOCAL>** and user file systems are mounted onto or
attached to the replicated root file system. The **<LOCAL>** file system is mounted automatically as part of the system initialization process (see Figure 1-5).

The **mount** command instructs the operating system to make a file system available for use. The **/machinename** directory is the "mount point" for the **<LOCAL>** file system, where **machinename** is the cluster machine name (initially **aix370**). The **<LOCAL>** and user file systems are integrated onto the replicated root file system so that they appear to be part of one seemingly continuous directory structure (see Figure 1-5).



/¦\ - indicates other files and directories
* - indicates a symbolic link to the corresponding directory or file in the <LOCAL> file system

Figure 1-5. All File Systems on One Cluster Site (Major Files and

¦ Copyright IBM Corp. 1985, 1991

Directories)

For example, suppose your cluster includes a PS/2 (or PS/55) named **hera**.

As part of its setup procedure, **hera's** fixed disk has been divided into two partitions, each of which acts as a separate minidisk. The local file system is placed on one minidisk and the replicated root file system on the other. There may also be other partitions (minidisks) allocated for other purposes. As another part of the setup procedure, an empty directory called **/hera** is created on the root.

If **hera** has been shut down, when it is reinitialized, any parts of the replicated root file system that have changed since its shutdown are copied from the primary site of the cluster into their respective locations on the root file system in the first partition of **hera's** fixed disk.

The local file system, which is also called **/hera**, will then be attached to the **/hera** directory by the **mount** command.

This recopy and mount procedure occurs every time the machine is reinitialized. This assures that the global file system is always up to date and consistent with what is contained on other cluster sites. In addition, each site has the benefit of its own site-specific files kept in its own local file system.

The concepts of hard links, symbolic links, and **<LOCAL>** aliases are useful in understanding the general file system organization.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.1.5.1 Hard Links and Symbolic Links

The computer system has two kinds of links that are used to create multiple names for files:

Hard links The link or hard link mechanism (sometimes called the UNIX link) permits multiple names to be created for a single file (not a directory). This link can only be made within the same file system.

Symbolic links The link mechanism that allows a user to create a pointer to any file or directory. Unlike a standard UNIX hard link, a symbolic link can point to a directory as well as a file; and it can cross a file system boundary.

A symbolic link is a special file that contains the path name of the destination file or directory. When the special file is read, the contents of the file (the path name) is used to identify the destination of the symbolic link. For example, if a file named **status.joe** (in directory **/u/sue/projects**) is set up as a symbolic link to **/u/joe/status**, **/u/sue/projects/status.joe** contains the path name **/u/joe/status**. When Sue runs the **cat** command on the file **status.joe**, the system substitutes **/u/joe/status** for **status.joe** (when Sue is in the directory **/u/sue/projects**). Figure 1-6 shows an example of a symbolic link.

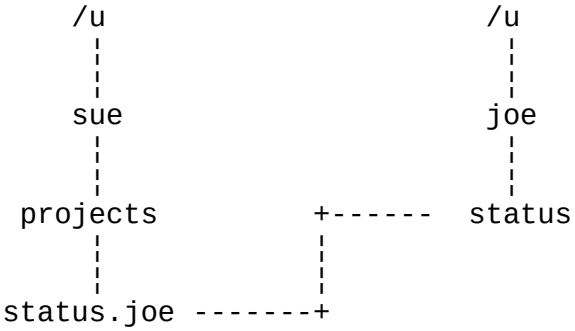


Figure 1-6. Example of a Symbolic Link

Several files and directories in the replicated root file system are symbolic links to other files and directories in the **<LOCAL>** file system.

¦ Copyright IBM Corp. 1985, 1991

1.1.5.2 The <LOCAL> Alias

The **<LOCAL>** alias is a special type of substitution mechanism. Under special circumstances, the string **<LOCAL>** is redefined by the operating system to be the path name of the directory on which the local file system is mounted. For example, you may decide to name your file system **arizona**. (This is done during installation and is described in the AIX Operating System installation manual.)

Symbolic links and **<LOCAL>** aliases can be used together to form a type of link between the replicated root file system and the **<LOCAL>** file system. For example, say the **<LOCAL>** file system is mounted onto the replicated root file system as **/arizona**. If a user types **ls/tmp**, then the **ls** program accesses **/arizona/tmp**. This is because **/tmp** is a symbolic link which points to **<LOCAL>/tmp**. In following or expanding the symbolic link, the system recognizes a use of the **<LOCAL>** alias mechanism and causes the symbolic link to access **/arizona/tmp**. In other words, the string, **<LOCAL>** is only recognized as an instance of the **<LOCAL>** alias if it is the first seven characters of the destination for the symbolic link. At this point, the string, **<LOCAL>**, is replaced by the value of the **<LOCAL>** alias for the current process (see Figure 1-7).

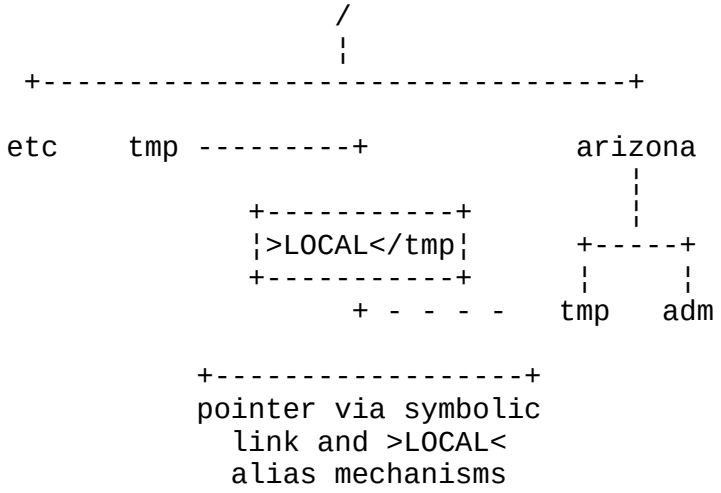


Figure 1-7. Use of Symbolic Link and <LOCAL> Alias

The **<LOCAL>** alias is translated into different strings on different cluster sites for different processes.

For example, to create a symbolic link to the **<LOCAL>** file system called **/etc/utmp** enter:

```
ln -s "<LOCAL>/utmp" /etc/utmp
```

This command creates a symbolic link **/etc/utmp** that points to a different file **utmp** on each site in the cluster. The quotes are required because the symbols < and > have special meanings to the shells.

For more information regarding <LOCAL> in a TCF environment, see *Using the Operating System* (Chapter 5).

¦ Copyright IBM Corp. 1985, 1991

1.1.5.3 The Replicated Root File System

The replicated root file system contains files that are critical to the normal operation of the system duplicated on each cluster site by the system. Examples of critical files include binaries for system utilities (the basic programs that run the operating system).

The following are brief descriptions of the major files and directories in the replicated root file system.

/ The highest directory in the file system hierarchy, usually called the ***root*** directory.

/dev A symbolic link to the **<LOCAL>/dev**.

/bin A directory of the UNIX program binaries (includes **sh** and **csh**). These are the basic commands that run the system. Additional commands are contained in **/usr/bin** and **/usr/lib**.

/etc A directory containing most of the commands required for system startup and maintenance. Several files in this directory (**ddi**, **inittab**, and **filesystems**, for example) are symbolic links to the corresponding **<LOCAL>** file system files.

master A file (in **/etc**) containing information about system devices used by the **config** program to create configuration files.

passwd A file (in **/etc**) containing the information that defines users for the system (including encrypted passwords).

group A file (in **/etc**) containing the information that defines groups of users to the system.

/usr A directory containing other directories of commands. Certain directories (such as **spool**) are symbolic links to the corresponding **<LOCAL>** file system file.

/lib A directory containing files that make up the C language programming library and parts of the C compiler.

/tmp A symbolic link to the **<LOCAL>/tmp**.

unix A symbolic link to **<LOCAL>/unix**.

For more information regarding Replicated files in a TCF environment, see *Using the AIX Operating System* (Chapter 5).

¦ Copyright IBM Corp. 1985, 1991

1.1.5.4 The <LOCAL> File System

The **<LOCAL>** file system contains directories and files that are an extension of the replicated root file system.

The following are brief descriptions of the major directories and files in the **<LOCAL>** file system:

dev A directory containing the special files that allow input and output operations to system devices (such as fixed disks, diskette drives, and terminals). For more information on special files and system devices, see "The Input/Output System" in topic 1.3.4.

inittab A file containing the system initialization table.

ddi A directory containing files that describe system devices (for example, printers); **ddi** stands for device dependent information.

unix A file containing the AIX kernel.

ports A file containing the terminal characteristics initialization table.

lpd A directory containing commands used to send data to system printers and to collect accounting statistics on printer usage.

tmp A directory containing temporary files created by processes.

¦ Copyright IBM Corp. 1985, 1991

1.1.5.5 The User File System

The user file system or **/u** contains all the user's home directories.

¦ Copyright IBM Corp. 1985, 1991

1.1.6 Finding and Viewing System Files

As you learn about the AIX Operating System, you probably will find it very helpful to be able to locate directories and files and look at their contents. The following three commands make it easy for you to do so:

find Searches the file system for a file (or directory) when you know its name. The command:

```
find / -name filename -print
```

searches the file system, from the root directory (**/**) down, for **filename**. If it finds a file with that name, it displays the complete path name for that file. If **filename** is not in the file system, the **find** command displays nothing when it completes. For more information on the **find** command, see *Using the AIX Operating System* and **find** in the *AIX Operating System Commands Reference*.

ls Lists the contents of a directory. The command:

```
ls dirname
```

lists the names of the entries in the directory **dirname**. The name of the directory can be a path name. You can use **ls** command flags to get different types of information about the items in a directory. For information about the different ways to use **ls**, see *Using the AIX Operating System* and **ls** in *AIX Operating System Commands Reference*.

pg Displays the contents of a file one page at a time. The command:

```
pg filename
```

displays the contents of **filename** one page at a time. At the bottom of each page on the screen, the **pg** command displays a colon (**:**). To display the next screen of information, press **Enter**. For more information on the **pg** command, see *Using the AIX Operating System* or **pg** in *AIX Operating System Commands Reference*. (You also can use the **cat** [concatenate] command to display the contents of a file; however, the **cat** command scrolls through the entire file at once, rather than one page at a time.)

For more information about these commands and the different ways in which you can use them, see the appropriate entries in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.1.7 MBCS Overview

Multibyte character set (MBCS) support, which is a feature of AIX/370 and AIX PS/2, is a new system of encoding characters in computer data. With MBCS, AIX is not tied to any single language or even to any family of languages. MBCS support is especially useful for users operating in non-European languages. It allows users to specify Japanese as their primary language. Their workstations then interact with them in Japanese characters. The rest of the machines in the cluster may recognize only the standard ASCII characters of English or one of the extended character sets used by Roman-based European languages.

Subtopics
1.1.7.1 MBCS Cluster

¦ Copyright IBM Corp. 1985, 1991

1.1.7.1 MBCS Cluster

AIX/370 provides MBCS facilities for creating a multilingual AIX that runs on IBM System/370 XA processors. If the proper message catalogs and fonts have been purchased and installed, the system offers:

Input/output features that permit the system to interact with the use in Japanese characters or the extended character sets needed by various European languages

Full *message catalog* facilities that allow the display of standard system messages and prompts in Japanese or several other languages

Printing of text in Japanese or extended Roman-based character set

The ability to display all national characters of a supported languag on terminal screens

One or more collation sequences for the characters of each supporte language

A monetary symbol appropriate to the regio

Decimal punctuation symbols and formats appropriate to the regio

Date and time representations appropriate to the region

AIX PS/2 is a full-function MBCS AIX that runs on the Intel-80386-based processor. In a normal installation in Japan, AIX PS/2 runs on the IBM Model 5570, which is a Japanese-language model of the IBM Personal System/2.

When both of these AIX software products are used together, they provide an integrated software cluster that can interact with users in Japanese or any of the other supported languages. Users receive the benefits of a distributed processing environment fully integrated under TCF. It presents a single-system image, yet interacts with each user in his or her own native language.

¦ Copyright IBM Corp. 1985, 1991

1.1.8 Support of Calligraphic Languages

The original AIX family of operating systems was designed to process input data composed primarily of single-byte characters. The eight bits of each byte are used to represent 256 possible symbols. This is sufficient to support any language written in characters derived from the Roman writing system. This includes English and most European languages. Calligraphic languages, however, are far more difficult to encode. They are written in very large sets of unique characters and cannot be represented by single-byte values. Two or more bytes must be allocated for each character. Familiar examples of calligraphic languages are Chinese and Japanese.

AIX MBCS support allows the creation of a transparent cluster of multibyte-capable computers that can include IBM System/370s, and PS/2s. Attached to such a cluster there is usually an array of other computers acting as workstations. These may include DOS machines, ASCII terminals, and IBM computers (Models 5570 and 5550). Several languages can be supported concurrently on such a cluster. The user may select certain locale-specific options, such as language, character set encoding and collating sequence. A default locale is set for each machine operating as a site in the cluster (or for each standalone site in a noncluster environment). This is selected by the system administrator but may be overridden by the user. This leaves the user free to set the locale for an entire session or even for individual processes within a session.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.1.8.1 Characteristics of Kanji-capable MBCS AIX Systems

AIX/370 and AIX PS/2 can be used by a person whose native language is Japanese. Japanese text can be represented by two phonetic "alphabets" (*Katakana* and *Hiragana*) and a set of pictographic characters called *Kanji*. The user may enter and manipulate text in the form of Katakana, Hiragana or the Roman alphabet. Text is displayed on the screen in the form of Katakana, Hiragana, Kanji, or Roman-based characters.

¦ Copyright IBM Corp. 1985, 1991

1.1.8.2 The Kanji-Capable User Interface under MBCS

MBCS supports Japanese language text in the following areas:

Commands that accept text input allow that input to be entered i either Kanji or ASCII characters

Messages presented to the user will appear in whatever language th user has chosen for his or her own session

Libraries of C language software routines used for programming suppor the processing of either ASCII or Japanese text.

Japanese language users may prefer to operate in Japanese character mode as much as possible. AIX commands are easiest for them to remember and use if they can be renamed and issued to the operating system in Japanese. AIX offers several ways to do this.

Shell scripts can be created under Japanese names by any Japanese user. The contents of the script file can be a series of AIX commands in ASCII characters, but the user uses only the Japanese name. When the name is issued to AIX on the command line, all the commands in the script are run.

Any of the system's commands or scripts accept filename arguments written in Japanese characters. This means users may name their data files with Japanese characters, even though option flags need to be entered in ASCII.

The system also provides two methods for equating frequently used commands strings to Japanese characters, the *link* and the *alias*. *Linking* can be done by any user. The *alias* function is available only to users of the C Shell. Users of either the Bourne shell or the C shell may create a *link* between a command file and a string of Japanese characters. When this string is issued on the command line, AIX runs the command to which it is linked. The command may be a program file or a script. The C Shell also allows aliases to be created for command names. These commands can be programs or shell scripts. The names can be strings of Japanese characters. When the Japanese name is given on the command line, the program or script is run. For both the link and the alias operations, the existing AIX command names are retained, as are the conventions for the entering of options and arguments. Thus a program that has been linked to a Japanese string still takes its option flags in ASCII. The same is true for a command that has been associated with a Japanese alias.

The primary advantage of the aliasing function over linking is that an alias can be equated to an entire string of ASCII characters, including embedded blanks. Thus an alias can be equated to the program name *and* any option flags the user wishes to specify. Thus the string *ls -l* might be aliased to a Japanese name for "long listing". Linking creates a link between the Japanese name and the ASCII program name only. It cannot cover blanks or the option flags. Users of the C shell can thus set up aliases for all their frequently used commands. These aliases could be complete with the flags they usually use.

This means that any Japanese language user (especially C shell users) has facilities which allow nearly all of his ordinary work to be entered as Japanese characters. The results will be presented back as Japanese characters. Only for unusual or complex manipulations will the user need to resort to ASCII. An ordinary clerical or data entry user should be able to do nearly all daily routine activities in his or her own native language. Only the more technical user (programmers and system

¦ Copyright IBM Corp. 1985, 1991

administrators, for instance) need frequent resort to ASCII. These are exactly the sort of users most likely to be familiar with English.

¦ Copyright IBM Corp. 1985, 1991

1.1.9 Hardware and Software in Kanji-Capable Clusters

The standard terminals for AIX in Kanji mode are the IBM Models 5570 and 5550. The Model 5570 is a Japanese-specific model of the 386-based PS/2 computer, equipped with a Japanese language keyboard. It needs X-Windows software and the proper Kanji fonts, since MBCS-Kanji applications for the Model 5570 run in an X-Windows environment. The Model 5550 is a Japanese-specific version of the XT class machine. It uses software that produces Japanese characters without X-Windows.

Any computer equipped with X-Windows and the necessary fonts can *display* Kanji output. However, only the Models 5570 or 5550 allow the user to respond with Japanese language input. Normally, the Model 5570 is used as one of hosts of the cluster. It typically has one or more 5550s attached to it by serial line, acting as intelligent terminals. The role of cluster host may also be served by a normal PS/2, and Model 5550s may be similarly attached by RS232 line. The console of the PS/2 functions only in ASCII mode, however. For some clusters this may be sufficient. The Model 5570 can also be attached to the cluster via Ethernet or Token-Ring. When equipped with X-Windows and the proper fonts they may function as X Terminals.

Typically, as the number of Japanese language users grows, more 5570 units are added to the system to function as hosts. Additional 5550s can then be attached as intelligent terminals. ASCII terminals can also be attached to the system and function as they would under standard AIX (without Japanese language support). These can be used as terminals to support non-Kanji users.

For users to interact with the systems in Japanese characters, the system needs to be provided with a set of Japanese message catalog files, collation table files, and character set files. Each user who wishes to operate in Japanese needs to set certain environment variables. This is covered in Chapter 7.

¦ Copyright IBM Corp. 1985, 1991

1.1.10 Compatibility and Interoperability

The following sections explore the abilities of MBCS AIX to communicate with other systems and to run programs created on other systems.

Subtopics
1.1.10.1 Compatibility with Earlier AIX Systems
1.1.10.2 Interoperability with Other Systems

¦ Copyright IBM Corp. 1985, 1991

1.1.10.1 Compatibility with Earlier AIX Systems

AIX Version 1.2.1 is highly compatible with previous versions of AIX/370 and AIX PS/2.

Data files created on a standard English-language AIX system (Version 1.2 or earlier) are accepted and processed correctly by AIX Version 1.2.1. Files created under AIX Version 1.2.1 can be processed by previous AIX systems provided they contain no multibyte characters. This means that an AIX Version 1.2.1 system may share data freely with a previous AIX system, as long as the data is in ASCII format. AIX provides facilities for the creation of such all-ASCII files. To enter straight ASCII text at the Kanji keyboard, for instance, the user switches to ASCII mode by pressing a single control key. Programs written for a standard AIX system run on an MBCS AIX Version 1.2.1 system as long as they are not given data containing multibyte characters. This compatibility exists at both source code and object code levels. You may recompile the program source with the AIX Version 1.2.1 compiler or copy a fully linked binary file to the system. The program runs either way but will not process multibyte characters.

AIX Version 1.2.1 systems may coexist on the same network with previous AIX systems (but not on the same cluster). Files may be transferred freely among these machines, provided the above caveats about non-MBCS programs being fed MBCS data are observed. In particular, files containing multibyte character data may be passed back and forth freely. The data must not be subjected to string processing by non-MBCS programs, however. There are a variety of file transfer programs available on the system. They vary in the support they provide for Japanese characters. Any of them provide facilities for a user to send a Japanese language text file to another user on a remote machine. Some of them may produce unreadable results at the receiver's end and should be avoided for Japanese text. Others will yield a fully readable file, even if that file must pass through one or more standard ASCII-only systems on the way.

This compatibility means that the reader may assume that a command or function works exactly the same way in Japanese characters as it would in ASCII characters. If this is not the case, it is specifically noted in the documentation.

The entire AIX document set is written from the viewpoint of a user working on a standard English language AIX system. This should not give the Japanese user any difficulty. AIX functionally is the same across the entire family. Wherever any change in function occurs, the difference is noted in the appropriate document appearing under the banner: *Japanese Language Support Information*.

¦ Copyright IBM Corp. 1985, 1991

1.1.10.2 Interoperability with Other Systems

An AIX system offers a great deal of flexibility. Many users can use the system at one time, each of them operating in a different locale and with a different character set. AIX Version 1.2.1 must also be able to interoperate with other systems. These others may be other versions of AIX, other versions of UNIX, or other IBM operating systems like VM/CMS. In any of these situations, a user should be able to move freely from one machine to another, send files and receive files. The programs involved with such operations are **telnet**, the Berkeley **r** commands, **mail**, **ftp**, **BNU**, **NFS**, and other network access facilities. All of these programs need to be somewhat adaptable in terms of the character set they can handle, because users may interact with them in any of the system's supported scripts.

There are two major reasons that interoperability may be problematic. The first is 7-bit vs. 8-bit support. Some operating systems make use of only seven bits in each byte for character encoding. Others use all eight. An 8-bit system can always read the files from a 7-bit system, but the reverse is not true. This topic is explored more fully in Chapter 5 and Chapter 7. The second area where MBCS support may hamper interoperability is the necessity of transmitting multibyte characters through several layers of software before they are read. A user operating in a Japanese locale is creating multibyte characters. File transfer operations must handle these files in a specific fashion:

All eight bits must be transmitted

The high-order bit must not be altered

The interpretation of the byte stream as characters must be delaye until the stream reaches software capable of handling multibyte characters. None of the intervening layers may impose meanings onto bit patterns in the stream or make assumptions about characters.

All systems involved in the interoperations must interpret th characters being read in the same way. For instance, a user creating files in Shift-JIS format cannot communicate freely to one using U-JIS. They must agree upon which file code to use.

Specific cautions apply to the network programs mentioned above. See the specific MBCS data contained in AIX Operating System documentation for further information.

¦ Copyright IBM Corp. 1985, 1991

1.2 Chapter 2. Routine System Management

Subtopics
1.2.1 Contents
1.2.2 About This Chapter
1.2.3 Starting the System
1.2.4 Stopping the System
1.2.5 Managing User Accounts
1.2.6 Tailoring the User Environment
1.2.7 Information about File Systems - The /etc/filesystems File
1.2.8 Creating and Mounting File Systems
1.2.9 Backing up Files and File Systems on AIX PS/2
1.2.10 Understanding System Security

¦ Copyright IBM Corp. 1985, 1991

1.2.1 Contents

¦ Copyright IBM Corp. 1985, 1991

1.2.2 About This Chapter

This chapter explains the routine tasks associated with managing an AIX Operating System. You may do some of the tasks described in this chapter daily, for example, starting up and shutting down the system, mounting and unmounting file systems, and backing up file systems. Other tasks you probably will do less often--setting up user accounts and creating new file systems. However, understanding the information in this chapter will help you use the AIX Operating System effectively and keep it running smoothly.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System,* and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

1.2.3 Starting the System

The following paragraph describes the start up procedure for a standalone PS/2 or PS/55. If your PS/2 is attached to a cluster as a cluster host, the start up procedure is different.

To start the AIX Operating System under normal circumstances, turn on the power switch for each system component. If there is a system diskette in drive 0 (A), the system attempts to initialize itself from that diskette. If there is no diskette in the drive, the system attempts to initialize itself from the fixed disk.

Note: If there is a non-boot diskette in drive 0, the system will not boot up, nor will it give any warning message.

Under normal circumstances, the system should initialize from the fixed disk. After the system successfully initializes from the fixed disk, it displays the copyright notice and the **login** prompt. At that time, the system is ready for normal use.

For certain system management tasks, you occasionally need to run a special version of your system: the ***maintenance system***.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.2.3.1 System Initialization

After loading the kernel into memory, the system goes through a period of internal initialization, during which it runs internal checks, initializes all memory, initializes some devices, and analyzes the standard file systems. After completing its internal initialization, the kernel starts the **init** (system initialization) process. All other processes on the system (except the scheduler) can be traced back to **init**. Most important of the processes that **init** creates are the shell and the processes that allow users to gain access to the system; that is, **init** runs the **getty** command for each ***port*** that **inittab** defined on the system. Each port can support an attached display station. (For more information about the initialization processes, see **init** and **getty** in *AIX Operating System Commands Reference*.)

If the kernel is loaded from the maintenance diskette, **init** initializes the maintenance system. The maintenance system consists of only a shell with superuser authority, the **init** process, and the scheduler. (For more information about the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.) To go from the maintenance system to normal operation, reboot, and remove the diskette when the screen flashes.

When the system is initialized for normal operation, **init** creates a shell process to run the commands as indicated by **/etc/inittab**. The commands in **/etc/inittab** prepare the system for users to log in. See the **inittab** entry in the "File Formats" section of the *AIX Operating System Technical Reference, Volume 2*, for information on the syntax of **/etc/inittab**. The **init** command automatically runs **fsck** on the root and **<LOCAL>** file systems.

To alter the way the system starts up, edit the **/etc/inittab** or the files in **/etc/initdir**. Generally, **/etc/inittab** contains three types of commands:

Housekeeping
: Startup files make certain that the system is in proper running condition, regardless of its condition when it was last stopped. The **fsck** command, which should be early in the startup sequence, checks the consistency of file systems and, as far as possible, corrects file system problems. If **fsck** cannot automatically repair the file system damage it detects, it causes the shell to stop the system. You cannot start the system until you repair the file system damage with **fsck** from single-user mode or by running the maintenance system as described under "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2. Other housekeeping commands remove files from temporary directories, initialize queues, save logged data, restore certain files to their original (default) states, and set up the ***ports*** that give display stations access to the system.

Mounting
: The **setmnt** command in the startup files restores the mount table to its default state. Then the **/etc/mount all** command mounts all of the file systems described in **/etc/filesystems**.

Starting daemons
: ***Daemons*** are processes that should start when the system starts and run until the system stops. The two most important daemons started by the startup files are **cron** and **qdaemon**.

¦ Copyright IBM Corp. 1985, 1991

The **cron** daemon runs the **sync** system call at 30-second intervals, completing any unfinished disk I/O. Thus, if the system fails, data on the disks cannot be more than 30 seconds out of date. You can also use the **cron** daemon to start processes at preset times. For example, **cron** can start the processing of system accounting data late at night.

The processes **cron** runs are specified in the files in the **/usr/spool/cron/crontabs** directory. The names of the files in this directory correspond to user IDs, such as root or adm. Any user can use the **crontab** command to create a file in this directory and have **cron** run commands that the user specifies by the file name.

The **qdaemon** provides queued access to certain system resources, such as printers. For more information on queues, see "Using the Queueing System" in topic 1.3.5.

Once it successfully completes the initialization process, **init** starts the necessary ***getty*** processes to allow display stations to access the system. After creating the initial login prompts, the only function of **init** is to create logins or set up new ports as required.

¦ Copyright IBM Corp. 1985, 1991

1.2.3.2 Running the PS/2 and System/370 Maintenance System

The AIX maintenance system provides you with a way to perform certain system management tasks without starting the system in the normal manner. You can use the maintenance system for one of two reasons:

The system will not initialize. The maintenance system may allow yo to correct the problem that prevents your system from operating normally.

You need to perform system management tasks that would be complicate by other processes running on the system. The maintenance system is a very limited form of the operating system, consisting of just the parts of the system required to perform certain system management tasks.

The maintenance system is most useful for backing up and repairing file systems or for making any change to the system that might be complicated if other processes are running. For example, during normal startup, the operating system tries to correct any file system damage. However, some types of damage cannot be corrected automatically. In such cases, the operating system displays a message indi- cating that the operator must help correct the damage. To make the repairs, start the maintenance system and do one of the following:

Use the **check a file system** maintenance command (a version of the **fsck** command described under "Checking and Repairing File Systems - The fsck Command" in topic 1.3.3.3).

Start the **standalone shell** and run **fsck** on the damaged file system.

Note: If you notice file system damage immediately, it is usually simple to repair. However, if the system runs on a damaged file system, the damage can spread, in some cases to the point that the entire file system is unusable.

You should run **fsck** only on unmounted file systems (idle file systems are also acceptable for most types of damage), and you should not need ***daemon*** processes while you use the maintenance system (a ***daemon*** process is a process that runs continually in the background, such as the process that manages printer queues). Therefore, when you start the maintenance system, none of the usual file systems are mounted and none of the usual daemon processes are started.

The remainder of this section explains how to start and use the maintenance system with the AIX Operating System for the PS/2 and System/370.

¦ Copyright IBM Corp. 1985, 1991

1.2.3.3 The PS/2 Maintenance System

Before you begin: for the following procedures you need the following items:

AIX 1.2.1 Operating System Boot Diskettes #1 and #2

A Write-enabled copy of the AIX 1.2.1 Operating System Maintenance Diskette

If your system is *not* powered on:

1. Insert the AIX Operating System PS/2 Boot Diskette #1 into diskette drive 0 (or A).
2. Turn on the power to all components as usual.
3. At the Boot Diskette Menu, highlight the first selection, "Boot from Diskette" and press **ENTER**.
4. At the prompt, "Module to be loaded:" press **ENTER** for the default kernel, **unix.gen**.
5. At the prompt, "System Mode:" press **ENTER** for the default mode of "single user."
6. At the prompt, "Run system from hard disk:" press **ENTER** for the default of "no."
7. Wait for information to load from your disk.
8. Insert Boot Diskette #2 at the prompt, "Please insert BOOT diskette number 2; Press any key when ready."
9. At this point the AIX PS/2 Installation Screen appears with a choice of either installing or maintaining the system.
10. Insert your write-enabled AIX Operating System Maintenance diskette into disk drive 0 (or A) and press **ENTER**.
11. The AIX PS/2 Maintenance Utility screen will appear. Refer to the menu below to select a PS/2 maintenance utility option:

AIX PS/2 Maintenance Utility Menu

Show Minidisk Information
Add AIX Minidisk
Delete AIX Minidisk
Change Active Operating System
Make Filesystem on Minidisk
Check Filesystem on Minidisk
Delete AIX From a Fixed Disk
Start Standalone Shell

¦ Copyright IBM Corp. 1985, 1991

```
¦ Exit Utilities and Reboot
¦
¦ Please Select an Option
¦ Press <SPACE> to toggle items; <ENTER> to select; <F3> to exit.
¦
+-------------------------------------------------------------------------
```

For more information on running the PS/2 Maintenance Utility menu commands above, refer to "minidisks" in the *Installing and Customizing the Operating System for the PS/2* manual, and the **minidisks** command in the *AIX Commands Reference*.

If your system *is* powered on:

1. Ensure that all other users log off the system.
2. Enter the following command:

   ```
   shutdown
   ```

3. When **shutdown** completes, insert AIX Boot Diskette #1 into diskette drive 0 (or A) and press **ENTER**.
4. At the Boot Diskette Menu, highlight the first selection, "Boot from Diskette" and press **ENTER**.
5. At the prompt, "Module to be loaded:" press **ENTER** for the default kernel, **unix.gen** name.
6. At the prompt, "System Mode:" press **ENTER** for the default of "single user."
7. At the prompt, "Run system from hard disk:" press **ENTER** for the default of "no."
8. Wait for information to load from your disk.
9. Insert Boot Diskette #2 at the prompt, "Please insert BOOT diskette number 2; Press any key when ready."
10. At this point the AIX PS/2 Installation Screen appears with a choice of either installing or maintaining the system.
11. Insert your write-enabled AIX Operating System Maintenance diskette into disk drive 0 (or A) and press **ENTER**.
12. The AIX PS/2 Maintenance Utility menu screen will appear. Refer to the AIX PS/2 Maintenance Utility Menu screen (above) to see the maintenance utility choices.

For more information on running the PS/2 Maintenance Utility menu commands above, refer to "minidisks" in the *Installing and Customizing the Operating System for the PS/2* manual, and the **minidisks** command in the *AIX Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.3.4 The System/370 Maintenance System

The procedure for starting and using the maintenance system on a System/370 is closely parallel to the procedure for a PS/2. The primary difference is that the maintenance system comes on tape, rather than diskette, and is started by the following steps.

To Start the Maintenance System on a System/370

1. Bring up CMS by entering:

   ```
   ipl cms
   ```

2. Start the maintenance system by entering:

   ```
   aixmaint
   ```

   The following menu appears:

   ```
                  **********START OF QUESTIONS**********
   Installation options

   1. Perform initial installation of a new site.
   2. Add this previously installed site to an existing TCF cluster.
   3. Perform maintenance on a previously installed system.

   Select installation option (default:  1)
   ```

3. Select option #3, and enter your password. The prompt **<generic<1>>#** should appear.

4. Enter **/etc/maint**.

Once initialized, the System/370 maintenance system displays the System Management menu:

```
                          SYSTEM MANAGEMENT

      ID    Item

       1    Install AIX

       2    Use Maintenance Commands

       3    Start the Standalone Shell

       4    End System Management
```

¦ Copyright IBM Corp. 1985, 1991

```
+-------------------------------------------------------------------------+
|       To SELECT an Item, type its ID and press Enter:  1                |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

Select item 2 to use the maintenance commands explained under "Using System/370 Maintenance Commands." Select item 3 to use the ***standalone shell*** explained under "Using the Standalone Shell" in topic 1.2.3.4.2. To stop the maintenance system, select item 4. Then, to initialize the system for normal operation, press **Ctrl-Alt-Pause**.

For more information about using the Install AIX and End System Management options of the maintenance system on a System/370, see *Installing and Customizing the AIX/370 Operating System for the System/370*.

Subtopics
1.2.3.4.1 Using System/370 Maintenance Commands
1.2.3.4.2 Using the Standalone Shell
1.2.3.4.3 Examples

¦ Copyright IBM Corp. 1985, 1991

1.2.3.4.1 Using System/370 Maintenance Commands

When you choose item 2 (**Use Maintenance Commands**) on the System/370 System Management menu, the maintenance system displays the following Use Maintenance Commands menu:

```
+-------------------------------------------------------------------------+
|                                                                         |
|                                                                         |
|                          USE MAINTENANCE COMMANDS                       |
|                                                                         |
|       ID    Item                                                        |
|                                                                         |
|        1    Show fixed disk minidisk information                        |
|        2    Change load status of a fixed disk minidisk                 |
|        3    Create a fixed disk minidisk                                |
|        4    Delete a fixed disk minidisk                                |
|        5    Check a file system                                         |
|        6    Make a file system                                          |
|        7    Restore commands                                            |
|        8    Backup commands                                             |
|        9    Delete all minidisks in a drive (DEBUG)                     |
|                                                                         |
|       To CANCEL and go back to the SYSTEM MANAGEMENT menu, press F3.    |
|                                                                         |
|       To SELECT an Item, type its ID and press Enter:                   |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

To select an item, enter its number. Each item leads you through a series of menus that allow you to accomplish a particular task.

For more information about the function of each maintenance command, see the following:

Item ID	Reference
1-4	Information on installing the AIX Operating System in the AIX Operating System installation manuals.
5	"Checking and Repairing File Systems - The fsck Command" in topic 1.3.3.3.
6	"Creating and Mounting File Systems" in topic 1.2.8.
8-9	"Backing up Files and File Systems on AIX PS/2" in topic 1.2.9.

Note: When you choose items 8 and 9 on the Use Maintenance Commands menu to back up a file system, the backups and restores are by inode only.

¦ Copyright IBM Corp. 1985, 1991

1.2.3.4.2 Using the Standalone Shell

When you choose item 3 (**Start the Standalone Shell**) on the System Management menu, the maintenance system displays the following screen:

```
+-------------------------------------------------------------------------+
|                                                                         |
|                                                                         |
|   /BIN/CP          COPIES FILES                                         |
|   /BIN/DD          CONVERTS AND COPIES FILES                            |
|   /BIN/ED          EDITS FILES                                          |
|   /ETC/FSCK        CHECKS AND REPAIRS FILE SYSTEMS                      |
|   /ETC/INIT        INITIALIZES SYSTEM                                   |
|   /BIN/LN          LINKS FILES                                          |
|   /BIN/LS          LISTS DIRECTORY CONTENTS                             |
|   /BIN/MKDIR       MAKES DIRECTORIES                                    |
|   /ETC/MKFS        MAKES FILE SYSTEMS                                   |
|   /ETC/MKNOD       CREATES SPECIAL FILES                                |
|   /ETC/MOUNT       MOUNTS FILE SYSTEMS                                  |
|   /BIN/MV          MOVES FILES                                          |
|   /BIN/OD          CONVERTS OCTAL TO DECIMAL                            |
|   /BIN/KILL        STOPS PROCESSES                                      |
|   /BIN/RM          REMOVES FILES                                        |
|   /BIN/SH          RUNS A SHELL                                         |
|   /BIN/SYNC        UPDATES STORAGE FROM BUFFERS                         |
|   /ETC/UMOUNT      UNMOUNTS FILE SYSTEMS                                |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

Notice that the standalone shell prompt is a pound sign (#) rather than the **$** of the standard shell.

The standalone shell is most useful for tasks that you cannot perform while the operating system is running, such as repairing a file system.

When you start the standalone shell, the **init** command runs automatically, much as it does when you start the system for normal operation. Once the standalone shell is started, the following commands are available:

chparm Changes system parameters

cp Copies files

dd Converts and copies files

ed Edits files

fsck Checks and repair file systems

init Initializes system

kill terminates processes

ln Links files

ls Lists directory contents

mkdir Makes directories

¦ Copyright IBM Corp. 1985, 1991

mkfs Makes file systems

mknod Creates special files

mount Mounts file systems

mv Moves files

od displays contents of binary files

rm Removes files

sh Runs a shell

sync Updates storage from buffers

umount Unmounts file systems.

Most of these commands are discussed in other sections of this book (see the index) or in *AIX Operating Systems Commands Reference*.

File systems used during normal mode operation may be mounted from the standard shell. The **/mnt** directory may be used for this.

Note: The device names in the following examples are specific to the PS/2. They may be used exactly as printed for any AIX PS/2 sites in the cluster. For AIX/370 sites, the device names are of the following form:

/dev/**disk**0000**n**

where **disk** indicates the disk type and **n** indicates the disk number; **/dev/chd00001** is **ckd** hard disk number 1 and **/dev/fhd00001** is fba hard disk number 1.

¦ Copyright IBM Corp. 1985, 1991

1.2.3.4.3 Examples

For example, the root directory of a normal system is created when the AIX Operating System is installed. On a PS/2 site, the **/dev/hd2** minidisk contains this file system (the replicated root file system). To access files on a normal replicated root file system from the standalone shell, use the command:

```
mount /dev/hd2 /mnt
```

Likewise, **/machinename** (**machinename** is the name of the host machine) is found on the **/dev/hd3** minidisk, and **/u** is found on the **/dev/hd1** minidisk. Thus, you could also use the command:

```
mount /dev/hd3 /mnt/machinename
```

The commands of the standalone shell may be extended by mounting **/dev/hd3** and **/dev/hd2** and running commands that exist on these file systems.

1. To examine data or to run normal mode commands, enter:

   ```
   mount /dev/hd2 /mnt
   mount /dev/hd3 /mnt/machinename
   mount /dev/hd1 /mnt/u
   PATH=/mnt/bin:/mnt/usr/bin:$PATH
   cd /mnt
   ```

2. To check the replicated root file system, enter:

   ```
   fsck /dev/hd2
   ```

3. To check the **<LOCAL>** file system, enter:

   ```
   fsck /dev/hd3
   ```

4. To make backups by name, enter:

   ```
   mount /dev/hd2 /mnt
   mount /dev/hd3 /mnt/machinename
   mount /dev/hd1 /mnt/u
   backup -i
   cd /mnt
   ```

   (Enter the file names relative to the current working directory, that is, **etc/system**, **bin/cp**.)

To end the standalone shell, press **END OF FILE**.

¦ Copyright IBM Corp. 1985, 1991

1.2.4 Stopping the System

The AIX Operating System can run multiple processes, including background processes, at the same time. To prevent damage to the file system, you should bring all processes to an orderly stop before you turn off the power to the computer. The **shutdown** command is the usual way to bring the system to an orderly stop.

The default value of a system shutdown is **shutdown -h**, which brings the system to a complete halt. During a default shutdown, all users are notified of impending shutdown via the **wall** command. The **phold** command then prevents any new logins.

The time for the shutdown may be specified in several ways:

As an absolute value, for example, **2:13 PM**

As a relative value, for example: **+30 minutes**

As the default value, which is **+1 minute**.

When the specified shutdown time arrives, the **shutdown** command runs the **halt** command to end any remaining processes and runs the **sync** command to flush all memory resident disk blocks. Finally, it unmounts the file systems and prints the message:

```
Normal System Shutdown
System halted, you may turn the power off
now.  Type Enter to reboot.
```

At this point you can safely turn off the power switch to fully disable the system, or you can reboot the system as indicated.

Note: If you have a single user system, you may wish to do a **shutdown -f** to eliminate the unnecessary **wall** command and the delay.

If you want to bring the system down for maintenance, you should use **shutdown -m** to disable all user terminals and associated functions and leave just enough of the system running to perform service procedures. For more information on service procedures, see *AIX/370 Planning Guide*.

If there are no users on the system, you can use the **reboot** command to reboot the system.

For more information about shutdown procedures, see "Disk Buffering - The sync Command" in topic 1.3.3.1.1 and **reboot** and **shutdown** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.5 Managing User Accounts

Giving users access to the system is an important system management task. Even if you are the only user on the system, you may need to create accounts, remove accounts, and change account information occasionally. You can perform these account management tasks with the **adduser** command described under "Creating, Changing, and Removing Accounts - The adduser Command" in topic 1.2.5.1 (below). To manage the accounts on your system most effectively, you also should be familiar with the information under "Types of User Accounts" in topic 1.2.5.2, "Using Different Login Names" in topic 1.2.5.3, and "User Account Files" in topic 1.2.5.4.

Notes:

1. The preferred method of adding users is to use the **adduser** command. However, AIX supplies another command, **/etc/users**, which is equivalent to the **adduser** command. To use the **/etc/users** command, you need to type the entire pathname. If you don't type the entire pathname, AIX defaults to **/usr/ucb/users**, which is a condensed form of the **who** command.

2. Do not alter **/etc/passwd** or **/etc/group** with an editor or through any procedure other than the **adduser** command.

3. Since the **/etc/passwd** file is automatically replicated, you only need to run the **adduser** command on one cluster site to add a user to all sites in the cluster.

4. Change the group information before changing the user information to assure the user has the correct group parameters.

Subtopics
1.2.5.1 Creating, Changing, and Removing Accounts - The adduser Command
1.2.5.2 Types of User Accounts
1.2.5.3 Using Different Login Names
1.2.5.4 User Account Files

¦ Copyright IBM Corp. 1985, 1991

1.2.5.1 Creating, Changing, and Removing Accounts - The adduser Command

The **adduser** command is a tool for managing user accounts. The **adduser** command provides the following subcommands:

add Creates new user accounts and new groups.

change Changes account and group information, such as user name, password, and group ID.

delete Removes accounts or groups from the system.

help Displays the **users** subcommands.

invalidate Makes a user password invalid.

quit Puts changes into effect and ends **users**.

show Displays user or group account information.

To run the **adduser** command, you must be a member of the system group or have superuser authority.

For more information on modifying files within the **adduser** command, see "Setting Environment Variables in the Shell" in topic 1.2.6.3, and the **adduser** command in *AIX Operating System Commands Reference*.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.2.5.1.1 Starting and Stopping the adduser Command

To Start the adduser Command

Enter **adduser**.

After you start **adduser**, you can use any of its subcommands to perform your account management tasks. Enter **?** (question mark) to display the list of subcommands:

```
  Available commands are:

  a[dd]         user or group
  c[hange]      user or group
  d[elete]      user or group
  h[elp]        print this message
  i[nvalidate]  a user
  q[uit]        finalize changes and exit
  s[how]        user or group

  <INTERRUPT> will exit without changing any of the files.
  >
```

Note: Be sure to use the appropriate **INTERRUPT** key for your own keyboard.

To select a subcommand, type its initial letter (for example, **a** for **add**) and press **Enter**. If the subcommand requires you to specify **user** or **group**, use the abbreviation **u** or **g** (for example, **a u tom**).

To show the information about a user, enter: **show user username** (or **s u username**). To show the information about a group, enter: **show group groupname** (or **s g groupname**).

To display help information, enter: **h**.

To Stop the adduser Command

To stop the **adduser** command without making any changes, press **INTERRUPT**.

OR

To put changes you have made into effect and stop **adduser**, enter **q**.

The following sections explain how to use the remaining **adduser** subcommands (**add**, **change**, **delete**, and **invalidate**).

¦ Copyright IBM Corp. 1985, 1991

1.2.5.1.2 Using the a[dd] Subcommand

To Add a User

1. Enter: **adduser**. (If **adduser** is already started, go to step 2.)
2. After the **>** prompt, enter: **a u username**. (Use lowercase letters for the names of users you add to the system.)
3. To add user information, enter: **n** after the **OK? (y)** prompt.
4. To add the user's full name:
   a. Enter: **fu** after the **Field?** prompt.
   b. Enter the user's full name (for example, john doe).
5. To add a password:
   a. Enter: **pa** after the **Field?** prompt.
   b. Enter the password.
   c. Re-enter the password.
6. To add other information:
   a. After the **Field?** prompt, enter enough of the field name to identify it.
   b. Enter the information.
7. When your changes are complete, press **Enter** after the **Field?** prompt. The **adduser** command displays the new information (passwords are encrypted).
8. If the changes are correct, press **Enter** after the **OK? (y)** prompt to add the new user to the system. (If the changes are not correct, enter **n** and then make the necessary corrections).
9. Press **Enter** after the prompt **Standard new user initialization? (y)**.
10. Enter another subcommand or stop **adduser**.

Note: You are not allowed to change the information in the **opt. groups** field from the **adduser** subcommand. To add a user to a group, use the **change group** subcommand.

You can set the values for other items with the same procedure used to set **Fullname** and **Password**. For more information about user accounts, see "Types of User Accounts" in topic 1.2.5.2.

For more information on modifying files within the **adduser** command, see "Setting Environment Variables in the Shell" in topic 1.2.6.3, and the **adduser** command in *AIX Operating System Commands Reference*.

To Add a Group

1. Enter: **adduser**. (If **adduser** is already started, go to step 2.)

Copyright IBM Corp. 1985, 1991

2. After the **>** prompt, enter **a g groupname**.

3. To accept the displayed information, press **Enter** after the **OK? (y)** prompt. (To add group information, enter **n** after the **OK? (y)** prompt.)

4. To add a group password:

   a. Enter **pa** after the **Field?** prompt.
   b. Enter the group password.
   c. Re-enter the group password.

5. To add members to a group:

   a. Enter **m** after the **Field?** prompt.
   b. Enter **a**.
   c. Enter the **username** of the user you are adding.

   Note: Be sure to type the name correctly because there is no check to verify if this name is defined.

6. To end this procedure, press **Enter** after the **Field?** prompt.

7. If the changes are correct, press **Enter** after the **OK? (y)** prompt to add the group to the system.

8. Enter another subcommand or stop **adduser**.

For more information about groups and the fields used by the **add group** command, see "The Group File" in topic 1.2.5.4.2.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.1.3 Using the c[hange] Subcommand

To Change User Information

1. Enter: **adduser**. (If **adduser** is already started, go to step 2.)
2. After the **>** prompt, enter **c u username**.
3. To make changes to the displayed information, enter **n** after the **OK? (y)** prompt.
4. Enter the name of the field you want to change after the **Field?** prompt.
5. Enter the information for the field you specified.
6. Use the same method to change the information in other fields.
7. When your changes are complete, press **Enter** after the **Field?** prompt.
8. If the changes are correct, press **Enter** after the **OK? (y)** prompt. The **adduser** command displays the **[UPDATED]** message.
9. Enter another subcommand or stop **adduser**.

For more information about the fields for which you can change values, see "The /etc/passwd File" in topic 1.2.5.4.1.

Note: You cannot change the information in the **opt. groups** field from the **c u** subcommand. To add a user to a group, use the **change group** subcommand.

To Change Group Information

1. Enter: **adduser**. (If **adduser** is already started, go to step 2.)
2. After the **>** prompt, enter **c g groupname**.
3. To make changes to the displayed information, enter **n** after the **OK? (y)** prompt.
4. Enter the name of the field you want to change after the **Field?** prompt. (For example, to change the members shown as part of the group, enter **m** or **member**.)
5. Enter **a** or **d**.
6. Enter the user name you wish to add or delete. Be sure to type the name correctly because there is no check to verify if this name is defined.
7. When your changes are complete, press **Enter** after the **Field?** prompt.
8. If the changes are correct, press **Enter** after the **OK? (y)** prompt.

¦ Copyright IBM Corp. 1985, 1991

The **adduser** command displays the **[UPDATED]** message:

9. Enter another subcommand or stop **adduser**.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.1.4 Using the d[elete] Subcommand

To Delete a User

1. Determine if you want to save the files in the user's directory. If you do, change the name of the directory, change the owner of the directory, or follow the instructions in step 3 for saving files. If you do not want to save the files, remove all files (including the hidden files such as the **.profile** file) from the directory of the user that is to be deleted.

2. Enter: **adduser**. (If **adduser** is already started, go to the next step.)

3. After the **>** prompt, enter **d u username**. The **adduser** command displays the prompt **Should the login directory (/u/username) be removed? (y)**. If you want to save the files in that directory, type **n** and press **Enter**.

   If you do not want to save the files, press **Enter**., and you should see the **[DELETED]** message. (If you have not removed all files from the user's login directory, **adduser** displays the message, **The user cannot be deleted because the user's login directory is not empty.**)

4. To verify that the user is deleted, enter **s u username**. The **adduser** command should display a message indicating that there is no such user.

5. Enter another subcommand or stop **adduser**.

Notes:

1. Be sure to change the ownership of (or delete) all files owned by the user who is being deleted, including files which may reside in directories other than the **/u/username** directory. Failure to do so may cause problems later. The owner of a file is identified by the user number (UID). Later, when a new user is added to the system, the same number may be used by the system again. The new user can then own any of the files which were not removed or for which ownership was not changed. For information about changing the ownership of a file, see the **chown** command in *AIX Operating System Commands Reference*.

2. When you delete a user, you have the choice of removing all of the files in that user's login directory or answering **no** to the prompt, **Should the user's login directory be removed?** If you answer **no** to the prompt, the entry for **username** is removed from the **/etc/passwd** file, but the **/u/username** directory will not be removed. You would not have to copy the files you want to save to another directory, but you would need to change the ownership of the files in the **/u/username** directory and of any other files owned by that user.

For more information on modifying files within the **adduser** command, see "Setting Environment Variables in the Shell" in topic 1.2.6.3, and the **adduser** command in *AIX Operating System Commands Reference*.

To Delete a Group

¦ Copyright IBM Corp. 1985, 1991

1. Enter: **adduser**. (If **adduser** is already started, go to the next step.)

2. After the **>** prompt, enter **s g groupname**. Write down the names of all the users listed under **Members**.

3. After the **>** prompt, enter **s u username** for one of the members of the group. Check to see if the group you want to delete is listed under **Group**. If it is, use the **c u *username*** subcommand to change the **Group** field. Repeat this step for each user you listed in the previous step.

   Note: See the example below for an alternate method for determining which users use the group you want to delete as their primary group. If there is a large number of members in the group, you may prefer the alternate method.

4. After the **>** prompt, enter **d g groupname**. The **adduser** command displays the **[DELETED]** message.

5. To verify that the group is deleted, enter **s g groupname**. The **adduser** command should display a message indicating that there is no such group.

6. Enter another subcommand or stop **adduser**.

Note: Before deleting a group, change the group field for all users using that group as their primary group. The next group added is assigned that group id. Users may get incorrect group assignments.

You can use an alternate method to delete a group. In the following example, the group is named **group1**. Use information in the **/etc/group** and **/etc/passwd** files to determine which users have **group1** as their primary group.

Enter **pg /etc/group** to display the group file. For more information about the **/etc/group** files see "The Group File" in topic 1.2.5.4.2. The following example shows **group1** with a group number (GID) of 200:

```
system::0:root,su
staff::1:user2,user3,user4,user5
bin::2:root,su,bin
sys::3:root,su,bin,sys
adm::4:root,su,bin,adm
mail::6:root,su
usr::100:guest
group1::200:user3,user4,user1
```

Note that the group number for **group1** is 200. You can also get this information by using the **s g group1** subcommand after starting **adduser**. The group number is shown in the GID field.

Enter **pg /etc/passwd** to display the password file. For more information about the **/etc/passwd** file, see "The /etc/passwd File" in topic 1.2.5.4.1. The following example shows **group1** with a group number (GID) of 200 for the primary group:

¦ Copyright IBM Corp. 1985, 1991

```
root::0:0::/:
su:*:0:0::/:
daemon:*:1:1::/etc:
netmail:*:1:1::/usr/spool/qftp:/usr/lib/INnet/waxsrvr
bin:*:2:2::/bin:
sys:*:3:3::/usr/sys:
adm:*:4:4::/usr/adm:
uucp:*:5:1::/usr/spool/uucppublic:/usr/lib/uucp/uucico
adduser:*:0:0::/usr/adm:/etc/adduser
guest:*:100:100::/usr/guest:
name::201:0::/u/name:
user1:utGzXG(TBLEso:200:200::/u/user1:/bin/sh
user2:6uxK1vlyz4eGc:202:1::/u/user2:/bin/sh
user3:Iu44UqcUnatjg:202:1::/u/user2:/bin/sh
user4:Yu101nqNFPurY:202:1::/u/user2:/bin/sh
user5:kumNeVh8GqHT2:202:1::/u/user2:/bin/sh
```

In the above example, notice that **user1** is the only entry with 200 as the group number field. It is the only entry in the above example with **group1** as the primary group. Using the information from the **/etc/group** and the **/etc/passwd** files, you can change the primary group for **user1** by using the **adduser** command as described below:

1. Enter: **adduser**.

2. After the **>** prompt, enter **c u user1**.

3. To change the displayed information, enter **n** after the **OK? (y)** prompt.

4. To change the primary group:

   a. Enter: **gr** after the **Field** prompt.
   b. Enter the name for the new primary group.

5. Press **Enter** after the **Field?** prompt.

6. If the changes are correct, press **Enter** after the **OK?** prompt. The **adduser** command displays the **[UPDATED]** message:

7. After the **>** prompt, enter **d g group1** to delete the group. The **adduser** command displays the **[DELETED]** message:

8. To verify that the group is deleted, enter **s g** group1. The **adduser** command should display a message indicating that there is no such group.

9. Enter **q** to stop **adduser**.

Note: When deleting a group, check to see if any users have it as their primary group. If they do, change the primary group for those users. Failure to do so may cause problems later. The primary group for a user is identified by the group number (GID). Later, when a new group is added to the system, it may be assigned the same GID that was used by the deleted group. Users whose primary group was not changed would then have the new group as their primary group.

Files also need to be checked. It may not be practical to change the group ownership of all files created by a deleted group.

¦ Copyright IBM Corp. 1985, 1991

However, there may be particular files which you may want to reassign. Use the **chgrp** command to change the group ownership of a file. For more information about the **chgrp** command, see *AIX Operating System Commands Reference*.

Because of the possible problems involved in using the **delete group** subcommand, use it sparingly. Give careful consideration to the consequences before deleting a group.

Note: You must enter all information using the **adduser** command in ASCII characters.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.1.5 Invalidating and Reinstating Users

You can prevent a user from logging in to the system by invalidating the user. You do not have to delete a user's files to invalidate the user.

To Invalidate a User

1. Enter: **adduser**. (If **adduser** is already started, go to step 2.)

2. After the **>** prompt, enter **i username**. The **adduser** command displays the **Invalidating username** message.

   Note: Do not specify **u** with the **i** subcommand.

3. Enter another subcommand or stop **adduser**.

To reinstate the user, use the **c**hange subcommand to clear the user's program field to set the user's shell to the default, **/bin/sh**. To set the user's shell to another program, such as **/bin/csh**, enter that program in this field.

Note: When you invalidate a user, the system does not save the user's preferred shell. You must enter the shell manually when you reinstate the user.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.2 Types of User Accounts

Each entry in the **/etc/passwd** file identifies an account. ("The /etc/passwd File" in topic 1.2.5.4.1 describes the **/etc/passwd** file.)

There are two types of accounts on the AIX Operating System:

The account with superuser authorit

User accounts

If you know the user name and password associated with an account, you can log in as that user. Thus, you can have one or more user accounts for your regular work and also, when necessary, use the account with superuser authority for system management work.

Subtopics
1.2.5.2.1 root - The Account with Superuser Authority
1.2.5.2.2 User Accounts

¦ Copyright IBM Corp. 1985, 1991

1.2.5.2.1 root - The Account with Superuser Authority

The user name ***root*** identifies the one user who operates without any restrictions--the user who has ***superuser authority***. (Another name for root is ***su***, for superuser.)

When you have superuser authority, you are immune from all permission checks in the system, you have **r**ead, **w**rite, and e**x**ecute permission for every file in the system, you can issue any system call, and you can cancel any process.

Because none of the usual protections apply when you have superuser authority, you should be extremely careful when you are logged in as root. Make it a practice never to log in as root when you can log in as a regular user and accomplish the same task. On a system that serves several different people, it is good practice to strictly limit who knows the root password and to change the password often in order to reduce the chance of accidental damage.

You can obtain superuser authority in three different ways:

After you log in with your normal user name, enter the **su** (switch user) command and supply the root password when prompted to do so. The shell prompt changes (usually from **$** to #) to remind you that you have superuser authority. To return to normal user status, press **END OF FILE**. This is usually the most convenient way to obtain and give up superuser authority.

Log in with the one of the user names **su** or **root**. Instead of the usual **$** (shell) prompt, the system displays the root prompt (usually #), to remind you that you have superuser authority. After you finish the work that requires superuser authority, either log out (press **END OF FILE** and the **$** prompt returns) or run the **shutdown** command to stop the system.

Start the maintenance system (se "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2). The maintenance system shell has superuser authority.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.2.2 User Accounts

All user accounts are subject to normal system checks and protections. To reduce the chance of accidental damage to the system, always work with a user account except when you must have superuser authority.

There are two types of user accounts: those created with the **adduser** program (described under "Creating, Changing, and Removing Accounts - The adduser Command" in topic 1.2.5.1) and those supplied with the system to perform certain system management tasks. Accounts created with the **adduser** program might be called ***ordinary user accounts***. These accounts are commonly identified with a particular person. They are used to run application programs (for example, the shell, text editors, or electronic spreadsheets) and store the data associated with them. As with any account, there is an entry in the **/etc/passwd** file for each ordinary user account. The home (or login) directories for ordinary user accounts are located in the **/u** directory (for example, **/u/jones**).

The second group of user accounts is supplied with the system and are known as **special user accounts**. They give users access to certain data and the ability to perform certain system management tasks which otherwise would require superuser authority.

Following is a list of these special user accounts:

bin Owns most of the directories, libraries, and programs or commands provided with the system. While most users have **x** (execute) permission for the files owned by **bin**, only **bin** or **root** (superuser) can modify those files.

In most respects, **bin** has ordinary user privileges. However, because **bin** owns the very important directories **/bin**, **/usr/bin**, **/lib**, and **/etc**, any mistakes you make while logged in as **bin** could affect the entire system.

adduser Is limited to one function: creating and maintaining user and group accounts. After logging in, the **adduser** command runs the account maintenance program.

For more information on modifying files within the **adduser** command, see "Setting Environment Variables in the Shell" in topic 1.2.6.3, and the **adduser** command in the *AIX Commands Reference*.

adm Owns the administrative and accounting files in **/usr/adm**. The **adm** user allows you to isolate billing from other system management tasks.

In addition to these special user accounts, there is also a special user group--the system group, or group **0**. Certain commands (in the directory **/etc**) which are not available to regular users can be run by any member of the system group.

A good reason for having a system group is to limit the number of users who need to know the root password. Members of the system group can do many system management jobs without obtaining superuser authority. However, because members of the system group do not have superuser authority, any mistakes they make are less likely to cause serious problems.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.3 Using Different Login Names

Once you are logged in to the system, the most convenient way to change the account name you are using is with the **su** (switch user) command. To obtain superuser authority with the **su** command, simply enter **su** and then enter the superuser password when the system prompts you for it:

```
su
password:
```

Notice that when you obtain superuser authority, the shell prompt changes from **$** to #. To return to your original user name, press **END OF FILE** (**Ctrl-D**).

You also can use **su** to change your user name to that of any other user on the system, provided you know the required password:

```
su username
password:
```

Notice that you must supply the **su** command with **username** except when you are obtaining superuser authority. To return to your original user name, press **END OF FILE** (**Ctrl-D**).

For more information about the **su** command, see **su** in *AIX Operating System Commands Reference*.

When you use a number of different accounts, and especially if you often change from one account to another with the **su** command, you may forget either your current user name or the user name you used when you logged in to the system. Use the **id** command to determine your current user name:

```
id
uid=0(root) gid=0(system)
```

Regardless of your current user name, you can use the **logname** command to determine the user name you used to log in to the system:

```
logname
jones
```

You can also use the **who am i** command to determine your user name:

```
who am i
jones      console       Apr 4  15:04
```

In addition to your user name, the **who am i** command also gives you the name of the display station you are using (**console** in this example) and the date and time that you logged in.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.4 User Account Files

To determine the authority associated with a user account, the system refers to two files:

/etc/passwd

/etc/group

The next two sections explain the information these files contain and how you can modify these files as necessary to maintain user accounts on your system.

Subtopics
1.2.5.4.1 The /etc/passwd File
1.2.5.4.2 The Group File

¦ Copyright IBM Corp. 1985, 1991

1.2.5.4.1 The /etc/passwd File

The **/etc/passwd** file contains all information that defines a user and contains one line for each user on the system. The content and name of this file must be in ASCII characters. Each line contains seven fields separated by colons (:), for example:

```
bud:Ym2CnRPo17Dyg:200:1:Bud Smith/3000;t;all:/u/bud:/bin/sh
--- ------------ --- - -------------------- ------ -------
 1        2        3  4          5             6       7
```

Figure 2-1. Fields in a Password File Entry

The fields, from left to right, are:

1. User name
2. Encrypted password
3. User number (**UID**)
4. Group number (**GID**)
5. Optional information field
6. Home (login) directory
7. Initial program (***login shell***).

Following are descriptions of the information in each field:

User name
The **user name** field contains the name (up to eight characters long) with which a particular user logs in to the system. The system always uses the name in this field to refer to that user.

Encrypted password
If this field is empty, the user does not have a password. (A user without a password does not receive a **password** prompt.) It is always good practice to use passwords, even on a single-user system.

If users forget their passwords, you can give them access to the system by being **superuser** and changing their password with the **password** command. (For information on how to change a password, see *Using the AIX Operating System*.) You can also use the **change** option of the **adduser** command to set a new password for the user.

A password field may also contain optional **aging** information, separated from the encrypted password by a comma. Aging information consists of encoded numbers that specify the maximum and minimum age (in weeks) for the password and the week during which the password was last changed. A user whose password expires is forced to set a new password before logging in to the system. Only a user with superuser authority can change a password before it is at least as old as the minimum age. If security is a major concern on your system, you can use the aging feature to force users to change their passwords frequently. To force a user to set a password before beginning

¦ Copyright IBM Corp. 1985, 1991

to work on the system, set the maximum and minimum times to zero and make sure there is not an indication of the time the password was last changed. The **adduser** command makes it easy to arrange for all new accounts to have standard aging parameters or to change the aging time limits for a selected user when you suspect that a password is no longer secure.

User number *and* **Group number**

The user number (**UID**) and group number (**GID**) define the identity of users for security purposes. All file protection on the AIX Operating System is based on three sets of permissions, those for **owner**, **group**, and **others**. Every process started by a user is identified with that user's user number and group number. Thus, the user number and group number together with permissions make it possible to control the files to which a particular user has access.

Optional information

The fifth field of the password file contains assorted information related to the user. It consists of four optional subfields in the form:

full_name/file_size;site_info;site_exec_perm

full_name is the real name of the person whose user name appears in the first field.

file_size specifies the maximum size (in 512-byte blocks) of files that the user is allowed to create. If you do not set the maximum file size, you can omit the separator (**/**), and a standard, system-wide value will apply to that user's files. (For more information on setting maximum file size, see **login** and **ulimit** (for **sh**) or **limit** (for **csh**) described in *AIX Operating System Commands Reference*.)

site_info can contain any printable character other than a colon (:) or semicolon (;). This subfield is available for any identifying character that may be required by a site-specific application. If you do not include the **siteinfo** field, you can omit the separator (**;**).

site_exec_perm can contain a site-group name which identifies the site or sites on which this user can log in or execute programs. For information about site-group names, see "Site Permission" in topic 1.2.10.4 and *AIX Operating System Technical Reference*.

Home (login) directory

The sixth field contains the name of the user's home (login) directory. The **login** program places the user in this directory at login. Typically, this directory is owned by the user and all the user's private files are kept in the directory tree below it. When you add a new user to the system, the **adduser** command automatically creates a login directory for the user.

Initial program (login shell)

The last field in the password file contains the name of the program that the user runs upon logging in (the login shell). Usually this is the **/bin/sh**, or shell program (described in *Using the AIX Operating System* and under **sh** in *AIX Operating*

¦ Copyright IBM Corp. 1985, 1991

System Commands Reference). If this field is blank, the system automatically starts the **/bin/sh** program for that user. You can specify another program in this field if, for example, the user requires a different command interpreter.

The following example is a sample of a typical password file. The entries for certain users (for example, **root** and **bin**) appear on all systems, but with different encrypted passwords. This sample password file includes users in different groups and users with special login programs:

```
root:8eCGXCGuI6HR2:0::/:
adduser:z1BclK147ta4Q:0:1::/usr/adm:/etc/adduser
daemon:mchxEqsYbP0MY:1:1::/:
bin:tbltZt2qfuk:2.:2:/3000:/bin:
adm:7dyLNacEX9oss:4:4::/usr/adm:
sync:4U8zktFt.GEj6:20:1::/:/bin/sync
chris:Wjpfqst3yXp0Q:201:1:Chris Cooper;tech:/u/chris:
heinz:mIk2beSZBxVq6:202:1:Heinz Hart;mgmt:/u/heinz:
ted:w.UVCyYgX28Es:203:1:Ted Black;tech:/u/ted:
jim:ckSsEP4GH/cwE:204:1:Jim Mori;tech:/u/jim:
bud:Ym2CnRPol7Dyg:200:1:Bud Smith/3000;tech:/u/bud:/bin/sh
```

It is good practice to reserve some number of user numbers (50 to 200) for use by programs that are created on your system and require special privileges. The remaining user numbers then are available for ordinary users.

¦ Copyright IBM Corp. 1985, 1991

1.2.5.4.2 The Group File

You can assign a user to one or more groups. Each group shares certain protection privileges. For example, you may want to place users in the same group because they work on the same project and need access to a common set of files. Or you may create a group (like the system group) to give certain users special privileges. Thus, you can use groups to increase or restrict the privileges of the users assigned to them.

When a user logs in, the system assigns the user to the group specified by the group number in that user's entry in **/etc/passwd**. The user can then use the **newgrp** command (described in *AIX Operating System Commands Reference*) to change the group associated with all processes the user creates. The group under which the user is currently working is called the primary group.

The AIX Operating System allows users to belong to several groups at the same time. A user's primary group is the one specified in the **/etc/passwd** file. However, you can use the **adduser** command (described under "Creating, Changing, and Removing Accounts - The adduser Command" in topic 1.2.5.1) to add users to other groups. (Also use **adduser** to create new groups.) Any files created by a user belong to that user's primary group, but the user has access to all files accessible to any group to which he belongs. To change the primary group, use the **newgrp** command (described under **newgrp** in *AIX Operating System Commands Reference*).

Use ASCII characters to ensure access by users of different locales.

The file **/etc/group** defines which users make up each group. Each line in **/etc/group** defines a group and consists of four fields separated by colons:

1. Group name

2. Encrypted password

3. Group number (GID)

4. Permission list (user names separated by commas).

The **group name** is a string of up to eight characters used to refer to the group. If there is a **password**, any user who attempts to enter the group is required to use it. The **group number** is simply the number assigned to that group. The **permission list** contains all users who have permission to enter the group, for example:

```
system::0:root,bin,jim,steve
staff::1:
managers::200:heinz,ted
```

The **adduser** command usually puts new users in the **staff** group. You can add users to groups and create newgroups as necessary. It may be useful to establish a group for new users; such a group, often called **other**, allows new users to work on the system before they are assigned to a particular group. As with user numbers, it is good practice to reserve some group numbers for uses unique to your system (perhaps 50 to 200 in all, with the first 10 to 20 reserved for use in system management tasks). The remaining group numbers can be assigned to users.

For users to change their primary group, they either must be on the list

¦ Copyright IBM Corp. 1985, 1991

of users permitted to enter that group or, if the group has a password, know the password for that group. The one exception to this rule is that a user always can return to the group specified in the password file (the primary group). That is, a user does not need to give a password or appear in the permission list in order to return to the user's primary group.

Note: Group passwords are rarely necessary. In practice, the permission list usually provides adequate group security, and most systems do not use group passwords.

¦ Copyright IBM Corp. 1985, 1991

1.2.6 Tailoring the User Environment

Several files and many of the programs that you run control the environment of the AIX Operating System. While the operating system provides prototypes for these files, you may need to change or add to the prototypes in order to adapt the system to your needs.

There are two shells available: the Bourne shell (**sh**) and the C shell (**csh**). For more information about either shell, see **sh** or **csh** in *AIX Operating System Commands Reference*.

The appropriate shell provides named variables and mechanisms for assigning string values to them, testing them, and substituting them into commands. In addition to their use as program variables for the shell as a high-level programming language, some of these variables control how the shell works. Furthermore, **exported** shell variables are passed in the process environment from the shell to the programs that it runs as user commands. Since exported variables can affect how any program runs, they provide potentially wide-ranging control over the user environment. Some of the commonly used environment variables are discussed under **sh** in *AIX Operating System Commands Reference*; others are discussed in the same book in conjunction with specific commands. This section discusses where the shell obtains exported variables other than those explicitly exported by the user.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.2.6.1 /etc/environment

Before any user logs in to the system, the **init** process (the main process involved in starting the system) reads the file **/etc/environment**. The **init** process passes variable assignments made in **/etc/environment** to each process it creates (its ***child*** processes). These variables automatically become part of the list of exported shell variables. To modify the **/etc/environment** settings, use a text editor to change the **/etc/environment** file. Use ASCII characters to ensure access by users of different locales.

Note: **/etc/environment** can contain only variable assignments, not shell commands.

Unless the values shipped with your system are correct, you should set at least the following variables in **/etc/environment**:

TZ This **TZ** (time zone) variable controls how the system translates your standard time (Universal Coordinated Time, also known as Greenwich Mean Time) into local time. Your time zone number, daylight savings time, standard time zone name, and daylight savings time zone name should have already been input during the installation process. For more information, see *AIX Operating System - Installing and Customizing the Operating System* (PS/2).

The value of TZ has the following form (spaces inserted for clarity):

std offset dst offset, rule

The expanded format is as follows:

stdoffset[dst[offset][,start[/time],end[/time]]]

Where:

std and *dst* Three or more bytes that are the designation for the standard (*std*) or daylight savings (*dst*) time zone. Only *std* is required; if *dst* is missing, then daylight savings time does not apply in this locale. Upper- and lowercase letters are explicitly allowed. For example, EST is the abbreviation for Eastern Standard Time and EDT for Eastern Daylight Time (U.S.). Any characters except a leading colon (:), digits, comma (,), minus (-), plus (+), and ASCII NUL are allowed.

offset Indicates the value one must add to the local time to arrive at Coordinated Universal Time. The *offset* has the form:

hh[:mm[:ss]]

The minutes (*mm*) and seconds (*ss*) are optional. The hour (*hh*) shall be required and may be a single digit. The *offset* following *std* shall be required. If no *offset* follows *dst*, summer time is assumed to be one hour ahead of standard time. One or more digits may be used; the value is always interpreted as a decimal number. The hour

¦ Copyright IBM Corp. 1985, 1991

shall be between zero and 24, and the minutes (and seconds) - if present - between zero and 59. Out of range values may cause unpredictable behavior. If preceded by a "-", the time zone shall be east of the Prime Meridian; otherwise it shall be west (which may be indicated by an optional preceding "+").

rule Indicates when to change to and back from summer time. The *rule* has the form:

date/time,date/time

where the first *date* describes when the change from standard to summer time occurs and the second *date* describes when the change back happens. Each *time* field describes when, in current local time, the change to the other time is made.

The format of *date* shall be one of the following:

J*n* The Julian day *n* (1= *n* = 365). Leap days shall not be counted. That is, in all years - including leap years - February 28 is day 59 and March 1 is day 60. It is impossible to explicitly refer to the occasional February 29.

n The zero-based Julian day (0 = *n* = 365). Leap days shall be counted, and it is possible to refer to February 29.

M*mn.d* The *d*th day (0 = *d* = 6) of week *n* of month *m* of the year (1 = *n* = 5, 1 = *m* = 12, where week 5 means "the last *d* day in month *m*" which may occur in either the fourth or the fifth week). Week 1 is the first week in which the *d*th day occurs. Day zero is Sunday.

The *time* has the same format as *offset* except that no leading sign ("-" or "+") shall be allowed. The default, if *time* is not given, shall be 02:00:00.

Your time zone information is set during the installation process (PS/2).

See Chapter 4, "Environment" and the *AIX Operating System Technical Reference, Volume 2*, for more information.

PATH Executable files (programs or commands) are located in various directories. The **PATH** variable contains a list of directory names (separated by colons) that the shell and other commands use when searching for an executable file. In the PATH assignment, a colon (:) that is not preceded by a directory name stands for the current directory. If the PATH variable is not otherwise set, its default assignment is:

```
PATH=:/bin:/usr/bin:/$HOME/bin:/etc
```

¦ Copyright IBM Corp. 1985, 1991

With this assignment, a command searches the directory **/bin** and the directory **/usr/bin**, and finally the **bin** directory in the user's home directory.

Two common reasons for setting the PATH variable to something other than its default are:

To increase the efficiency of searches. In some cases, a PATH that searches the current directory last is most efficient. The following assignment for **sh** causes the current directory to be searched last: **PATH=/bin:/usr/bin:**. The **csh** equivalent is:
set path=(/bin /usr/bin.)

To include other directories in the search. For example, if you have a directory named **/usr/local** containing commands that you have created, you can include it in the search path with the assignment **PATH=:/bin:/usr/local:**.

umask The file creation mask, **umask**, controls the **mode** (permissions) set for new files by specifying what permissions are not given. **umask** is set by the **umask** command in **/etc/profile**.

If no **umask** were set, files would be created with mode **777** (**r**ead, **w**rite, and e**x**ecute permission for **owner**, **group**, and **others**). Typical settings of the **umask** variable are:

umask=022 Files are created in mode **755**. **Group** and **others** do not have **w**rite permission.

umask=002 Files are created in mode **775**. **Others** do not have **w**rite permission.

umask=007 Files are created in mode **770**. **Others** do not have permissions; **owner** and **group** have all permissions.

For a general description of permissions, see *Using the AIX Operating System*.

filesize The **filesize** variable sets a default value for the system-wide maximum file size (in 512-byte blocks). The default file size limit is 8192 blocks or slightly more than four million bytes. You can override the default setting for individual users by setting a different value for **filesize** in the **/etc/passwd** file.

For more information on setting maximum file size, see **ulimit** described under **sh** and **login** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.6.2 Login

The **login** program adds the following values to the environment:

LOGNAME (user name - ASCII only)

TERM (display station or **terminal**). The default terminal type can be changed to your site in **/etc/ports**.

HOME (login directory).

For more information regarding **login**, see the *AIX Operating System Commands Reference*. For more information regarding TERM, see the **ports** command in the "File Formats" chapter of *AIX Operating System Technical Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.6.3 Setting Environment Variables in the Shell

A user's initial shell includes in its environment the variables passed to it by **login** and **init**.

The **sh** command executes:

The commands in **/etc/profile**

The file **.profile** in the user's login directory.

The **csh** command executes the files **.cshrc** and **.login** in the user's login directory.

Therefore, you should place those commands that all users should execute upon login in the appropriate shell file. The command, **news -n**, which draws attention to news items, is an example.

The user can override assignments to variables in **/etc/profile** with assignments in the user's "private" **.profile** file. Nevertheless, all commands in the **/etc/profile** files will be executed. Therefore, it is best to leave out non-essential commands. Generally, the **umask** command runs from **/etc/profile**. A ***mask*** determines the permissions that AIX sets initially for a file or directory. The **umask** command creates this mask. AIX applies the mask to the permission code **777** for directories and **666** for files. As a parameter, the **umask** command takes a string of three octal digits; the digits specify which permissions to remove. For example, the default **umask** command, which removes write permission for group and others, is:

```
umask 022
```

This mask causes the AIX Operating System to create directories with the permission code **755** and files with the permission code **644**.

You also can run **umask** from the command line or from your **$HOME/.profile**. In either case, you set the file creation mask for your own processes only, not for all system users.

For a general discussion of permissions, including octal representation, see *Using the AIX Operating System*.

Two other capabilities that can be controlled by variable settings and commands in **/etc/profile** and the user's **.cshrc** files:

User timeout — Sets the number of minutes a user can remain at command level without entering a command. After this interval of inactivity, the shell automatically exits and logs the user off. Timeouts are available only to users of **sh** and are controlled by the setting of the variable TIMEOUT.

The TIMEOUT variable is generally most useful on systems that serve a number of users or on systems where security is a major concern. On some systems, a terminal left unattended for a long time could give unauthorized people access to classified data.

Mail notification — Gives the shell the ability to check a user's mail drop file at intervals and display a message when new mail has been received. This check is performed and the message

¦ Copyright IBM Corp. 1985, 1991

displayed just before the shell would ordinarily print a prompt. The default value is for the shell to check the mail drop every 10 minutes.

The methods used to control mail notification differ between the shells:

For **sh**, add the following line to the **/etc/profile** file:

```
MAIL=$HOME/.newmail
```

For **csh**, instruct users to add the following line to their **.cshrc** file:

```
set mail=~/.newmail
```

If there are commands that you believe most users will want to run when they log in, but that are not appropriate for **/etc/profile**, you can create a default **.profile** (using **sh**) or a default **.cshrc** and **.login** file (using **csh**) to be added automatically to the home directories of new users. The **adduser** command and the autolog function makes it possible to set up default **.profile** or **.cshrc** and **.login** files through the **/usr/adm/newuser.usr** command file, which runs when a user is added to the system.

You can also change the **/usr/adm/newuser.sys** file (an initialization shell for adding users) to automatically copy the .cshrc and .login files to a new user.

The example below shows an actual **/usr/adm/newuser.sys** file, and the program that has been added to cause the .cshrc and .login files of user **Dinh** to be copied to a new user file:

```
# SCCSID(@(#)newuser.sy.t    1.3 changed 9/1/89 02:23:36)
#@(#) newuser.sys    7.1 - 87/06/15 - 18:48:09

echo "NEWUSER System initialization procedures
Password file entry:
  $1

"


#Getting the new user's home directory
dir='echo #1|awk -F':' '[print $6]''

echo "Creating a default.cshrc file from an existing user.cshrc"
cp/u/dinh/.cshrc $dir

echo "Creating a default .login file from an existing user.login"
cp /u/dinh/.login $dir
```

For more information see "Creating and Removing Accounts - the **adduser** Command," in *AIX Managing the AIX Operating System*, and the **adduser** command in the *AIX Operating System Commands Reference*

¦ Copyright IBM Corp. 1985, 1991

1.2.7 Information about File Systems - The /etc/filesystems File

Information about file systems is stored in the file **/etc/filesystems**. Most of the file system maintenance commands use **/etc/filesystems** to relate file system names to corresponding devices and to provide information about file systems to those commands.

The **/etc/filesystems** file is organized into ***stanzas*** that describe file systems. A stanza has the same name as the file system that it describes, and contains a series of **attribute** = **value** pairs that specify the characteristics of the file system. Stanzas named **default** contain information common to several file systems.

Use ASCII characters to ensure access by users of different locales.

Following is part of a sample **/etc/filesystems** file:

```
  *   This file describes all of the known file systems.  It
  *   is used by most of the file system maintenance routines
  *   to map file system names into the corresponding device
  *   names.  It also provides the information which tells fsck,
  *   df, mount and other programs what file systems to operate
  *   on by default.

  default:
          vol       =  "AIWS"
          mount     =  false
          check     =  false
          free      =  false
          backupdev =  /dev/rfd0
          backuplen =  1440

  /:
          dev       =  /dev/hd2
          vol       =  rcplrt
          mount     =  automatic
          check     =  false
          free      =  true
          gfs       =  1,1
          site      =  1
          ftype     =  repl,primary

  /aixps:
          dev       =  /dev/hd3
          vol       =  local
          mount     =  automatic
          check     =  false
          free      =  true
          gfs       =  2,1
          site      =  1
          ftype     =  nonrepl

  /mnt:
          dev       =  /dev/fd0
```

Stanzas can contain attributes in addition to the ones shown in the example. Following is a list of the possible **/etc/filesystems** attributes and a brief explanation of their meanings:

account Determines if the file system is to be processed by the

¦ Copyright IBM Corp. 1985, 1991

accounting system. The value can be either **true** or **false**.

backupdev Used by the **backup** and **restore** commands to determine the backup device associated with each file system. The value is usually the name of a diskette or magnetic tape special file. Use ASCII characters to ensure successful communication of **/etc/filesystem** and directory names across different locales.

backuplen Used by the **backup** command to determine the size of the default backup device associated with each file system.

backuplev Used by the **backup** command to determine the default backup level for each file system. For an explanation of backup levels, see "Backing up Files and File Systems on AIX PS/2" in topic 1.2.9.

boot Used by the **mkfs** command to initialize the boot block of a new file system. The value of **boot** specifies the name of the load module to be placed into the first block of the file system. Use ASCII characters to ensure successful communication of files.

check Used by the **fsck** command to determine the file systems to be checked. The value can be **true**, **false**, or some number that corresponds to a particular phase of the **fsck** program. For more information about **fsck**, see "Checking and Repairing File Systems - The fsck Command" in topic 1.3.3.3.

cyl Used by the **mkfs** command to initialize the free list and superblock of a new file system. The value is set to the number of blocks in one cylinder.

dev Identifies the block special file where the file system resides. System management programs use this attribute to relate file system names to the appropriate device names. Use ASCII characters to ensure successful communication of files.

free Used by the **df** command to determine which file systems are to have their free space displayed. This value is set either **true** or **false**.

gfs Used by file system commands to identify the global file system (GFS) number and the pack number of the file system. The two numbers are separated by a comma with the first number being the global file system number.

mount Used by the **mount** command to determine whether to mount the file system. If the value of **mount** is **true**, then the **mount all** command mounts the file system. The most convenient way to accomplish routine file system mounting is to place the **mount all** command in the startup files. The value of mount can be set **true**, **false**, and/or **readonly**. When the **readonly** value is set, the file system is mounted, but its contents cannot be changed. There is also a **removable** value, which should be set for file systems only.

The **mount** attribute for the root and **<LOCAL>** file systems have a special value: **automatic**. The **automatic** value causes the root file system to be mounted whenever the system is initialized and prevents the **mount all** command from attempting to mount the already-mounted file system.

¦ Copyright IBM Corp. 1985, 1991

site Used by file system commands to identify which cluster site mounts the specified file system. The value is the cluster site number of the site which mounts the specified file system.

size Used by the **mkfs** command to specify the size of a new file system. The value of **size** is some number of 1024-byte blocks. (The actual size will be rounded, so as to be a multiple of 4096 byte blocks.)

skip Used by the **mkfs** command to initialize the free list and superblock of a new file system. The value of **skip** is some number of 1024-byte blocks to be skipped between data blocks (the ***interleave factor***).

ftype Used by the file system commands to specify the type of the file system. Possible values include:

nfs - used as the type for an NFS mounted file system.
repl - used as the type for a replicated file system, which is accessible by all sites in the cluster. This specification must include one of the following qualifiers:
primary - used for the primary copy of the replicated root file system; this is the only copy that is readable and writable by all sites in the cluster.
backbone - used for the full copy of the primary site minidisk. This file system is writable only by the primary site.
secondary - used for the subset copy of the primary site minidisk. This file system is writable only by the primary site.
nonrepl - used for a non-replicated file system. This file system is local to the system you are on and not accessible by other sites in the cluster.

vol Used by the **mkfs** command when it initializes the label on a new file system. The value is a volume label with a maximum length of six characters.

¦ Copyright IBM Corp. 1985, 1991

1.2.8 Creating and Mounting File Systems

A file system is a complete directory structure confined to a single minidisk or diskette. Before you can use a minidisk or diskette to store data, create a file system on it. The usual way to create a file system on a PS/2 is with the **minidisks** command. For information about the **minidisks** command, see **minidisks** in *AIX Operating System Commands Reference* and the appropriate section in the *AIX Operating System Installing* and *Customizing the Operating system for the PS/2*.

However, if you need to recreate a file system on an existing minidisk, use the **mkfs** command.

The general form of the **mkfs** command is:

/etc/mkfs **name size**

where **name** is the name of the minidisk on which the file system is to be created and **size** is the total size of the file system in 1024-byte blocks. For more information about creating a file system, see *AIX/370 Planning Guide*. Allocate more space than you need because the inodes use up to 5% of the blocks. In order to create a new file system, the **mkfs** command must have the following information:

dev The name of the device on which the file system is to be created.

size The size of the new file system in 1024-byte blocks. For information about the limits for **size**, see *AIX/370 Planning Guide*.

vol A volume label.

cyl The number of 1024-byte blocks per cylinder on the disk. This parameter is used with **skip** to form the **interleave specification**. For an explanation of the interleave specification, see **mkfs** in *AIX Operating System Commands Reference*.

skip The number of 1024-byte blocks to be skipped between data blocks. Used with **cyl** to form the **interleave specification**.

boot The name of a file containing a **bootstrap** (initialization) program to be loaded into the boot block of the new file system.

Note: "Information about File Systems - The /etc/filesystems File" in topic 1.2.7 discusses these and other file system attributes more fully.

You can supply these parameters to **mkfs** in one of two ways:

From information contained in **/etc/filesystems** (ASCII data only).

1. Use a text editor to create a stanza for the new file system in **/etc/filesystems**. The most convenient way to create this stanza is to copy a similar stanza, give it the label you want to assign to the new file system, and then modify it as necessary. Save the modified **/etc/filesystems** file.

2. At the shell prompt (**$**), enter the command **/etc/mkfs filesystem_name**.

A stanza in **/etc/filesystems** contains, in one place, all of the

¦ Copyright IBM Corp. 1985, 1991

information that defines that file system. Thus, if you need to know something about how a file system is defined, it is very easy to get the information you need if the file system has a stanza in **/etc/filesystems**.

On the command line along with the **mkfs** command. For information on supplying parameters in this way, see **mkfs** in *AIX Operating System Commands Reference*.

Subtopics
1.2.8.1 Mounting and Unmounting File Systems
1.2.8.2 Creating and Mounting Diskette File Systems on AIX PS/2

¦ Copyright IBM Corp. 1985, 1991

1.2.8.1 Mounting and Unmounting File Systems

File systems are independent from each other and from the operating system. Each file system is associated with a different **device** (a minidisk or a diskette). Before you can use a file system, it must be connected to the existing directory structure (either to the replicated root file system or to another file system that is already connected). The **mount** command makes this connection. During the startup process, the system automatically mounts the file systems with the line **mount = true** in their **/etc/filesystems** stanza. However, you must use the **mount** command to mount any file system that:

Does not have a stanza in **/etc/filesystems**

or

Has the line **mount = false** in its **/etc/filesystems** stanza.

The **mount** command attaches one file system to another at a specified directory.

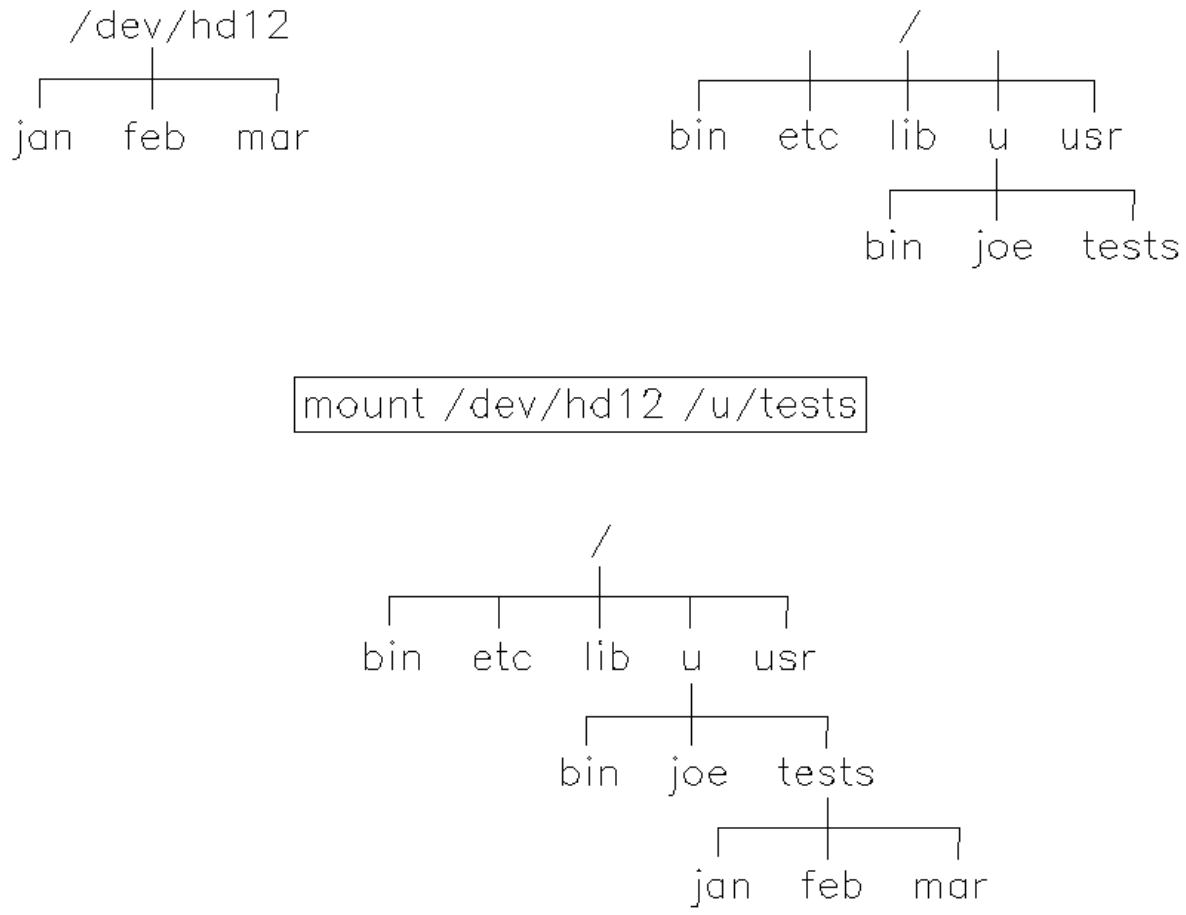


Figure 2-2. Mounting a File System

The general format for the **mount** command is:

mount **directory**

where **directory** is the name of the directory on which the file system is

¦ Copyright IBM Corp. 1985, 1991

to be mounted. For example, to mount a file system identified in **/etc/filesystems** as **/controls** (**mount = false**), enter:

```
mount /controls
```

If you have superuser authority, you also can specify the device that contains the file system to be mounted. The format for **mount** when you specify both the device and directory is:

```
mount device directory
```

For example, on a PS/2 to mount the file system on device **hd9** on the directory **controls**, use the command:

```
mount /dev/hd9 /controls
```

On a System/370, to mount the file system on device **disk9** on the directory **controls**, use the following command:

```
mount /dev/disk0009 /controls
```

If there is a stanza in **/etc/filesystems** for the directory to be mounted, simply specify the name of the stanza with the **mount** command. For example, on a PS/2, **mount /u** mounts the file system described by the following **/etc/filesystems** stanza onto the **/u** directory:

```
/u:
        dev       =  /dev/hd1
        vol       =  "/u"
        mount     =  true
        check     =  true
        free      =  true
```

On a System/370, **mount /u** mounts the file system described by the following **/etc/filesystems** stanza onto the **/u** directory:

```
/u:
        dev       =  /dev/disk0001
        vol       =  "/u"
        mount     =  true
        check     =  true
        free      =  true
```

Notice that the **mount** attribute in this stanza is set to **true**, which means that this file system is automatically mounted by the **mount all** command in the system startup scripts. Thus, under normal circumstances, you do not have to use **mount** to mount this file system.

After you mount a file system, **directory** functions as the root directory of that file system. Also, the operating system records the presence of this file system in the file **/etc/mtab**.

Use the **mount** command by itself (without **device** or **directory**) to list what file systems are mounted, where they are mounted, when they were mounted, and whether they are **writable** (that is, whether their contents can be modified).

You can disconnect mounted file systems with the **umount** command. The general format for the **umount** command is **umount filename**. For example, for AIX PS/2:

¦ Copyright IBM Corp. 1985, 1991

```
  umount /dev/fd0
```

For AIX/370:

```
  umount /dev/disk0001
```

where **filename** is either the special file for a mounted device or the name of the directory on which a device is mounted. To unmount all mounted file systems, use the command **umount all**.

¦ Copyright IBM Corp. 1985, 1991

1.2.8.2 Creating and Mounting Diskette File Systems on AIX PS/2

In general, the procedures for creating and using file systems are the same, whether the file systems reside on minidisks or diskettes. However, before you use diskette file systems, you should be aware of the information in this section. The information in this section is for AIX PS/2 systems only.

Notes:

1. If you are not familiar with the proper way to handle diskettes, see *Installing and Customizing the AIX PS/2 Operating System* before you begin to use diskettes for AIX file systems.

2. If you want to transfer data to or from a DOS formatted diskette, see "Using DOS Formatted Diskettes" in topic 1.2.8.2.4.

Subtopics
1.2.8.2.1 Formatting Diskettes
1.2.8.2.2 Creating Diskette File Systems
1.2.8.2.3 Mounting and Unmounting Diskette File Systems
1.2.8.2.4 Using DOS Formatted Diskettes

¦ Copyright IBM Corp. 1985, 1991

1.2.8.2.1 Formatting Diskettes

Before you can create a file system on a diskette, you must use the **format** command to prepare the diskette to contain data.

Warning: Formatting erases any data stored on the diskette.

Following is the procedure for formatting diskettes:

1. Insert a diskette into the diskette drive. If you have more than one diskette drive, insert the diskette into drive 0 (A).

2. Enter: **format**

For more information about the **format** command, see **format** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.8.2.2 Creating Diskette File Systems

After you have formatted a diskette, you must create a file system on it:

1. Insert a formatted diskette into the diskette drive (or simply leave the diskette in the drive if you have just formatted it). If you have more than one diskette drive, insert the diskette into the drive 0 (A).

2. Enter: **/etc/mkfs /dev/fd0**

¦ Copyright IBM Corp. 1985, 1991

1.2.8.2.3 Mounting and Unmounting Diskette File Systems

To use a diskette file system, insert the diskette into the diskette drive and enter a command of the form:

```
mount directory
```

The standard directory for mounting a diskette file system is **/diskette0**. (If you have a second diskette drive, it is associated with the directory **/machinename/diskette1**.)

Warning: Do not remove a mounted diskette. Damage to the diskette file system could result.

When you finish your work with the diskette file system, unmount it with a command of the form:

```
umount directory
```

When you set up the **/etc/filesystems** stanza for **machinename/diskette0**, the value for **mount** should be **true, removable**. With this value in effect, the system does not automatically mount the diskette file system at startup, but it is ready for you to do so manually.

If you want to mount a diskette on a directory other than **/diskette0**, use a **mount** command of the form: **mount device directory**. For example, if you want to mount a diskette file system on the directory **/mnt**, insert the diskette in drive 0 and enter:

For AIX/PS2:

```
mount /dev/fd0 /mnt
```

¦ Copyright IBM Corp. 1985, 1991

1.2.8.2.4 Using DOS Formatted Diskettes

You can use diskettes to transfer data between the AIX Operating System and a computer that uses the Disk Operating System (DOS). The method you choose to transfer data depends upon whether the DOS Merge Licensed Program Product or the AIX Access for DOS Users Licensed Program Product is installed.

The DOS Merge Licensed Program Product is Installed

If the DOS Merge Licensed Program Product is installed on your system, you can do one of the following to transfer data between the AIX Operating System and a DOS formatted diskette:

Start a DOS session and use the DOS **copy** command to copy data, unaltered, from a DOS formatted diskette to the AIX file system.

At the AIX shell prompt ($), use the DOS **copy** command by entering either **/usr/dbin/copy** or **dos copy source target**. The data is transferred unaltered.

From either the AIX shell prompt ($) or from within a DOS session enter either **aix2dos** or **dos2aix**.

The **aix2dos** and **dos2aix** commands provide options to make the data usable on the destination system.

See *AIX PS/2 DOS Merge User's and Administrator's Guide* for more information on transferring files between a DOS formatted diskette and the AIX file system.

The AIX Access for DOS Users Licensed Program Product is Installed

If the AIX Access for DOS Users Licensed Program Product is installed, you can transfer data between AIX and DOS by doing the following:

1. Start a file services session and use the DOS **copy** command to copy data, unaltered, from a DOS-formatted diskette to the AIX file system.

2. From the DOS prompt, enter either **aix2dos** or **dos2aix**. The **aix2dos** and **dos2aix** commands provide options to make the data usable on the destination system.

See *AIX Access for DOS Users Administrator's Guide* for more information about transferring files between a DOS-formatted diskette and the AIX system.

Neither the DOS Merge Licensed Program Product nor the AIX Access for DOS Users Licensed Program Product is Installed

With the **dosread** and **doswrite** commands, you can use diskettes to transfer data between the AIX Operating System and a DOS formatted diskette. The **dosread** command copies a DOS file from the diskette to the AIX file system. The **doswrite** command copies an AIX file to a DOS diskette. Both commands perform the necessary translations required to make the data usable on the destination system.

The most common form of the **dosread** command is:

dosread -a **dosfilename filename**

¦ Copyright IBM Corp. 1985, 1991

where:

-a specifies translations usually required for text files.

dosfilename is the name of the file you want to copy from the DOS diskette.

filename is the name you want to copy the file to in the AIX file system.

The most common form of the **doswrite** command is:

```
doswrite -a filename dosfilename
```

where:

-a specifies translations usually required for text files.

filename is the name of the file you want to copy from the AIX file system.

dosfilename is the name you want to copy the file to on the DOS diskette.

For more information the **dosread** and **doswrite** commands, see **dosread** and **doswrite** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.9 Backing up Files and File Systems on AIX PS/2

Once your AIX Operating System is set up and in use, your next consideration should be backing up the file systems (there is one file system per minidisk). Whether due to system malfunction or user error, file systems and the data they contain can be damaged or lost. If you take a careful and methodical approach to backing up your file systems, you should always be able to restore recent versions of files or file systems with little difficulty. This section discusses different file and file system backup procedures for an AIX PS/2 system, and how those procedures can be combined into a dependable backup policy.

For information on backup procedures for AIX/370 systems, see the *AIX Administration Guide*.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.2.9.1 Guidelines for Backup Policies

No single backup policy can meet the needs of all AIX users. A policy that works well for a system with one user, for example, could be inadequate for a system that serves five or ten different users. A policy developed for a system on which many files are changed daily would be inefficient for a system on which data changes infrequently. Only you can determine the best backup policy for your system, but the following general guidelines should help:

Make sure you can recover from major losses.

Can your system continue to run after any single fixed disk fails? Can you recover your system if all of the fixed disks fail? Could you recover your system if you lost your backup diskettes or tape in a disaster? Although these things are not likely, any of them is possible. Think through each of these possible losses and design a backup policy that would enable you to recover your system after any of them.

Use your backups periodically.

Diskettes, diskette drives, and tape devices can be unreliable. A large library of backup tapes or diskettes is useless if their data cannot be read back onto a fixed disk. Thus, it is a good idea to make certain that your backups are usable. Try to restore files periodically (for example, to **/dev/null**), just to make sure that they can be read. If you use diskettes for your backups and have more than one diskette drive, try to read diskettes from a different drive than the one on which they were created. Therefore, you may want the security of repeating each level 0 backup with a second set of diskettes. If you use a streaming tape device for backups, you can use the **tapechk** program to perform rudimentary consistency checks on the tape. For more information about the **tapechk** command, see **tapechk** in *AIX Operating System Commands Reference*.

Keep old backups.

You probably will develop some cycle for re-using your backup media--tapes or diskettes. However, you should not re-use all of your backup media. Sometimes it may be months before you, or some other user of your system, notices that an important file is damaged or missing. You should save old backups for just such possibilities. For example, you could have three cycles of backup tapes or diskettes:

- Once per week, recycle all daily diskettes except the one for Friday.

- Once per month, recycle all Friday diskettes except for the one from the last Friday of the month. This makes the last four Friday backups always available.

- Once per quarter, recycle all monthly diskettes except for the last one. Keep the last monthly diskette from each quarter indefinitely, perhaps in a different building.

Check file systems before backing them up.

A backup that was made from a damaged file system may be useless. Before making your backups, it is good policy to check the integrity

¦ Copyright IBM Corp. 1985, 1991

of the file system with the **fsck** command (covered under "Maintaining the File System" in topic 1.3.3).

Your system should not be in use when you make your backups. If you back up a file system while it is in use, files can change while they are being backed up. The backup copy of such a file would not be accurate.

Finally, 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.

¦ Copyright IBM Corp. 1985, 1991

1.2.9.2 Types of Backups

Backup procedures rely upon the following commands:

backup Use the **backup** command to back up individual files or entire file systems. "Individual File Backup" in topic 1.2.9.5 explains how to back up individual files. "Using the backup and restore Commands" in topic 1.2.9.4 explains how to back up complete file systems with the **backup** command.

To use **backup** from the maintenance system, select **Backup commands** from the **USE MAINTENANCE COMMANDS** menu and then select **Back up a file system**.

Note: When you select **Back up a file system** from the **USE MAINTENANCE COMMANDS** menu, the backup is by inode only. (For information about the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.)

restore Use the **restore** command to read files backed up with **backup** to a device or directory. For information on how to restore individual files, see *Using the AIX Operating System*. "Using the backup and restore Commands" in topic 1.2.9.4 explains how to restore complete file systems. (The **restore** command can restore individual files from a complete file system backup.)

To use **restore** from the maintenance system, select **Restore commands** from the **USE MAINTENANCE COMMANDS** menu, and then select **Restore a file system**.

Note: When you select **Restore a file system** from the **USE MAINTENANCE COMMANDS** menu, the restore is by inode only. (For information about the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.)

dd Use the **dd** (device-to-device copy) command to back up entire file systems. The **dd** command is a faster way to back up entire file systems. However, you cannot restore individual files from a **dd** backup. "Backing Up Complete File Systems with the dd Command" in topic 1.2.9.6 explains how to perform file system backups with the **dd** command.

You also can use **dd** from the **USE MAINTENANCE COMMANDS** menu to:

Back up a file system.

Select **Backup commands** and then select **Back up a minidisk image**.

Restore a file system.

Select **Restore commands** and then select **Restore a minidisk image**.

(For information about the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.)

It is very important for you to understand how to use these commands to

¦ Copyright IBM Corp. 1985, 1991

protect the users of your system from loss of data. You should be familiar with the information in:

"Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2

"Backup Media" in topic 1.2.9.3

"Using the backup and restore Commands" in topic 1.2.9.4

"Backing Up Complete File Systems with the dd Command" in topic 1.2.9.6

backup in *AIX Operating System Commands Reference*

restore in *AIX Operating System Commands Reference*

dd in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.9.3 Backup Media

Diskettes are the standard backup medium. Unless you specify a different value for **backupdev** in **/etc/filesystems**, the **backup** command automatically writes its output to **/dev/rfdx**, the device name for the diskette drive. The values for **x** vary according to whether the diskettes in use are low or high density. Therefore, the full device name is either **/dev/rfdl** or **/dev/rfdh**.

When you install a streaming tape drive, the **devices** command automatically changes the value of **backupdev** to **/dev/rmt0**, the device name for the tape (on a PS/2). The **rmt0** value makes the tape rewind when it is finished backing up a file system. If you change the value to **rmt4**, the tape does not rewind.

Note: You may need to change the value for **backuplen**. **backuplen** is the size in blocks of the backup device (diskette or tape) to be used for computing the amount of data that will fit on each diskette. For example, a **backuplen** of 2700 represents a 300-foot tape with nine tracks (300x9=2700).

If you remove a tape drive (with the devices command) from the system, **devices** changes the value of **backupdev** back to **/dev/rfd0** and the value of **backuplen** back to 2400.

After you install a tape device, the **backup** command uses the tape as the backup device for complete file system backups (the procedure described under "Using the backup and restore Commands" in topic 1.2.9.4). However, even with a tape installed, the **backup** command still backs up individual files to diskettes. For information about file backups, see *Using the AIX Operating System*.

¦ Copyright IBM Corp. 1985, 1991

1.2.9.4 Using the backup and restore Commands

The **backup** command allows you to back up files selectively or to back up entire file systems. Similarly, you can use the **restore** command to restore all or part of the backups you create with **backup**.

Both **backup** and **restore** are available from the Use Maintenance Commands menu in the maintenance system. For information on the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.

Note: If you only want to back up and restore complete file systems, it may be faster to use the **dd** command than it is to use **backup** and **restore**. Among the factors that determine which backup method is faster are the size of the minidisk to be backed up and how many of its blocks are allocated.

This section explains how to use **backup** and **restore** for complete file systems.

Notes:

1. If you use diskettes as your backup media, have several formatted diskettes ready to use before you enter the **backup** command.

2. The time and date information recorded for a backed up file reflect when the backup was made. Ordinarily, the **restore** command preserves this time and date information. However, if you want restored files to reflect when they were restored (rather than when they were backed up), use the **-h** flag with **restore**.

To Back Up a File System with backup

1. Prepare the backup medium:

   Make certain that the tape device is ready to operate, or

   Insert a formatted diskette into drive 0.

2. Enter:

   backup -0 -u filesystem

where **filesystem** is the name of the file system to be backed up.

(The system prompts you for additional backup media as required.)

To Restore a File System

1. Load the tape or diskette containing the file system to be restored. (If you restore from diskettes and the backup occupies more than one diskette, insert the first diskette of the group into drive 0.)

2. Enter:

   restore -r filesystem

¦ Copyright IBM Corp. 1985, 1991

where **filesystem** is either the name of a physical device or a directory name listed in the **/etc/filesystems** file. The **-r** flag is for inode backups only.

(The system prompts you for additional media as necessary, for example, for the second of a group of diskettes.)

Subtopics
1.2.9.4.1 Volume Backups
1.2.9.4.2 Incremental Backups

¦ Copyright IBM Corp. 1985, 1991

1.2.9.4.1 Volume Backups

The **backup** command individually backs up each file in a file system. Thus, you do not have to restore an entire file system if all you need to do is restore specific files. Also, you can restore a file or group of files to any file system that is large enough to accommodate them.

To back up an entire file system, use a command of the following form:

```
backup -0 -u filesystem
```

Following is an explanation of the parts of this command:

-0 A flag that indicates the **level** of the backup. Level **0** causes every file in the file system to be backed up. Use other backup levels (1-9) for incremental dumps, as is explained under "Incremental Backups" in topic 1.2.9.4.2.

-u A flag that causes **backup** to record the date and level of the backup in the **/etc/budate** file. This file provides the information needed for incremental backups.

filesystem The name of the file system to be backed up. **filesystem** can be either the name of the physical device that contains the file system or the name of the file system's root directory.

Unless you specify otherwise on the command line, the **backup** command uses information in **/etc/filesystems** to determine:

dev The device, or minidisk, that contains the file system.

backupdev The device (for example, a diskette drive) on which the backup is to be made.

backuplen The size in blocks of the backup device (diskette or tape) to be used (for computing the amount of data that will fit on each diskette). For more information, see "Backup Media."

Note: When you restore a complete file system, the **restore** command organizes the files efficiently, but does not reduce the total amount of storage space needed for the file system. The **-m** flag backs up an entire minidisk as an exact image. An image backup is appropriate for backing up very large AIX file systems or minidisks that do not contain AIX file systems. A backup by minidisk (**backup -m**) must be restored by minidisk (**restore -m**).

¦ Copyright IBM Corp. 1985, 1991

1.2.9.4.2 Incremental Backups

A level 0 backup can require a considerable amount of time and a large number of diskettes to complete. Because only a fraction of the files on your system can change from day to day, it is not necessary for you to make a level 0 backup daily. Instead, you can make **incremental** backups--backups of only the files changed since a previous backup.

An incremental backup backs up all files changed since the last backup at the next lower level. For example, the following level 1 **backup** command backs up all files changed since the last level 0 backup:

```
backup -1 -u filesystem
```

The parameter that indicates the level of the backup (**-1** in this example) can be any number from 0 through 9. Thus, a level 4 backup backs up all files that have changed since the last level 3 backup, and so forth. You can use the backup levels to develop a very dependable backup system. For example, you might make level 0 backups monthly, level 1 backups weekly, and level 2 backups daily. Then, should you have to restore a complete file system, you would restore from the latest level 0 backup, then from the latest level 1 backup, and finally from the latest level 2 backup. For many systems, it should be adequate to use only two levels of backup (for example, level 0 weekly and level 1 daily or level 0 monthly and level 1 weekly).

¦ Copyright IBM Corp. 1985, 1991

1.2.9.5 Individual File Backup

You can also use the **backup** command to back up individual files, for example, when some or all of the files that belong to a particular user will not be needed for an extended period. Backup copies of individual files are also a convenient means for exchanging data among different systems (for example, you backup the files from your system onto a diskette and then the user of another system restores the files from the diskette). To back up individual files, use the **-i** (backup by name) flag with the **backup** command.

The following sequence backs up three files to a backup device defined in **/etc/filesystems**:

```
backup -i
Please mount volume 1 on /dev/backup device
. . . and type return to backup
file1
file2
file3
END OF FILE
Done at date and time
n blocks on n volume(s)
```

You can also use the **backup** command with a list of file names created with an editor. In the following example, **backup** makes backup copies of the files named in the file **list**:

```
backup -i < list
```

Note: When piping the output of the **find** command to **backup**, be sure to use the **find -hidden** flag if you wish to include files in hidden directories.

To back up all files and subdirectories of the current directory, use the command:

```
find . -print | backup -i
```

To restore files that have been backed up by name, use the command:

```
restore -x
```

If your backup device is a streaming tape drive, you can perform faster **backup -i** and **restore -x** operations by using these commands in a pipeline with the **dd** command. For example, the following pipeline pipes the output of **find** to **backup**, and then pipes the output of **backup** to **dd**:

```
find . -print | backup -if- -C30 | dd of=/dev/rmt0 bs=30b
```

In this example, the **f-** flag causes **backup** to write to standard output, and **-C30** specifies the number of blocks in a single output operation (the cluster size). The **of=/dev/rmt0** parameter causes **dd** to write to the streaming tape device, and **bs=30b** specifies the size of a single input or output operation (30 blocks). You may determine that other values for **-C** and **bs=** give better performance in your application.

To restore files backed up by piping the output of **backup** to **dd**, use a pipeline of the form:

¦ Copyright IBM Corp. 1985, 1991

```
dd if=/dev/rmt0 bs=30b | restore -xf-
```

In this example, **if=/dev/rmt0 bs=30b** causes **dd** to read from the tape device with an input and output operation size of 30 blocks. The flags to the **restore** command cause it to restore by name (**-i**), reading from standard input (**f-**).

¦ Copyright IBM Corp. 1985, 1991

1.2.9.6 Backing Up Complete File Systems with the dd Command

When you want to back up (and restore) only complete file systems, the **dd** (device-to-device copy) command gives you a potentially faster alternative to **backup** and **restore**. The **backup -m** command actually uses **dd** and is a convenient way to backup file systems. However, this section describes how to use the **dd** command.

The **dd** command is available from the Use Maintenance Commands menu in the maintenance system. To use **dd** from the maintenance system:

To back up a file system, select **Backup commands** and then select **Back up a minidisk image**.

To restore a file system, select **Restore commands** and then select **Restore a minidisk image**.

For information on the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.

To back up a file system with **dd**, you need to specify the following parameters:

if The name of the input file. Use the value of **dev =** in the **/etc/filesystems** stanza for the appropriate file system. **dd** works more quickly with the raw versions of devices (that is, the ones which begin with **r**).

of The name of the output file. Use the device name for the streaming tape device (**/dev/rmt0**).

bs The input and output block sizes, in bytes.

The following **dd** command backs up 80 blocks from the file system on **/dev/hdn** to the streaming tape, **/dev/rmt0**, using a logical block size of 4096 bytes:

```
dd if=/dev/rhdn of=/dev/rmt0 bs=4096 count=80
```

To restore a file system backed up with **dd**, use the **dd** command again, copying the file system from the backup device to the minidisk.

Note: The **dd** command may be less efficient than the **backup** command if the file system being backed up does not contain many files. Also, if you restore a backup made with **dd** to a different minidisk, its size may change.

¦ Copyright IBM Corp. 1985, 1991

1.2.9.7 Using Stand-alone Backup on an AIX PS/2 System

The stand-alone backup procedure is available only on AIX PS/2 systems. There are two types of backups you can perform using the stand-alone backup procedure:

A minidisk backup. A backup by minidisk is an image copy of a minidisk that can only be restored in one piece and must be restored onto at least as large a minidisk as the original minidisk that was backed up. If the minidisk is larger than the original, the leftover space becomes unusable after restoring the minidisk.

An inode backup. A backup by inode is an inode by inode backup; that is, each file and directory is individually saved. This allows incremental backups--only those files that have been changed since the previous level of backup need to be backed up. In addition, individual inodes (files) can be restored without restoring the entire backup, and inode backups can be restored onto larger minidisks without wasting the unused space.

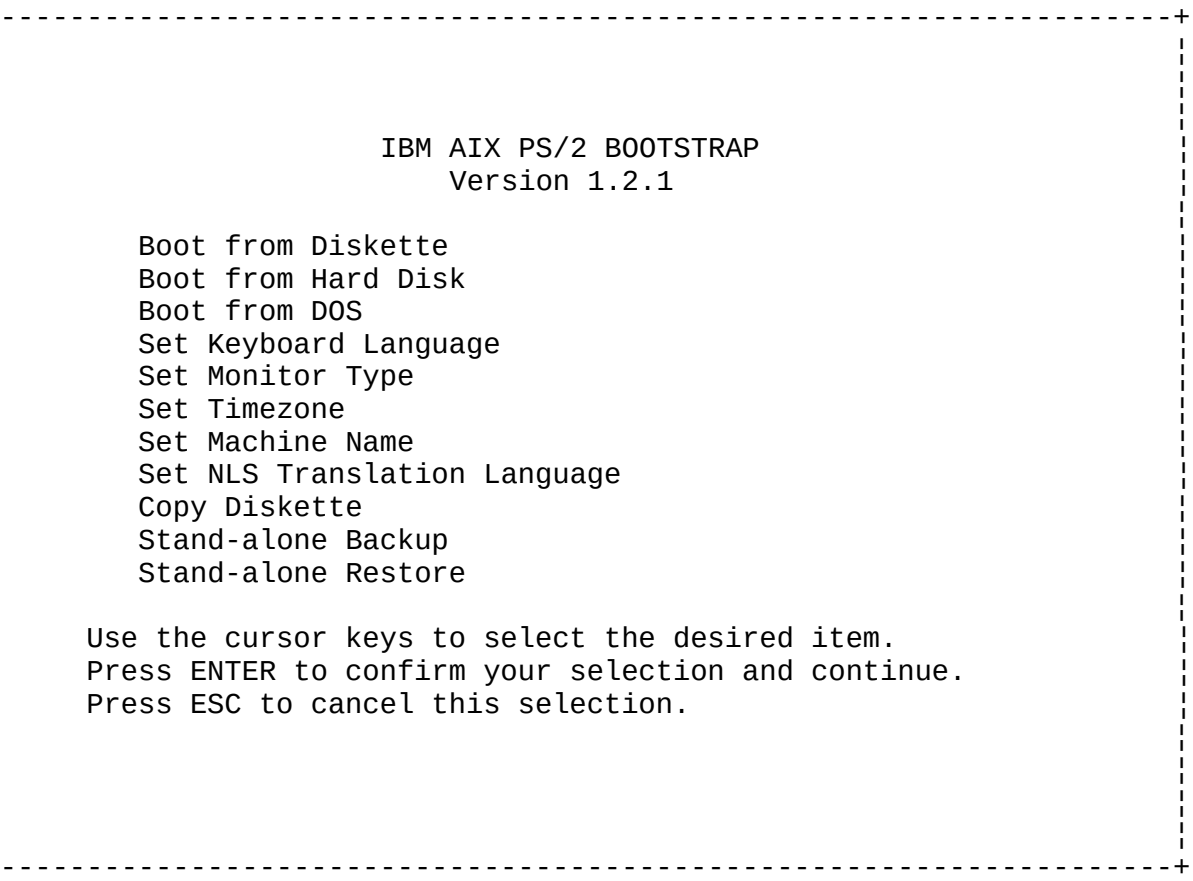
For more information about backups, see **backup** in *AIX Operating System Commands Reference*.

To perform a stand-alone backup, do the following:

1. Insert the Boot diskette in the disk drive.

2. Boot your computer from the diskette.

   The following menu appears on your screen:



3. Select **Stand-alone Backup** and press **Enter**.

¦ Copyright IBM Corp. 1985, 1991

The backup procedure begins by displaying the following message:

```
backup by minidisk (m) or inode (i) (default m)?
```

4. Enter **m** for a minidisk backup or **i** for a backup by inode.

   If you do not enter a value in this field, the system defaults to a backup by minidisk.

   Backing up by minidisk creates an exact image of the entire minidisk. For this reason, backing up a large minidisk with a small or sparsely used file system may take longer and require more backup medium than backing up this file system by inode.

5. When the system prompts you for the output device, enter the output device you want to use for this backup.

   You should specify the device as a file name. For example, you might enter **/dev/fd0** to use diskette drive 0 for the backup. If you enter a device other than a tape device, go to step 8. Otherwise, continue.

6. When the system prompts you for the density, enter the density of the tape in bytes per inch that you plan to use for this backup. The default density is 700 bytes per inch.

7. When the system prompts you for tape length, enter in the appropriate tape length.

   The tape length is a combination of the physical length and the number of tracks on the tape. In the case of a streaming tape, multiply the physical length of the tape by 9 (the number of tracks) to determine the usable space available.

8. When the system prompts you for the file system disk, enter the device name for the file system disk that you want to back up.

   You must enter the physical device name of the file system or minidisk to be backed up.

9. When the system prompts you for the output cluster size, enter the number of blocks to write in a single output operation.

   If you do not specify the output cluster size, a default value is used that is appropriate to the physical device entered. Larger output cluster sizes result in longer physical transfers to tape devices. For diskette devices, the output cluster size entered is ignored and the backup program writes in clusters that occupy a complete track on the diskette.

   If you are performing a backup by minidisk, you have finished answering the necessary questions. Messages similar to the following are displayed to confirm the information:

   ```
   Filesystem is /dev/hd7
   backing up /dev/hd7 to /dev/fd0
   Please mount volume 1 on /dev/fd0
   ```

   If you are performing a backup by minidisk, go to step 13.

   If you are performing a backup by inode, continue.

¦ Copyright IBM Corp. 1985, 1991

10. When the system prompts you, enter the total number of blocks to use on the output device.

    If you do not enter a number of blocks, the backup will use all the blocks on the output device.

11. When the system displays the following prompt: **incremental backup (0-9) (default 9)?**, enter the backup level of this backup if you are performing an incremental backup.

    By entering a backup level, you specify the files you want to backup. You can specify a backup level between 0 and 9. If you specify 0, the backup includes all files on the file system. Otherwise, a backup level of **n** includes all files modified since the last **n**-level backup. If you do not specify a backup level, a level of 9 is used. For more information on backup strategies and using incremental backups, see "Guidelines for Backup Policies" in topic 1.2.9.1 and "Types of Backups" in topic 1.2.9.2.

12. When the system prompts you for the last modification date of files to be backed up, enter the last modification date for this backup.

    Files modified on or after this date will be backed up.

    After you enter this date, messages similar to the following are displayed:

    ```
    WARNING:  only files modified since date will be backed up.
    However, restore will think these files were backed up since the epoch.
    Filesystem is /dev/hd7
    backing up /dev/hd7 to /dev/fd0
    Please mount volume 1 on /dev/fd0
    ```

13. Mount the backup medium on the backup device and press **Enter**.

    For example, if you are backing up to diskette, insert the diskette on which you want to backup the files in the diskette drive and press **Enter**. If you are backing up to tape, mount the tape on the tape drive and press **Enter**.

    When you press **Enter**, messages similar to the following are displayed as the system begins backing up the files:

    ```
    Backing up to /dev/fd0
     Cluster 24576 bytes (24 bytes), maximum 1440 blocks
    Volume 1 on /dev/fd0
    ```

14. When the system prompts you to mount volume 2, remove the first backup tape or diskette from the backup device, mount the second backup diskette or tape, and press **Enter**.

    The system continues the backup. Each time a diskette or tape is filled, the system prompts you for the next volume. When the backup is complete, the system displays the number of blocks backed up and the number of volumes used. The system also displays a message to insert the bootable diskette and press any key to restart.

15. Remove the backup medium from the backup device, make sure that the

¦ Copyright IBM Corp. 1985, 1991

Boot diskette is in drive **fd0**, and reboot by pressing **Enter**.

For example, if you are backing up to diskette, remove the diskette from the diskette drive and reboot your system. If you are backing up to tape, remove the tape on the tape drive and reboot your system.

¦ Copyright IBM Corp. 1985, 1991

1.2.9.8 Restoring from a Stand-alone Backup on an AIX PS/2 System

The stand-alone restore procedure is available only on AIX PS/2 Systems. The stand-alone restore procedure reads backups made with the stand-alone or nonstand-alone backup procedure. This sections describes the procedure you should follow for restoring information from a stand-alone backup.

You can use the stand-alone restore procedure in two ways:

To restore a minidis

To restore an entire file system by inode

Before you begin the restore procedure from a stand-alone backup, you should consider the following precautions:

On all restores, files must be restored using the same method by whic they were backed up. For example, if a file system was backed up by minidisk, you must restore it by minidisk.

On inode restores, if you are restoring a full (level 0) backup, ru the **mkfs** command to create an empty file system before doing the restore. If you are restoring an incremental backup, run **mkfs**, then restore the appropriate level 0 backup, and then the lower-numbered incremental backups in numerical order. For example, if you are restoring a level 2 incremental backup, you should run **mkfs**, restore the level 0 backup, restore the level 1 backup, and then restore the level 2 backup.

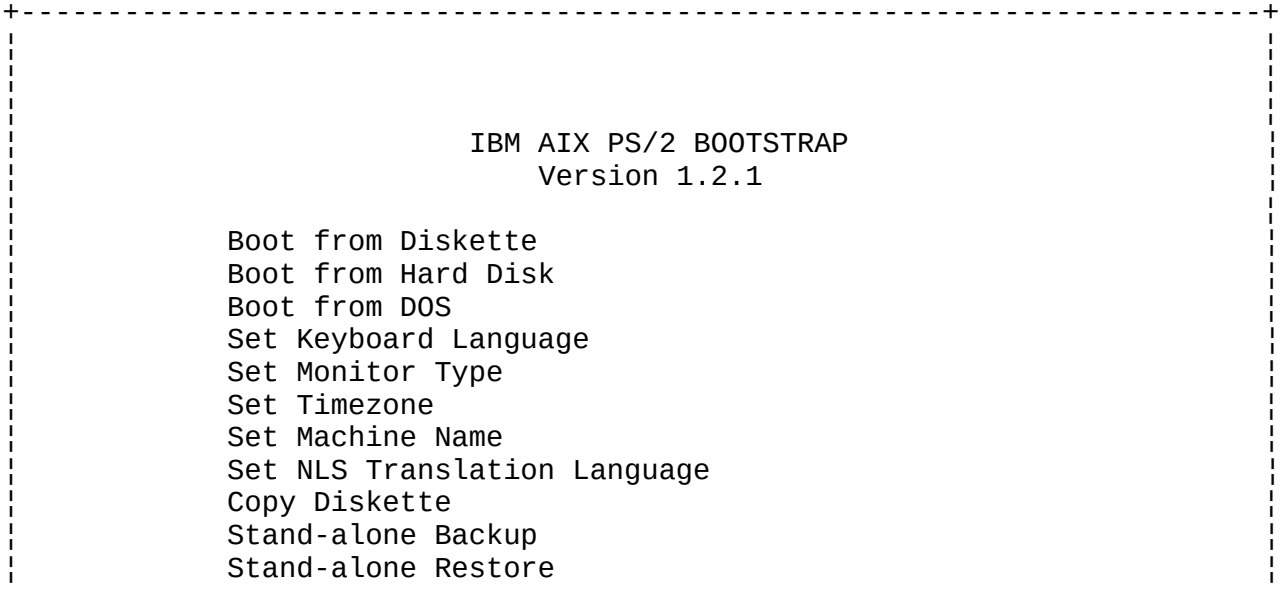
On minidisk restores, you must use a minidisk that is at least a large as the original minidisk that was backed up. If the minidisk is larger than the original, the leftover space becomes unusable after restoring the minidisk.

To perform a restore:

1. Insert the backup boot diskette in the disk drive.

2. Boot your computer from the diskette.

   The following menu appears on your screen:



¦ Copyright IBM Corp. 1985, 1991

```
|                                                                      |
|        Use the cursor keys to select the desired item.               |
|        Press ENTER to confirm your selection and continue.           |
|        Press ESC to cancel this selection.                           |
|                                                                      |
|                                                                      |
|                                                                      |
|                                                                      |
+----------------------------------------------------------------------+
```

3. Select **stand-alone restore** and press **Enter**.

   The restore procedure begins by displaying the following message:

   ```
   restore by minidisk (m) or inode (i):
   ```

4. Enter **m** for a backup by minidisk or **i** for a backup by inode.

   Files must be restored using the same method by which they were backed up. For example, if a file system was backed up by minidisk, you must restore it by minidisk.

   If you restore by minidisk, you must use a minidisk that is at least as large as the original minidisk that was backed up. If the minidisk is larger than the original, the leftover space becomes unusable after restoring the minidisk.

   If you do not enter a value in this field, the system defaults to restore by minidisk.

5. When the system prompts you for the input device, enter the input device you want to use for this restore.

   You should specify the device as a file name. For example, you might enter **/dev/fd0** to use diskette drive 0 for the restore.

6. When the system prompts you for the file system disk, enter the device name for the file system disk that you want to restore.

   The file system must be a device name (block or character special file).

   If you are restoring a full (level 0) backup, run the **mkfs** command to create an empty file system before doing the restore. If you are restoring an incremental backup, run **mkfs**, then restore the appropriate level 0 backup, and then the lower-numbered incremental backups in numerical order. For example, if you are restoring a level 2 incremental backup, you should run **mkfs**, restore the level 0 backup, restore the level 1 backup, and then restore the level 2 backup.

   Warning: If you do not follow this procedure carefully, you can ruin an entire file system. As an added safety precaution, run **fsck** after you restore each backup level.

7. When the system prompts for input cluster size, enter the number of blocks to read in a single input operation.

   If you do not specify the input cluster size, a default value is used that is appropriate to the physical device entered. Larger input cluster sizes result in longer physical transfers from tape devices.

¦ Copyright IBM Corp. 1985, 1991

For diskette devices, the input cluster size entered is ignored and the restore program reads in clusters that occupy a complete track on the diskette.

```
Filesystem is /dev/hd7
Please mount volume 1 on /dev/fd0
```

8. Mount the diskette or tape containing the files you want to restore and press **Enter**.

   For example, if you are restoring from diskette, insert the diskette containing the files in the diskette drive and press **Enter**. If you are restoring from tape, mount the tape on the tape drive and press **Enter**.

9. When the system prompts you to mount volume 2, remove the first tape or diskette, mount the second diskette or tape, and press **Enter**.

   The system continues the restore. Each time the system finishes restoring information from a diskette or tape, the system prompts you for the next volume. When the restore is complete, the system displays the number of files restored.

10. Remove the diskette or tape from the drive and reboot your system.

    For example, if you are restoring from diskette, remove the diskette from the diskette and reboot your system. If you are restoring from tape, remove the tape on the tape drive and reboot your system.

¦ Copyright IBM Corp. 1985, 1991

1.2.10 Understanding System Security

The key to effective security is understanding how the security features work and then using them conscientiously. If more than one person uses your system, it is important for everyone to understand how security works and the importance of observing certain security guidelines.

The principal AIX security features are described in other sections of this book. This section summarizes important security features and indicates where you can find more detailed information.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.2.10.1 Passwords

Passwords help protect your system against unauthorized access. The AIX Operating System encrypts passwords, stores them in a file, and then compares the password supplied when a user tries to log in with the encrypted version. If the two match, the user gains access to the system.

While it is not mandatory to use passwords on your system, it is strongly advised, even if you are the system's only user. Without password protection, all data on your system is available to anyone who knows how to turn on the power switch and enter a valid user name (generally, user names are easy to obtain or guess). Even if your system does not contain sensitive data, an unauthorized user can cause problems by altering the contents of files or turning off the system without running **shutdown**.

For more information about passwords, the password file, and the **passwd** command, see the following:

Using the AIX Operating System

"Managing User Accounts" in topic 1.2.5

passwd in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.2.10.2 File Protections

Every file and directory in the AIX Operating System has a group of permissions associated with it. The permissions define who can use the file in what way. In other words, the permissions grant certain users access to the files and directories and protect the files and directories from other users.

For an explanation of how the file and directory permissions work, see *Using the AIX Operating System*. While permissions afford an effective way to control access to data stored in the system, they are only as effective as users make them. Thus, it is important for all users of your system to understand how the permissions work, how they can be modified, and how their effectiveness can be undermined by a lax attitude toward system security.

¦ Copyright IBM Corp. 1985, 1991

1.2.10.3 Invalid Login Attempts

Invalid login attempts due to an incorrect login name or password are recorded in the **/etc/.ilog** file. The contents of this file can be examined only by the user **root** or **su**, or a member of the system group. Each time you log in as **root** or **su** and there is an entry in **/etc/.ilog**, the system displays a message advising you to check the contents of **/etc/.ilog**. To look at this file, use the **who** command. For example:

```
who /etc/.ilog
```

¦ Copyright IBM Corp. 1985, 1991

1.2.10.4 Site Permission

Site permission protects your system from unauthorized process access. When TCF is being run, all users in the cluster can access any files mounted on the replicated root file system, run processes on any host they choose, and transfer their login location to any processor they want. Access to file systems is restricted by setting the appropriate permissions on the files themselves and is discussed in "File Protections" in topic 1.2.10.2. Access to processors is restricted by setting site permission.

By setting site permission, you can restrict a user to a single host or any number of hosts in the cluster. You can also assure that certain machines are used only by members of a certain group. For example, you may want to set aside a certain group of machines for work requiring high security, work that is potentially dangerous to system files, or work that is likely to disrupt system service frequently.

To set up a restricted group of machines, follow these steps:

1. Edit the **/etc/sitegroup** file, adding the name of the site group and the sites accessible to this group. The site group name is terminated by a colon (:). Each entry following is separated by a semicolon (;). For example,

   ```
   accounting:1;2;3;4;5;6;9;10
   warehouse:3;4;5;6;7;8;9;10
   ```

   The above example describes a ten-site cluster. The site group **accounting** cannot run processes on sites **7** and **8**. The site group **warehouse** is restricted from sites **1** and **2**. Users who are not members of either of these groups can run processes on any site.

2. Use the **adduser** command (see "Creating, Changing, and Removing Accounts - The adduser Command" in topic 1.2.5.1) to add the appropriate users to the site group you have defined in the **/etc/sitegroup** file.

The simplest use of site permission is to divide the system into two groups: ordinary and privileged. Ordinary users would be restricted to certain sites while privileged users can run processes anywhere. Privileged users would be those users who you do not define as members of the ordinary site group. By default, they have access to every site in the cluster.

To see the site permission settings currently in effect, use the following command:

```
pg /etc/passwd
```

The site permission settings are found in the **site_exec_perm** subfield of the **full name** field in this password file.

For additional information, refer to:

The **adduser** command in *AIX Operating System Commands Reference*

The **/etc/sitegroup** file in *AIX Operating System Technical Reference*

"The /etc/passwd File" in topic 1.2.5.4.1 and the **/etc/passwd** file in

¦ Copyright IBM Corp. 1985, 1991

AIX Operating System Technical Reference.

¦ Copyright IBM Corp. 1985, 1991

1.2.10.5 Terminal Logging

The terminal logging daemon, **tlogger**, collects all data read or written to the associated terminal, except passwords, and writes the data to a log file. If standard error is a terminal device, it is used as the associated terminal; otherwise the process's **usrinfo** is used to identify the login tty and that device is used as the associated terminal. You can log data to a terminal other than the login terminal by redirecting standard error. You can also have multiple **tlogger** commands running at the same time if each logs data for a different tty device, and each uses a different log file.

To start **tlogger** enter:

```
/etc/tlogger &
```

Flags available with **tlogger** are:

-cfile Specifies **file** as the name of the log file. The log file defaults to **/usr/adm/ras/tlogfile**.

-bfile Specifies **file** as the name of the back-up log file. The back-up log file defaults to **/usr/adm/ras/tlogfile.bk**.

Each time that you start **tlogger** the current log file is written to the back-up file and a new log file starts with the permissions set to allow read and write by the owner.

Two commands, **tlog off** and **tlog on**, let you stop and then restart sending data to the log file. Enter **tlog off** to stop sending data to the log file while you are logged on the terminal. The **tlogger** daemon continues to run but your activities are not logged to the log file. Before you log off of the terminal enter **tlog on** to restart the logging process. The **tlogger** daemon continues to run even though you are not logged on; this provides information on any console activity while you are not logged on.

You can end **tlogger** with the **kill** command or it ends when shutdown occurs. **tlogger** also ends if the log file becomes full and no data can be written to it. You should not use SIGKILL to stop **tlogger** since system resource clean-up cannot be performed.

¦ Copyright IBM Corp. 1985, 1991

1.3 Chapter 3. Maintaining the AIX Operating System

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.3.1 Contents

¦ Copyright IBM Corp. 1985, 1991

1.3.2 About This Chapter

This chapter covers the tasks required to maintain the AIX Operating System on your workstation and to adapt the system to your needs. Anyone who manages an AIX Operating System will perform some of the tasks described in this chapter (for example, maintaining the file system). Other tasks described here may become more important the longer you use the system (for example, using the queueing system). A third set of tasks only concern you if you modify your system (for example, generating a new kernel).

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System,* and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

1.3.3 Maintaining the File System

Each time a file is created, changed, or deleted, the operating system performs a series of file system updates. These updates, when written to a disk, produce a ***consistent*** file system. If all of the updates do not complete successfully, the result can be an inconsistent file system. Some inconsistencies may not be severe, but others, if not promptly corrected, can spread and eventually make an entire file system unusable. The **fsck** (file system consistency check) command analyzes a file system, locates inconsistencies, and often can correct certain inconsistencies automatically.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.3.3.1 Causes of File System Damage

The major causes of file system damage are:

A system halting (or failing) before it completes all pending fil system updates, for example, due to power failure

Poor operating procedure

Failure of a disk drive or drive controlle

Physical damage to a diskette

When a system halts before it completes its pending file system updates, damage to the file system is minor and easily corrected. File system damage that results from poor operating procedures or from fixed disk or disk controller failure, however, can be severe enough to destroy an entire file system.

Subtopics
1.3.3.1.1 Disk Buffering - The sync Command

¦ Copyright IBM Corp. 1985, 1991

1.3.3.1.1 Disk Buffering - The sync Command

Any operation involving a file requires data to be retrieved from the fixed disk (read into memory), returned (written) to the disk, or both. These read and write operations also are known as disk input and output, or ***disk I/O***. To make some disk I/O more efficient, the operating system maintains a ***buffer*** (temporary storage area) in memory for recently accessed data blocks. The operating system writes data from the buffer to the disk only when:

There is a **sync** system call.

There is an **fsync** system call.

There is a **fcommit** system call.

The system needs the buffer for another purpose

Disk buffering can increase the efficiency of I/O operations because it allows repeated reads and writes to the same block to occur without the block being physically read from the disk or written to the disk each time. The system is also able to schedule certain I/O operations (especially read operations) so that they occur in an order that is more efficient for the disk, rather than in the order in which they were requested. Together, disk buffering and disk scheduling help match the demand for I/O operations to the physical characteristics of the fixed disk.

Although it may improve the efficiency of disk I/O, disk buffering can indirectly cause one type of file system damage--if the system stops before writing all data from the buffer back to the disk, the data on the disk is not current (and the data in memory is lost). Thus, it is extremely important for buffered data to be written to the disk (with the **sync** command) before the system is stopped. The **shutdown** command automatically runs the **sync** command. Failure to run **sync** (with **shutdown**) before stopping the system is a common cause of file system damage, but one that is very easy to avoid. During normal operation, the system automatically does a **sync** on a regular basis.

Even if the system fails, which usually means that you do not have time to run the **sync** command, any structural damage to the file system is usually easy to find and repair. The **fsck** command often can automatically repair this kind of damage. Thus, before you use the file system, you check it with **fsck** (either use the **Check a file system** item on the AIX Operating System Maintenance diskette, run **fsck** from single-user mode, or run **fsck** from the standalone shell, as described under "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2). Otherwise, if files are created on the damaged file system, the damage can become much worse.

The **fsck** command lets you check two or more file systems on two different drives at the same time. By specifying a check number in **/etc/filesystems**, it is possible to check several file systems in parallel. **fsck** prints the file system name for each message that it sends. For more information about **fsck**, see the *AIX Operating System Commands Reference*.

Note: The **fsck -p** command should be included in **/etc/rc** so that it always runs during normal startup. To check two file systems on two different drives at the same time during normal startup, include the **dfsck** command in **/etc/rc**.

¦ Copyright IBM Corp. 1985, 1991

1.3.3.2 Examples

Here are two examples of how a file system can become inconsistent. In each example, the system stops before all disk updates are complete.

Example 1. Block Allocation Inconsistenc

When the operating system allocates a new block to a file, three things must be updated:

- The new block must be written.
- The inode, to indicate that the file contains a new block.
- The free list, to indicate that a block that was free is no longer available.

If the system stops between the second and third update operations, the inode shows that the new block is allocated to the file; while the free list still shows that the block is available. There is an inconsistency in the file system.

The **fsck** command often can find and correct such inconsistencies if it is run on the file system immediately. However, if you begin to use the file system again without correcting the inconsistency, more serious problems can occur. For example, the block shown as both allocated (in the inode) and free (in the free list) could be allocated to a second inode; that is, the same block would be part of two different files or directories. File system damage of this type can spread quickly and be extremely difficult to correct; it can result in the appearance of strange information in files and the loss of some files or even an entire file system.

Example 2. File Deletion Inconsistenc

When the operating system deletes a file, up to five things must be updated:

- The directory entry that points to the file must be cleared.
- The reference count in the inode for the file must be reduced by one.
- If the reference count goes to zero, all blocks in the file must be added to the free list.
- If a large number of blocks are freed, a new link in the free chain must be written.
- The inode for the file must be made available again.

Depending upon which of these update operations is completed at the time the system stops, different types of file system inconsistencies can occur.

Note: Lost data does not necessarily produce file system inconsistency. Generally, the data stored in ordinary files has nothing to do with the consistency of the file system. For example, if you begin a **write** operation and the system stops before the operation completes, the file on the disk simply does not get updated. While

¦ Copyright IBM Corp. 1985, 1991

you have lost data, that loss may not affect the file system. The **fsck** command can only determine whether the structure of the file system is internally consistent; it cannot check the data within files.

¦ Copyright IBM Corp. 1985, 1991

1.3.3.3 Checking and Repairing File Systems - The fsck Command

A file system contains redundant information about its structure. If the file system is consistent, each instance of a particular piece of information is identical. The **fsck** command uses this redundant information to detect and, in many cases, correct file system inconsistencies.

The **fsck** program makes several passes through a file system, performing a different phase of checking in each pass. When it detects an inconsistency, **fsck** reports the error condition, displaying a message on the screen and waiting for a response. (This is the default mode for **fsck**.) For an explanation of the **fsck** messages and what actions you should take to respond to them, see *AIX Operating System Messages Reference*.

The system startup files contain a list of the commands run each time the system starts. Generally, startup scripts contain the **/etc/fsck -p** command. This line starts **fsck** before the system mounts any file systems. If the **/etc/filesystems** stanza for a file system contains either of the following lines, **fsck** checks the file system at this point:

check=true

check=group.

Note: A ***stanza*** is a group of lines that define a part of the system. A stanza in **/etc/filesystems** defines a file system. Use ASCII characters to ensure successful communication across different locales.

The **-p** (preen) flag allows the **fsck** command to check file systems in parallel. If you are checking many file systems, it is a good idea to use the **-p** flag for efficiency.

File systems that are located on separate disks and administrated by separate controllers can be checked in parallel with no interference problems. However, file systems on the same physical disk should be checked sequentially; otherwise, the read/write heads will be in constant motion. A worst case scenario would be parallel checking of two file systems located on AIX minidisks at opposite ends of the same physical disk. The heads would be in constant motion from one end of the disk to the other.

The solution is to divide the file systems into groups. All file systems on the same physical disk should be grouped together. Then, using the **fsck -p** command starts a separate process for each file system group. The processes run in parallel, but the file systems in each group are checked sequentially.

For details on how files are grouped together, see the *AIX Commands Reference* (the **fsck**, **dfsck** command).

The **-p** flag causes **fsck** to run automatically, correcting all simple inconsistencies without requiring any action from the user. The **-p** flag enables **fsck** to repair most inconsistencies that result from the system stopping unexpectedly.

Without the **-p** flag, **fsck** runs interactively, displaying each error condition that it locates and waiting for you to decide whether

¦ Copyright IBM Corp. 1985, 1991

corrections should be made. For information about other **fsck** flags, see "**fsck**" in *AIX Operating System Commands Reference*.

It is very important to run **fsck** on unmounted file systems. The **fsck** command goes completely through the file system several times, often comparing data collected on one pass with data collected on another. Thus, any activity on the file system can interfere with the ability of **fsck** to check and repair file system damage.

Subtopics
1.3.3.3.1 The fsck Consistency Checks

¦ Copyright IBM Corp. 1985, 1991

1.3.3.3.1 The fsck Consistency Checks

Here is a list of types of file system inconsistencies that **fsck** checks for together with descriptions of how they can be corrected:

Superblock Inconsistencie

The superblock is the most frequently updated part of a file system; and it is, therefore, the most likely to be inconsistent after the system stops unexpectedly. Every time a block or inode is allocated or deallocated, the superblock should be updated.

The **fsck** program checks all information about the file system in the superblock to make sure that it is consistent with the standard requirements for a file system. This information (including file system size, number of blocks allocated to inodes, cluster size, and interleave factors) never changes during normal operation. Should any of these values change, there is evidence that the file system is severely damaged. In such cases, **fsck** reports the condition and does nothing else with the file system. Usually the only way to recover such a file system is to restore it from a backup.

If the information about the file system seems to be consistent and reasonable, **fsck** uses the file system and i-list size to validate all block numbers. All reasonable blocks are between the end of the i-list and the end of the file system. Any reference to a block not within that range is not valid and probably indicates severe file system damage.

- Free block list

  The free list is a series of logically connected blocks which are available (that is, they are not allocated to any file). The superblock contains the first block of the free list. The **fsck** command checks the structure of the free list and also checks the block numbers of every block in the list. **fsck** makes separate checks to make sure that no allocated blocks appear in the free list and that all blocks that are not allocated do appear in it. **fsck** can repair any inconsistencies found in the free list, usually by constructing a new free list that contains all blocks found not to be allocated.

- Free block count

  The superblock contains a count of the total number of free blocks within the file system. **fsck** compares this count to the number of blocks it found free within the file system. If the counts disagree, **fsck** can replace the count in the superblock with the actual free block count.

- Free inode count

  The superblock contains a count of the total number of free inodes within the file system. The **fsck** command compares this count to the number of inodes it found free within the file system. If these counts disagree, **fsck** can replace the count in the superblock with the actual free inode count.

Inode Inconsistencie

¦ Copyright IBM Corp. 1985, 1991

An individual inode is not likely to be as inconsistent as the superblock. There is, however, a danger that the inodes for files that are in use when the system stops will be inconsistent.

The **fsck** command checks the list of inodes in sequence, starting with inode 1 and continuing to the last inode in the file system. **fsck** checks each inode for inconsistencies in format and type, link count, duplicate blocks, bad blocks, and inode size.

- Inode format and type inconsistencies

  Each inode contains a ***mode word***--a description of the type and state of the inode. There are eight inode types:

  -- Regular
  -- Directory
  -- Hidden directory
  -- Block special
  -- Character special
  -- Symbolic link
  -- First-in-first-out (FIFO, also called a named pipe)
  -- Multiplexed file.

  An inode whose type is not one of these eight does not have a valid type.

  There are three inode states:

  -- Allocated
  -- Unallocated
  -- Neither allocated nor unallocated.

  An inode that is neither allocated nor unallocated is not correctly formatted. The format of an inode can be damaged when incorrect data are written to the inode list (for example, as a result of a system failure).

- Inode link count inconsistencies

  Each inode contains a count of the total number of directory entries that refer to the inode (the ***reference count***). The **fsck** command verifies the reference count in each inode by scanning the entire directory structure, beginning with the root directory, and counting the actual number of links to that inode.

  If the stored link count is not zero and the actual link count is zero, the file is inaccessible. If the inaccessible file contains any data, **fsck** links the file into the **lost+found** directory on the file system. If the file contains no data, **fsck** deallocates the inode for that file.

  If the actual link count is neither zero nor equal to the stored link count, it is likely that a directory entry was added or removed without the inode being updated. The **fsck** command can replace the stored link count with the correct value.

- Inconsistencies from duplicate and invalid blocks

  Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by that inode. The **fsck**

¦ Copyright IBM Corp. 1985, 1991

command first determines whether each block number in the list is valid and then compares the block number against a list of allocated blocks. If it finds inconsistencies, **fsck** displays the names of all files that contain invalid or duplicate blocks.

Inconsistencies of this type usually occur when you create or extend files on a file system that has duplicate blocks in its free list. If the duplicate block contains data in both files, the damage is not severe. If the file system uses the duplicate block for a directory or an indirect block, however, severe inconsistencies in the file system structure may occur. Although such inconsistencies are serious, you can prevent them by routinely checking every file system with **fsck** before you mount it. (For example, do not use the **fsck -f** command in the system startup files. The **-f** [fast] flag saves time by not checking file systems that were cleanly unmounted the last time they were used. The extra precaution of running **fsck -p** from the system startup files reduces the chance that your system will mount a damaged file system.)

Generally, **fsck** corrects such inconsistencies by deleting the files with invalid or duplicate blocks, but not automatically--that is, you must respond to a prompt from **fsck** before it will destroy the files. Before allowing **fsck** to destroy such damaged files, you should examine the files to determine whether you can save any important data.

- Size inconsistencies

  In each inode is a ***size field*** that indicates the number of bytes in the file associated with the inode. The **fsck** command can determine whether the number of bytes is consistent with the number of blocks allocated to the file. In the case of directories, **fsck** also checks to see that the directory format is correct.

  If it finds size inconsistencies, **fsck** displays a warning message. Size errors are not serious, but **fsck** does not have sufficient information to correct the size. You can correct the problem by copying the data from the old file into a new file and then deleting the old file.

Indirect Block Inconsistencie

There are two types of indirect blocks: ***single-indirect*** and ***double-indirect***. A single-indirect block contains a list of some of the block numbers associated with an inode. Each entry in a single-indirect block is a data block number. A double-indirect block contains a list of single-indirect block numbers.

Indirect blocks are associated with a particular inode. Thus, inconsistencies in indirect blocks directly affect the associated inode. The **fsck** command can check for inconsistencies such as blocks that are already claimed by another inode and block numbers that are outside the range of the file system. The **fsck** command finds and corrects indirect block inconsistencies just as it does inconsistencies between the inode and data blocks.

Data Block Inconsistencie

¦ Copyright IBM Corp. 1985, 1991

There are two types of data blocks: plain and directory. Plain data blocks contain the information stored in a file. Because an ordinary file may contain any type of data, **fsck** cannot check the validity of the contents of a plain data block.

Directory data blocks contain directory entries. Each directory data block can be checked for inconsistencies involving directory inode numbers pointing to unallocated inodes, directory inode numbers greater than the number of inodes in the file system, incorrect directory inode numbers for the current and parent directories (. and .., respectively), and directories disconnected from the file system.

If a directory entry inode number points to an unallocated inode, then **fsck** tries to remove that directory entry. This condition might occur if the data block containing the directory entry were modified and written to the disk, but the system stopped before the inode could be updated.

If a directory entry inode number points beyond the end of the inode list, **fsck** tries to remove that directory entry. This condition occurs if incorrect data are written into a directory data block.

The directory inode number entry for **.** (period) should be the first entry in the directory data block. Its value should be equal to the inode number for the directory itself.

The directory inode number entry for **.** (period) should be the second entry in the directory data block. Its value should be equal to the inode number for the parent directory (or the inode number of the directory itself if the directory is the root directory).

The **fsck** command checks the internal connections of the file system in general. If it finds a file or directory not linked into the file system, **fsck** links the file or directory to the **lost+found** directory. This condition can be caused by inodes being written to disk and the system stopping before the corresponding directory data blocks are updated. Items linked to **lost+found** are identified by their inode number (for example, if **fsck** finds a file whose inode number is **1234**, it links the file to **/lost+found/1234**).

¦ Copyright IBM Corp. 1985, 1991

1.3.3.4 Repairing File Systems by Destroying Files

Under some conditions, the only way **fsck** can restore consistency to a file system is to clear inodes--that is, by destroying files. For example, if a block is allocated to two files, **fsck** clears both files. This section explains how to keep your loss of possibly useful data to a minimum.

Before letting **fsck** destroy a file, be sure you know which file is to be destroyed and what is wrong with the file. Most files damaged when a system stops unexpectedly are temporary files and they usually are of little value. However, if it is necessary to destroy a permanent file, you should try to salvage its contents; and if the file belongs to another system user, notify that person that the file is to be destroyed. Before **fsck** destroys a file, it lists all the names linked to that file, and prompts you to determine whether each link to the file can be removed. Only when all links are removed can **fsck** destroy the file.

Often, the name of a file indicates whether the file's contents are important. For example, temporary files may have names like **ctm5a.P5024** and often are in special directories like **/tmp** or **/usr/tmp**. Like temporary files, editor backup files (files with a **.bak** extension) and object modules (files with a .o extension) usually are not important. However, it is likely that a file named **boss/employees/raises** is quite important, and you should make every effort to salvage its contents. If you cannot tell from its name whether a file is important, try to salvage its contents.

If you know the i-number of a file, you can find the name of the file with the **ncheck** program. For example, if **fsck** indicates that two files, i-numbers 408 and 1212, on the **/u** file system have a duplicate block, you could find the names of those files by entering:

```
ncheck -i 408 1212 /u
```

For more information about the **ncheck** command, see **ncheck** in *AIX Operating System Commands Reference*.

When a file contains one or two bad or duplicate blocks, you probably can salvage most of its contents by copying it into a new file before destroying the original (damaged) file. To copy the contents of the file, mount the file system temporarily (and **readonly**, if possible) and use the **cp** command. It is safest to copy the file onto another (consistent) file system. If you want to copy the file within its original file system, first run **fsck** to make sure the free list is consistent.

You can destroy a damaged file in several different ways. If a file contains invalid or duplicate blocks, **fsck** removes all links to the file and then destroys the file by deallocating its inode (assuming that you respond **y** to the **fsck** prompts). You can destroy a file directly with the **clri** (clear inode) command. However, it is safer to let **fsck** destroy the file, since **fsck** automatically restores the file's data blocks to the free list and clears the corresponding directory entry; **clri** does not do this.

You should not use the **rm** (remove file) command to destroy files (in a damaged file system) for two reasons:

- You must mount the damaged file system to run **rm**. Mounting the damaged file system can compound the damage.

- **rm** frees all bad and duplicate blocks. Thus, if you removed the

¦ Copyright IBM Corp. 1985, 1991

damaged file with **rm**, the same type of damage is likely to occur the next time the system allocates those blocks.

Any time you destroy files, run **fsck** on the file system before using it again. The **fsck** command eliminates inconsistent directory references and makes sure that the free list is correct.

¦ Copyright IBM Corp. 1985, 1991

1.3.4 The Input/Output System

An I/O (input/output) system is a mechanism for transferring data among system devices (for example, when you print a file, the I/O system transfers the data from the file to the printer). The AIX I/O system is independent of the particular devices that produce output or receive input. If three devices (for example, a printer, a fixed disk, and a display station) can all take input in the same form, the I/O system makes no distinction among them. To understand how the I/O system works, you should first have a general understanding of device drivers and special files.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.3.4.1 Device Drivers and Special Files

A ***device driver*** is a program that makes the logical connection between the system and a device. The kernel contains a device driver for each type of device on the system. For example, if your system has two identical printers, you need only one device driver to run both of them; if you have two different types of printers, your system needs a separate device driver for each printer type.

Each device has a ***class***, a ***major device number***, and a ***minor device number***. There are two device classes, ***block*** and ***character***, and each class is associated with a group of device drivers. (For descriptions of the device classes, see "Block I/O System" in topic 1.3.4.2 and "Character I/O System" in topic 1.3.4.3.) The major device number indicates which driver the device uses. The minor device number passes additional information to the driver (for example, to specify which of several identical printers should be used). For more information on major device numbers and minor device numbers, see **master** and **system** in *AIX Operating System Technical Reference*.

Device characteristics are recorded in ***special files*** (ordinarily located in the **/dev** directory). A process can have read and write access to a special file just as it does to an ordinary file. However, the read and write operations on special files produce device I/O. To perform I/O operations, a process simply has to access the appropriate special file; it does not have to deal with the characteristics of the device itself.

¦ Copyright IBM Corp. 1985, 1991

1.3.4.2 Block I/O System

A model block I/O device consists of randomly addressed memory blocks of a uniform size. The blocks are addressed 0, 1, ... up to the size of the device (because the addresses start at 0, the highest-numbered address is the device size less one). A block device driver creates this model on a physical device.

Input and output for block devices is through a group of buffers. The system maintains a buffer pool for all block devices. When a process issues a request to read data from a block device, the system searches the buffer pool for the requested block. If the block is found in the buffer pool, the system makes it available to the requesting process without any physical I/O. If the requested block is not in the buffer pool, the system performs a physical I/O operation, replacing the least recently used block in the buffer pool with the requested block. The requested block is then made available to the process. Write operations to block devices are handled similarly.

Generally, block (buffered) I/O increases system efficiency. First, block I/O can greatly reduce the number of read and write operations required to perform I/O. Second, block I/O allows the operating system to schedule I/O for the most efficient system operation. In some instances, however, block I/O is a disadvantage. If the system stops unexpectedly, there may be physically incomplete I/O--changes made to the buffers that were not yet written to the disk.

¦ Copyright IBM Corp. 1985, 1991

1.3.4.3 Character I/O System

The character I/O system includes all devices that do not fit the block I/O model. Unlike block I/O requests, character I/O requests go directly to the device driver.

A model character device places characters directly onto a queue until the queue contains the maximum number of characters. As soon as there are any characters in the queue, the I/O operation starts, removing the characters from the queue one at a time and sending them to the device. When the number of characters falls to a specific level, the process passing characters to the queue refills the queue to its maximum capacity.

However, many devices in the character class are more accurately described as ***raw*** devices. Raw devices are included in the character class simply because they do not conform to the definition set forth for block I/O devices. For example, I/O for both fixed disks and streaming tapes occurs in units that have no specific relationship to characters as such. They are written in blocks which are simply some convenient multiple of whatever quantity of bytes the device driver handles. The application must specifically control block size in these instances, rather than having it handled automatically by the system.

¦ Copyright IBM Corp. 1985, 1991

1.3.5 Using the Queueing System

A ***queue*** is any sequence of things waiting for some sort of sequential service. In AIX, a queue refers to an ordered list of jobs waiting to be done by the computer. The main job of the AIX queueing system is to manage printer use, especially on systems that have more than one printer. However, you also can use the queueing system to control access to other system resources, as is explained under "Friendly and Unfriendly Backends" in topic 1.3.5.3.1. You can adapt the queueing system to your requirements easily. For example, by changing one word in a master configuration file, you can set a printer to take jobs in the order in which they are submitted or according to their sizes. Because your requirements of the queueing system may change periodically, you should understand how the system works and how to modify it.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.3.5.1 Parts of the Queueing System

The queueing system has four parts:

The print program
The **print** program accepts requests for print jobs and places them in the queue directory, **/usr/lpd/qdir**. For more information on the **print** command, see *Using the AIX Operating System*.

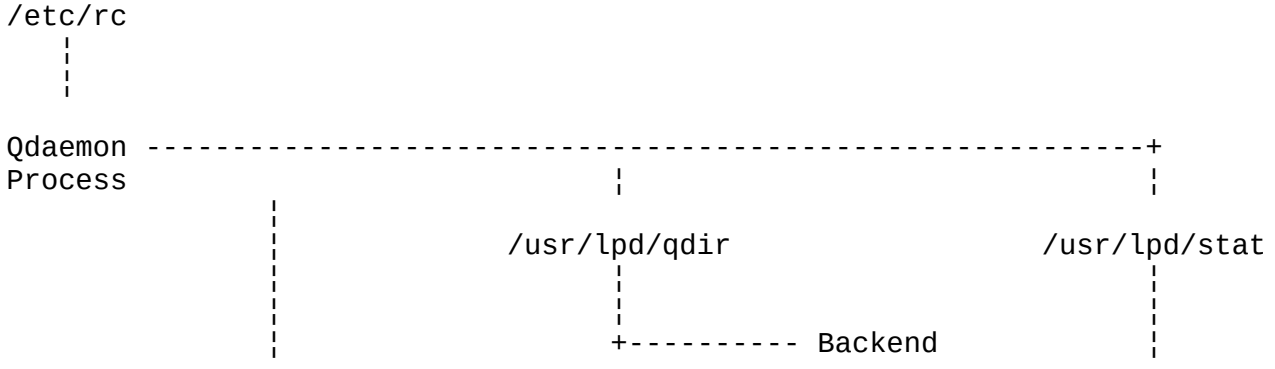
Note: The **print** program calls the **/usr/lpd/digest** command, running it invisibly at start up. For more information regarding **qdaemon** error messages, see "Changing the Configuration File," later in this chapter.

The qdaemon program
A ***daemon*** is a process that does not terminate until it is explicitly stopped. Daemons are often started during system initialization and terminated only when the machine is shut down. The **qdaemon** is a daemon that keeps track of print job requests (or other output) and the printers (or other devices) available to handle them. The **qdaemon** process maintains queues of outstanding requests and sends them to the proper device at the proper time. The **qdaemon** also records printer usage data for system accounting purposes.

Note: The **qdaemon** program calls the **/usr/lpd/digest** command, running it transparently at start up. For more information regarding **qdaemon** error messages, see the **Note** under "Changing the Configuration File," later in this chapter.

The backend program
A ***backend*** program sends output to a particular device (a printer, for example). You do not run a backend program directly; rather, **qdaemon** runs a backend program, sending it the names of the files to be printed and any print flags that you specify. The backend returns this and other information to the **qdaemon** through a status file in the **/usr/lpd/stat** directory. You can use the **print -q** command to interpret and display this status information; such as, how many pages were printed, what the status of the printer is, and how much of the print job is finished.

The configuration file
The ***configuration file***, **/etc/qconfig**, describes the configuration of available queues and devices. Both the **print** command and the **qdaemon** process refer to the configuration file.



¦ Copyright IBM Corp. 1985, 1991

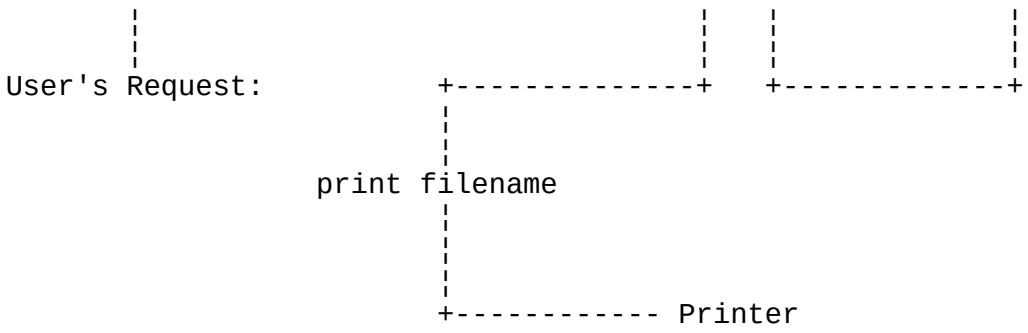


Figure 3-1. How the Queueing System Works

¦ Copyright IBM Corp. 1985, 1991

1.3.5.2 Queues and Devices

A queue is simply an ordered list of requests for a particular service. A ***device*** is something that can handle those requests one at a time. Each queue must be handled by at least one device, but it often may be handled by more than one.

Subtopics
1.3.5.2.1 Configuration
1.3.5.2.2 Queue and Device Names
1.3.5.2.3 Status Control
1.3.5.2.4 Job Order
1.3.5.2.5 Accounting

¦ Copyright IBM Corp. 1985, 1991

1.3.5.2.1 Configuration

Often you do not have a choice about which device services which queue. For example, if your system has only one printer, then that printer must service the print queue. There are times, however, when you should consider carefully the relationship between queues and devices. For example, if your system has two identical printers located next to each other, you probably will want both printers to service a single printer queue. You could accomplish this by placing the following stanzas in the configuration file **/etc/qconfig**:

```
lpr:
        argname = -l
        device = lpdev0,lpdev1

lpdev0:
        file = /dev/lp0
        backend = /usr/lpd/lp

lpdev1:
        file = /dev/lp1
        backend = /usr/lpd/lp
```

The first stanza describes the **lpr** queue. The **argname** line sets the flag used to request this queue, **-l**. A print command requesting this queue has the form **print -l filename**. The **device** line specifies which devices service the queue. The second and third stanzas describe the printers that service the queue. In each case, **file** is the special file associated with the printer, and **backend** is the file that contains the printer backend program. Once a print request reaches the top of the queue, the backend sends the file to the next available printer.

In another situation, you might want to define a different queue for each printer. For example, if the two identical printers are in different buildings, or if there is a difference in speed or quality between the two printers, you (or those who use your system) should be able to select a particular printer. In this case, the stanzas in **/etc/qconfig** could be:

```
lpa:
        argname = -la
        device = lp0

lp0:
        file = /dev/lp0
        backend = /usr/lpd/lp

lpb:
        argname = -lb
        device = lp1

lp1:
        file = /dev/lp1
        backend = /usr/lpd/lp
```

In this example, there are two queue stanzas, **lpa** and **lpb**. Following each queue stanza is a stanza describing the printer that services that queue. Use ASCII characters for the device name, backend routine name, and file name to ensure that users of different locales can access the printer being described.

¦ Copyright IBM Corp. 1985, 1991

1.3.5.2.2 Queue and Device Names

The names of queues and devices are similar, but there are important differences in how they work. Most system users do not use device names, since a **print** command is actually a request for a queue, not for a device.

Two lines in the **/etc/qconfig** file relate to the naming of queues:

queue name. This is the name of the stanza that describes the queue (like **lpa** and **lpb** in the previous example). The queueing system refers to a queue by its queue name and uses the queue name in any messages about the print job (including responses to the **print -q** command).

argname. This is the name you use with the **print** command to request a specific queue (like **-la** and **-lb** in the previous example).

Although the **queue name** and **argname** can be the same, you may find it helpful to make them different so that, as in the previous example, the **queue name** can look like the name of a printer and **argname** can look like a command flag.

When more than one printer services a queue, you can request a particular printer by naming it explicitly. (Ordinarily, your request would be serviced by the first device on that queue that became available.) To name a device explicitly, use the **argname** of the queue serviced by that device, followed by a colon (:) and the **device number** of the printer. (Devices are numbered independently for each queue, starting at 0; device numbers refer to devices in the order in which they appear in **/etc/qconfig**.) The following stanza (taken from an earlier example) defines a queue service by two printers:

```
  lpr:
          argname = -l
          device = lpdev0, lpdev1
```

The command **print -l:0** requests the printer **lpdev0**; the command **print -l:1** requests the printer **lpdev1**.

All references to devices by the **print** command and **qdaemon** are based upon the names contained in the **/etc/qconfig** file. Thus, within the guidelines given in "Changing the Configuration File" in topic 1.3.5.4, you can choose your own names for queues and devices.

¦ Copyright IBM Corp. 1985, 1991

1.3.5.2.3 Status Control

By setting the status of queues or printers to **up** or **down**, you can control access to them. A status setting of **up** or **down** refers to how the system treats the queue or device, not to the physical state of any system component.

A queue is available to both the **print** command and the **qdaemon** if its stanza in **/etc/qconfig** contains the line:

```
up = TRUE
```

or if the stanza does not contain the line **up =** (since the default value for **up** is **TRUE**). A queue is not available if its stanza contains the line:

```
up = FALSE
```

A queue that is **up** works normally. However, if a queue is **down**:

The **qdaemon** command does not send jobs to devices on the queue.

The **print** command does not accept requests for the queue.

The **print -q** command shows that all devices on that queue are **OFF**.

The only way to set the status of a queue **on** or **off** is to change the **/etc/qconfig** file. Generally, you should not set the status of a queue to **off** unless you know that printers that service the queue cannot be used for several days (for example, if they are to be taken down for repair).

You can set the status of a device more easily, using the **print -dd** (device down) or **print -du** (device up) command. To set status **down** for the first device servicing queue **l**, use the command:

```
print -l:0 -dd
```

To set status **up** for the same device, use the command:

```
print -l:0 -du
```

In both examples, the device name **-l:0** consists of the queue name (**-l**), a colon (**:**), and the device number (**0**).

The **qdaemon** does not send jobs to **down** devices, but **print** accepts requests for their queues. It also is possible for a backend to set the status of a device to **off**. Once a device is **down**, either because of a **print** command or the action of a backend, you must use the **print** command to set its status back to **up**.

The **qdaemon** records device status in a status file in the directory **/usr/lpd/stat**. When you start the system, **qdaemon** checks this status file before trying to run any device. Thus, you do not have to reset the status of devices each time you stop and restart your system.

¦ Copyright IBM Corp. 1985, 1991

1.3.5.2.4 Job Order

The order in which each printer services the job requests in its own queue is determined by two values placed in the **/etc/qconfig** file: ***discipline*** and ***priority***.

Discipline refers to the general rule used by the printer to decide the order in which to do the jobs submitted to it, such as first come first served and largest job first. The discipline is set individually for each printer. In a queue stanza, the line:

```
discipline = fcfs
```

sets the order to **first come first served**. If there is no **discipline** line in the queue stanza, requests are serviced in first-come-first served (**fcfs**) order. The line:

```
discipline = sjn
```

in a queue stanza sets the order to **shortest job next**.

The priority of a job is its general importance within its own discipline. Priority is assigned individually, one to each job. For example, if the discipline has been set to shortest job next and three print jobs of equal size are sent to the printer at once, the job that is given the highest priority is processed first. The ***priority number*** can be changed with the **print** command flag **-pr=n**. Print jobs with higher priority numbers (**n**) are handled before requests with lower priority numbers. Any system user can alter the priority of a print request with a command such as:

```
print -pr=20
```

The default priority number is 15. The maximum priority number for ordinary users is 20. The maximum priority number for privileged users (su and members of the system group) is 30.

¦ Copyright IBM Corp. 1985, 1991

1.3.5.2.5 Accounting

Accounting information consists of the user number, user name, and number of pages printed for each request. The backend program determines what constitutes a page. You can cause the queueing system to produce accounting records by placing the following line in the appropriate queue stanza in **/etc/qconfig**:

```
acctfile = /usr/adm/qacct
```

Use ASCII characters for the *acctfile* name to ensure that users of different locales can access it.

If a queue stanza does not contain an **acctfile** line or contains the line:

```
acctfile = FALSE
```

the queueing system does not produce accounting records for that queue.

The **qdaemon**, which produces the accounting records, does not create the accounting file. Thus, you can substitute another file name for **/usr/adm/qacct** if you choose.

You also can use a separate file to record the accounting data for each queue. Using separate accounting files can help you collect usage statistics for each queue (a useful ability if, for example, you are looking for ways to improve the efficiency of your queueing system) or allow you to assess different charges to the users of your system for the use of different printers.

For a full description of AIX facilities for accounting and monitoring system activity, see "Running System Accounting" in topic 5.3 and "Using the System Activity Package" in topic 5.4.

¦ Copyright IBM Corp. 1985, 1991

1.3.5.3 Backends

The **backend** line of the stanza for a device in the **/etc/qconfig** file determines what backend the **qdaemon** runs for that device. A line like:

```
backend = /usr/lpd/lp -fw = 132 -fl = 66
```

specifies the name of the backend program and includes any flags that are to be passed to that program. A line such as:

```
file = /dev/lp
```

causes **qdaemon** to send the backend's standard output to this file (in this case, the special file associated with a printer).

The line:

```
access = both
```

gives the backend both **read** and **write** access to the file. The line:

```
access = write
```

or the absence of an **access** line in the stanza gives the backend only **write** access to the file. If **file** does not have a value, **qdaemon** ignores the value of **access**.

The default value for **file** is **FALSE** and is only used by backends that access devices without help from **qdaemon**.

AIX provides some standard queueing backends, but you can also write your own. For information about the standard backends available and about writing a backend, see Appendix B of *AIX Operating System Technical Reference*.

Subtopics
1.3.5.3.1 Friendly and Unfriendly Backends
1.3.5.3.2 Burst Pages

¦ Copyright IBM Corp. 1985, 1991

1.3.5.3.1 Friendly and Unfriendly Backends

Ordinarily, the actions of the queueing system and a backend (as described so far) are neatly integrated. As long as a backend program follows certain conventions for communicating with **qdaemon** and the **print** command, it is called a ***friendly backend***. The queueing system also can handle ***unfriendly backends***--backends that do not follow these communication conventions. An unfriendly backend can be virtually any program; **qdaemon** requires no special communication or coordination with an unfriendly backend. The **friend** line in a queue stanza:

```
  friend = TRUE
```

or

```
  friend = FALSE
```

indicates whether a backend is friendly or unfriendly. If there is not a **friend** line in the stanza, the default value is **TRUE**.

You can use the queueing system to provide organized access not only to system devices (often printers), but to other system resources as well. Non-print applications of the queueing system typically use unfriendly backends. For example, if you create a log file (**/etc/logfile**) that is available to all users, only one user at a time should be allowed to edit the file. By adding the following stanzas to **/etc/qconfig**, you can require all access of the file to be through the queueing system (that is, one user at a time):

```
  log:
          argname = -lf
          device = logdev
          friend = FALSE

  logdev:
          file = /etc/logfile
          backend = /bin/cat
```

Use ASCII characters for the *logfile* name to ensure that users of different locales can access it.

To add a paragraph to the logfile, you now must first write the paragraph into a file (named, for example, **temp**) and then use the **print** command:

```
  print -lf temp
```

When the queueing system services your request for access to **/etc/logfile**, the **print** command adds the contents of **temp** to **/etc/logfile**. The (unfriendly) backend in this example is a standard command (**cat**). The **qdaemon** writes the contents of the temporary file to the end of **/etc/logfile**, not the beginning. Any system users who do not have **write** permission to **/etc/logfile** can change the file only by going through the queueing system.

You also can use the queueing system to require queued access to a program. For example, suppose a program on your system (**/bin/bigjob**) requires so much processing that, if two people run it at once, your entire system slows down seriously. You can make the program accessible only through the queueing system by adding the following stanzas to **/etc/qconfig**:

¦ Copyright IBM Corp. 1985, 1991

```
hog:
        argname = -b
        friend = FALSE
        device = bigjob

bigjob
        backend = /bin/bigjob
```

To run **bigjob**, enter **print -b name** (**name** is the name associated with the queued request and is used when you cancel the job, change its priority, or display its status, and is not necessarily the name of a file). The queueing system services requests for the **bigjob** program one at a time. Queued access is not appropriate for interactive programs.

¦ Copyright IBM Corp. 1985, 1991

1.3.5.3.2 Burst Pages

The term ***burst pages*** refers to printer output formatted so that each sheet of paper contains one page of output (on continuous form paper, the pages can be **burst** apart at the perforations). With the appropriate instructions in **/etc/qconfig**, most friendly backends print burst pages on their devices. For example, the following lines usually appear in the device stanza for a line printer:

```
header = always
trailer = group
feed = 3
```

The first line tells the backend to print a header page that gives the name of the job, who requested it, its date, and other information before every file printed. Other possibilities for the **header** line are:

```
header = group
```

which requests a header before every group of files for a single user, and:

```
header = never
```

(or the absence of a **header** line) which eliminates headers completely.

The **trailer** line in the device stanza tells the backend to print a trailer page, which gives the name of the user, only after a group of files is printed. (You also can use the **group** and **never** values with trailers.)

The **feed** line specifies that three **feed** (blank) pages are to be printed whenever the printer becomes idle. The feed operation pushes paper out of the printer so that it is easier to tear off. If the feed line in the stanza is:

```
feed = never
```

the backend is not invoked when the machine becomes idle; and, therefore, feed pages are not printed. The result of **feed = never** is usually the same as the result of:

```
feed = 0
```

except that the **0** value invokes the backend but tells it not to feed any pages. For some printers, **feed = never** and **feed = 0** produce different results. For example, some line printers make a loud noise if the paper fold rests under the spinning drum. The backend for such a printer might interpret **feed = 0** to mean that three blank lines should be fed, moving the fold off of the printer drum.

The **align** value is another way to control the number of extra pages between print jobs. The following line in the device stanza tells the backend to print an aligning form feed before any job requested after the printer is idle:

```
align = TRUE
```

This value applies only to printers using continuous form paper. As a rule, alignment on such printers is unnecessary since backends maintain proper paper alignment. However, on some continuous form printers, it is

¦ Copyright IBM Corp. 1985, 1991

possible for users to get the paper out of alignment when they remove their jobs. If this is a problem on your system, the alignment line will help. The absence of an **align** line is the equivalent of the line:

```
align = FALSE
```

¦ Copyright IBM Corp. 1985, 1991

1.3.5.4 Changing the Configuration File

Both **print** and **qdaemon** read **/etc/qconfig** when they start. **qdaemon** begins when you start the system; **print** starts each time someone requests a print job. Thus, if you change **/etc/qconfig**, **print** will read the new version of the configuration file the next time it runs, while **qdaemon** continues to rely on the original version of **/etc/qconfig**. To avoid problems due to this inconsistency, give the command:

```
print -rr
```

which causes **qdaemon** to reread **/etc/qconfig** and to reinitialize itself based on the new version of **/etc/qconfig** after all of its running jobs complete. (You also can cause **qdaemon** to reread the configuration file by shutting down and restarting the system, but the **print -rr** command usually is more convenient.)

You may receive a **qdaemon** error message that makes reference to **/usr/lpd/digest**. This **/usr/lpd/digest** is run invisibly by **print** and **qdaemon** when they start up, if the current **config** has not already been digested by the **qconfig** "digester." (The **qconfig** digester reads the **config** file, builds structures, then writes them out to a binary file.)

For example, a sample error message after executing **print-rr** is:

```
# print-rr
error in qconfig file /etc/qconfig, line 13
queue lp:  missing node field
PRINT (FATAL ERROR):  error from digester /usr/lpd/digest
Please notify system administrator
```

This message refers to the **qconfig** digester.

You should be alert to two common problems associated with installing a new **/etc/qconfig**:

If you install a new queue, someone may request that queue before yo reinitialize **qdaemon**.

If you remove an old queue, someone may request that queue at the las minute and find that it is no longer valid.

In either case, **qdaemon** logs an error message. You must determine whether the message refers to a new queue (in which case there is no longer a problem once you reinitialize **qdaemon**) or to an old queue (in which case the problem will exist until you remove the obsolete queue entries from **/usr/lpd/qdir**).

¦ Copyright IBM Corp. 1985, 1991

1.3.5.5 Keeping the qdaemon Running

Under normal circumstances, **qdaemon** starts when the system starts, runs until the system shuts down, and should require no attention from you. Under unusual circumstances, however, the **qdaemon** may stop running or be unable to perform its function. This section explains what you need to do under these conditions.

Any of the following conditions indicates that the **qdaemon** needs maintenance:

- **print** requests return the error message:

  cannot awaken qdaemon (request accepted anyway)

- **qdaemon** detects serious inconsistencies within itself and displays an error message

- **ps -ef** (the process status command that gives a full listing of all processes) does not show a process named **/etc/qdaemon** or **qdaemon**.

To restart the **qdaemon**, use the following command:

```
/etc/qdaemon
```

Generally, only privileged users can use this command. The new **qdaemon** goes through an initialization process.

If the **qdaemon** does not stay running, make sure that both **qdaemon** and **print** have the appropriate permissions. The user root owns both **qdaemon** and **print**. However, **qdaemon** and **print** must run as if they are owned by the user who starts them. The permission **s** sets the effective owner (user ID) of a process to that of the command
that is running it. Thus, the appropriate permissions for these two commands are:

qdaemon -r-sr-sr--

To check these permissions, enter **ls -l /etc/qdaemon**.

If the permissions need to be reset, enter: **chmod 6554 /etc/qdaemon**. (You must have superuser authority to reset these permissions.)

print -r-sr-sr-x

To check these permissions, enter **ls -l /bin/print**.

If the permissions need to be reset, enter: **chmod 6555 /bin/print**. (You must have superuser privileges to reset these permissions.)

If you continue to have problems with **qdaemon**, you can reinitialize the entire queueing system by going through the following procedure:

1. If the **qdaemon** is running (use the **ps -ef** command to find out), end it with the **kill** command (**kill process id number**).

2. If any backends are running, use the **kill** command to stop them.

¦ Copyright IBM Corp. 1985, 1991

3. Delete the contents of the following directories:

   /usr/lpd/stat (actually **/<LOCAL>/lpd/stat**)

   /usr/lpd/qdir (actually **/<LOCAL>/lpd/qdir**)

   /tmp/copies (actually **/<LOCAL>/tmp/copies**)

4. Restart the **qdaemon** with the command **/etc/qdaemon**.

¦ Copyright IBM Corp. 1985, 1991

1.3.6 Handling System Errors

Whenever some part of your AIX system does not function properly, the system has an ***error condition***. Some error conditions are minor as when, for example, you make a typing mistake and enter a command that the system does not recognize. Other error conditions can be so severe that they cause your system to halt. The first part of this section summarizes what you should do if your system halts unexpectedly. The remainder of this section describes the AIX facilities for collecting, analyzing, and reporting system errors:

Error logging service

The **dump** command

The **trace** command.

See *AIX/370 Diagnosis Guide* for more information.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.3.6.1 Recovering from Unexpected System Failures

Under normal conditions, before you turn off the power to the system, you use the **shutdown** command to bring all system processes to an orderly halt. Among other things, the **shutdown** command assures that all files are updated and stops any running system processes. In the event of a power failure, the power to the system being turned off without running **shutdown**, or unexpected system failures, your file system may have inconsistencies. You may be able to restore your system to normal operation by simply supplying power again. As part of the normal initialization procedure, the system runs the **fsck** program. If the file system sustained only minor damage, the form of **fsck** that runs automatically may be adequate to repair the damage. If not, a system message indicates what action to take next.

Warning: LAN Channel Stations such as IBM 8232, or 9370 Integrated LAN adapters must be reset manually before bringing up VM and AIX after a system failure. Manual reset is not necessary when the system is shut down normally.

¦ Copyright IBM Corp. 1985, 1991

1.3.6.2 Error Logging, Analysis, and Reporting

The following components provide the AIX error logging services:

event log
: Two files, **/usr/adm/ras/errfile.0** and **/usr/adm/ras/errfile.1**, of a predetermined size, which contain entries about system errors and other system events. When one file is full, the system begins to log events in the second file. When the second file is full, the system begins to overwrite data in the first file with new event entries.

error analysis facility
: A program that provides information about the probable cause of errors.

errdemon
: The ***error daemon process***, which collects error records from a buffer in memory, calls the error analysis facility, and writes error records, together with analysis information, to the event log. When it writes certain types of error records to the event log, the error daemon sends an alert message to the display.

error device driver
: The device driver, **/dev/err**, used by **errdemon** to write error records to the event log.

errpt
: A command that produces a report of logged errors from the data stored in the event log. **errpt** can produce summary or detailed reports, and you can specify what type of error records the report includes or the time span that the report covers. Generally, this is the only part of the error logging system that you work with directly.

When you start the AIX Operating System, the error logging services start automatically. When you stop your system, the error logging services stop automatically.

To generate a report of entries in the event log, including errors, enter a command of the form:

```
errpt flags filenames
```

where **filenames** specifies the event log files. The **errpt** command accepts the following flags:

-sdate Ignores all records logged earlier than **date**. **date** must be in the form **mmddhhmmyy**.

-edate Ignores all records logged later than **date**.

-a Produces a detailed report on a single record.

-nnodename Includes only error entries from **nodename** in the report.

-dlist Limits the report to the types of error records specified in **list**. For the values that can appear in **list**, see **errpt** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

There are two ways to limit the information in the error log report to that which you find useful:

Periodically remove the error log files (you must have superuse authority):

1. Enter **errstop** to stop the error daemon

2. Enter **del /usr/adm/ras/errfile.0**

3. Enter **del /usr/adm/ras/errfile.1**

4. Enter **/usr/lib/errdemon** to restart the error daemon.

Narrow the error log report by selecting the **errpt** options to limit the error report generator to the type of information you find useful.

¦ Copyright IBM Corp. 1985, 1991

1.3.6.3 AIX Dump Facility

The AIX dump facility records the computer's state at the time of a failure. The data collected by this dump facility is intended to help you or the person servicing your system determine the cause of the failure. The AIX dump facility needs to write its output to a configured dump device/area. Therefore, before a dump can be performed, you must designate where the dump should be stored. This storage area can be either minidisk or diskette.

See the *AIX/370 Diagnosis Guide* for more information on the AIX/370 dump facility.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

1.3.6.3.1 Designating a Minidisk as the Dump Area

When you install the operating system, you are given the option of creating a dump minidisk. If you did not create a minidisk during installation but want to use a minidisk for storing dumps, you can create a minidisk with the **minidisks** command. See the AIX installation manual for your workstation for information about the **minidisks** command.

¦ Copyright IBM Corp. 1985, 1991

1.3.6.3.2 Designating Diskette as the Dump Area

To designate a diskette as the storage area for operating system dumps, do the following:

1. Edit the **/etc/system** file. Add the following line to the **sysparms** stanza:

   ```
   dump=fd0
   ```

2. Build a new kernel. At the # prompt, enter:

   ```
   /usr/sys/newkernel
   ```

   Note: If **/usr/sys** is in your path statement, you can build a new kernel by just entering the **newkernel** command. For more information, see *Installing and Customizing the AIX PS/2 Operating System,* Chapter 5.

3. Reboot the operating system to activate the new kernel.

¦ Copyright IBM Corp. 1985, 1991

1.3.6.3.3 Starting a Dump

If the system detects an operating system failure, it may start an operating system dump automatically. If the system, or part of the system, appears to be hung up (but a dump has not been started automatically), you can start an operating system dump on a PS/2 or PS/55 by pressing **Ctrl-Alt-Pad 7**. If you are using a minidisk as the storage device, the dump is automatically placed on minidisk. If you are using diskette as the storage area, the system writes all of your system's real memory to diskette and requires multiple diskettes.

You should have enough formatted diskettes (approximately one for each megabyte of memory) available for your system, for example:

System Memory	**Number of Diskettes**
2M bytes	2
3M bytes	3
4M bytes	3

If the dump is being stored on diskette, the system prompts you to insert a formatted diskette. You may find it useful to number the diskettes in the sequence you will use them; such as 1 for the first diskette, 2 for the second diskette, and so on.

The operating system restarts automatically after a system dump.

¦ Copyright IBM Corp. 1985, 1991

1.3.6.3.4 Analyzing a Dump

To determine the cause of the system failure, use the **crash** utility to examine the operating system dump. If you are a very experienced user, you may find the **crash** command useful. See **crash** in *AIX Operating System Commands Reference* for more information about this command.

Analyzing a Minidisk Dump: Before you analyze a dump to a minidisk, you may want to save the dump. If you do not save the dump and another operating system dump occurs, the old dump on minidisk is overwritten by the new dump. To save a minidisk dump, use the **dd** command to copy the dump from the dump device (**/dev/dump**) into a file or onto diskettes. For example, to save a six-megabyte dump, you might enter the following:

```
dd if=/dev/dump of=/u/dumps/dump.xxxx bs=4k count=1536
```

where **xxxx** is a unique suffix that you use to identify that particular dump.

Analyzing a Dump to Diskette: Before you can use the **crash** command to analyze a dump on diskette, you must first use the **dd** command to copy the dump into a file. For example, to copy a three-megabyte dump into a file, you would use a procedure like the following:

1. Insert first diskette and enter:

   ```
   dd if=/dev/rfd0 bs=40k > /u/dumps/dump.xxxx
   ```

2. Insert second diskette and enter:

   ```
   dd if=/dev/rfd0 bs=40k >> /u/dumps/dump.xxxx
   ```

3. Insert third diskette and enter:

   ```
   dd if=/dev/rfd0 bs=40k >> /u/dumps/dump.xxxx
   ```

   where **xxxx** is a unique suffix that you use to identify that particular dump.

Use ASCII characters to ensure communication by users of different locales.

¦ Copyright IBM Corp. 1985, 1991

1.3.6.4 trace Services

The **trace** command monitors system events while a program runs. You can use **trace** data to analyze system performance and to detect problems with programs. The **trcrpt** command processes the data collected by **trace** into a usable form and writes it into a file.

The **trace** command monitors the system events specified in a ***trace profile***. The default profile is the **/etc/trcprofile** file. To change the events that **trace** monitors, you can either modify **/etc/profile** (with an editor) or create alternate **trace** profiles.

To use **trace**, enter **trace** and then start the program with the activity you want to trace. To stop **trace**, enter **trcstop**.

For more information about using these commands, see **trace** and **trcrpt** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

1.3.7 Generating a New Kernel

Customizing is the process of adapting the AIX Operating System for a particular set of requirements. When you customize a system, you define for the kernel such things as:

Devices attached to the system (for example, printers, displa stations, and fixed disks)

Limits on system resources (for example, the maximum number o processes that can run at the same time or the maximum number of file systems that can be used at the same time).

The system is initially customized when it is installed; much of the customizing happens automatically during the installation process described in the AIX installation manual. You can customize some aspects of the system while it is in use (for example, with the **devices** command described in the System/370 and PS/2 AIX installation manuals).

For more information regarding customizing the kernel, see the *AIX Operating System Installing and Customizing the Operating System* (System/370 or PS/2).

Subtopics
1.3.7.1 TCF Requirements

¦ Copyright IBM Corp. 1985, 1991

1.3.7.1 TCF Requirements

If your system has TCF installed, you first need to ensure that your site has joined the TCF cluster and that the primary cluster site is also present before running the newkernel command. The primary site must be present in order to read the system libraries and create the new kernel.

¦ Copyright IBM Corp. 1985, 1991

4.0 Chapter 4. Additional System Management Topics

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.1 Contents

¦ Copyright IBM Corp. 1985, 1991

4.2 About This Chapter

This section covers an assortment of topics that you may find useful in managing the AIX Operating System (such as, installing applications, setting the system date, running commands at pre-set times, and setting display station characteristics). You may find it useful to browse through this chapter looking for features that will best adapt the system to suit your needs. If your system supports several users, you also should acquaint yourself with the AIX facilities for accounting and monitoring system activity. "Running System Accounting" in topic 5.3 describes the AIX accounting facilities and how to use them. "Using the System Activity Package" in topic 5.4 describes the facilities for monitoring system activity.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System,* and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

4.3 Introduction: Theory of LPP Service Process

This section describes commands and processes that allow you to apply service to the system or cluster you administer. Be sure to read this entire chapter, and the peripheral information (like "Information File" that comes with the LPP) before you perform any process.

The service process exists to install corrected and updated software on your cluster. In order to give you some control over service, this process will allow you to also remove updates with which you are unsatisfied. These updates can exist on your cluster in any of four different states: applied, commit, uncommit and reject.

For additional information regarding LPP's, refer to *AIX/370 Planning Guide, Installing and Customizing the AIX/370 Operating System, Installing and Customizing the PS/2 Operating System,* and *AIX Programming Tools and Interfaces*.

Note: For more information on the files and scripts involved in an update, refer to the chapter entitled "Installing and Updating an LPP" in *AIX Programming Tools and Interfaces*.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.3.1 Servicing System Components and Field Serviceable Units (FSU)

The service process updates a component from one Version Release Level (VRL) to another and provides systems administration tools and changed files containing fixes to known problems.

This AIX operating system release allows you to service the system in FSUs which can be applied via **updatep**. An FSU may correspond to an LPP or an LPP may be made up of several FSUs. Using FSUs you may update a portion of an LPP without disturbing other parts. Service will be distributed in this form for this release.

The following tables show which LPPs have a single FSU and those which include more than one FSU. These tables do not contain the separate LPP installables which were separately serviced prior to this AIX Release. However, new installables for this release are included.

Table 4-1. AIX System/370 Table of LPPs and FSUs

LPP/FSU	Arch	Type	Installable	Updateable	Description
aix370	370	LPP	X		AIX System/370
admn0	370	FSU		X	Administrative Support
adst0	370	FSU		X	Adv. Dev. Supp. Tools
bnuu0	370	FSU		X	Basic Net. Utilities
english0	370	FSU		X	I370 English Catalogs
extp0	370	FSU		X	Extended User Support
games0	370	FSU		X	Games
graph0	370	FSU		X	Graphs
ined0	370	FSU		X	INed
kanji0	370	FSU		X	I370 Japanese Kanji Catalogs
learn0	370	FSU		X	Learn
man0	370	FSU		X	Manual Pages
mhmh0	370	FSU		X	Message Handler
oscmds0	370	FSU		X	Command Updates
oscrit0	370	FSU		X	Critical Updates
oskern0	370	FSU		X	Kernel Updates

Copyright IBM Corp. 1985, 1991

osserv0	370	FSU		X	Serv. Tool Updates
pci0	370	FSU		X	PC Interface
samp0	370	FSU		X	Sample Programs
sccs0	370	FSU		X	Source Code Control System
send0	370	FSU		X	Sendmail
tcf0	370	FSU		X	Transparent Computing Facility
tcpip0	370	FSU		X	Transmission Ctrl Protocol/Internet Protocol
tfs0	370	FSU		X	Text Formatting System
x110	370	FSU		X	X-Windows
x11j0	370	FSU		X	X-Windows - Japanese Supplement
x11smp10	370	FSU		X	X Windows Samples
inmn	370	LPP	X		INmail
nfs0	370	LPP	X		Network File System

Table 4-2. AIX PS/2 Table of LPPs and FSUs

LPP/FSU	**Arch**	**Type**	**Installable**	**Updateable**	**Description**
adt	386	LPP	X	X	Advanced Dev. Toolkit
adst	386	LPP	X	X	Advanced Dev. Support Tools
sccs	386	LPP	X	X	Source Code Control System
ate	386	LPP	X	X	Administrative

Copyright IBM Corp. 1985, 1991

					Support
BOS	386	LPP	X	X	Basic Operating System
opsys	386	LPP	X		Operating System
oscmds	386	FSU		X	Command Updates
oscrit	386	FSU		X	Critical Updates
oskern	386	FSU		X	Kernel Updates
osserv	386	FSU		X	Serv. Tool Updates
cdrom	386	LPP	X	X	CD-ROM Support
english	386	LPP	X	X	English Catalogs
kanji	386	LPP	X	X	Japanese Kanji Catalogs
C compiler	386	LPP/FSU			Compiler
ext	386	LPP	X	X	Extensions
adm	386	LPP	X	X	Administration Support
bnuu	386	LPP	X	X	Basic Network Utilities
extp	386	LPP	X	X	Extended Programming Support
games	386	LPP	X	X	Games
mhmh	386	LPP	X	X	Message Handler
samp	386	LPP	X	X	Sample Programs
send	386	LPP	X	X	Sendmail Program Subset
gsl	386	LPP	X	X	Graphics Support

¦ Copyright IBM Corp. 1985, 1991

					Library
ined	386	LPP	X	X	INed
inmn386	386	LPP	X	X	INmail
learn	386	LPP	X	X	Learn
man	386	LPP	X	X	Manual Pages
Merge	386	LPP	X	X	DOS Merge
Extended C compiler	386	LPP/FSU			Compiler
Motif	386	LPP	X	X	Motif
nfs	386	LPP	X	X	Network File System
pci	386	LPP	X	X	PC Interface
tcf	386	LPP	X	X	Transparent Computing Facility
tfs	386	LPP	X	X	Text Formatting System
tcpip	386	LPP	X	X	Transmission Ctrl. Protocol/Internet Protocol
x11	386	LPP	X	X	X Window System
x11j	386	LPP	X	X	X Windows Japanese Supplement
x11smpl	386	LPP	X		X Windows Samples
x25	386	LPP	X		X.25 Program
xdt	386	LPP	X		AIX Windows Desktop
X WINDOWS	386	LPP/FSU	X	X	X Windows Program
japanese	386	FSU		X	Japanese Language
IAA	386	FSU		X	Image Adaptor/A (Display ext.)

Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

4.3.2 Applying Service

When you install service, you "apply" updates. This option allows you to update your cluster with corrected programs for the LPP you have installed on your system. The **updatep** program saves old versions of files and programs so that you may "reject" service with which you are not satisfied. When you are installing or servicing your system, you are, in effect, building up layers of new software, one on top of the other. This has important implications. Service is applied in this system with a Last-In-First-Out strategy. In order to remove any particular layer of installation or service, you will have to remove all other layers of service that was previously installed.

Note: There is no mechanism to remove an LPP after you have installed it, and you cannot remove service beyond the last LPP installed unless you take special precautions during the installation. For a description of these special precautions see "Rejecting Service Past an LPP Installation" in topic 4.3.11.

Also note that you can apply service for more than one LPP at a time during an **updatep** session. In order to maintain consistency across updates to multiple LPPs, the service process bundles together service applied to more than one LPP. When service is applied together for multiple LPPs, that service is manipulated as a single entity, and must then be committed, uncommitted, or rejected together. You may reuse your update media to apply other updates, or if you chose to do so, apply service to one LPP at a time. This uses a good deal of disk space to save the older versions of the system; therefore, you should apply updates together.

Note: You can have only one applied update on your system (or cluster) at a time.

If problems are found during the update process that level of service is automatically rejected and the system returns to its original state.

¦ Copyright IBM Corp. 1985, 1991

4.3.3 Committing Service

Once you are satisfied with an update you have applied, you can "commit" that update to put your service into a more permanent state. The previous levels of software are now moved to a save-stack frame and may be retrieved later, or archived to backup media. With all your service in this state, you may now apply other updates to the system.

¦ Copyright IBM Corp. 1985, 1991

4.3.4 Uncommitting Service

Should you decide that you want to restore previous levels of service that you have committed, you must "uncommit" the current service level. This option restores updates back to the applied state, where they may be rejected. If you have used the **cleanup** command to archive save-stack frames to tape, you must make sure you have restored the frames that you are about to uncommit. The uncommit option will not prompt you for them.

¦ Copyright IBM Corp. 1985, 1991

4.3.5 Rejecting Service

Once you have restored an update back to the applied state using the **uncommit** option, you may reject that service. This removes the new update from your cluster and replaces it with the previous commands and files as they existed before you tried to install the update.

¦ Copyright IBM Corp. 1985, 1991

4.3.6 Updating Locals

The above commands apply service in general to an entire cluster. The new versions of commands are automatically propagated to the proper sites. This automatic mechanism does not work for updates applied to the locals for any sites in your cluster. In order to update your local (with any of the above states of service) you need to run the **qproc** command. The **qproc** command runs scripts that apply or reject service to a single site. In general, you must run **qproc** on all sites in your cluster every time you do any of the service functions above. Running **qproc** on a system that needs no service applied does no harm. You may, at your discretion, wait to run **qproc** until you have an entire session of updates to be applied and committed. The **qproc** command then applies as much service as it can. Note that **qproc** is the mechanism that may install new kernels on your systems, and may request that you reboot the system after service is applied.

¦ Copyright IBM Corp. 1985, 1991

4.3.7 Managing Your Service Process

While this update process can be run from any site on your network, it is suggested that you service your cluster from its primary site. This will reduce network traffic and reduce the amount of time the update will take. While AIX supports automatic propagation of new and updated files within the cluster, it is recommended that you only update your cluster when it is idle. This is to make sure a minimal set of processes within the cluster are busy. See the explanation of the ETXTBSY error in the **/usr/include/sys/errno.h** file. More information on error conditions can be found in Appendix A of the *AIX Operating System Technical Reference*.

The **restore** command handles the processes that are busy during update. But since these files cannot be removed while they are still active, the **restore** command renames them by appending the file called **original_filename.deletemeccccc** where **ccccc** are unique identifiers assigned by the system. These files may be removed subsequent to the update process.

For each LPP update, there may be a file included on the update that contains important user information. You should make sure that you read these files as they may contain special instructions you may be required to perform before or during this update. They will also contain useful information about the update, its limitations, and other related material. Although most updates to a cluster are independent, you should strive to keep your system at the latest service levels. In particular, due to the close interaction of these LPPs, the updates to aix370, opsys, and TCP/IP should always be kept at the same levels of service.

¦ Copyright IBM Corp. 1985, 1991

4.3.8 Managing Your Disk Space

The update process is disk-space intensive. The documentation that you receive with each update will tell you how much space any particular update will require. This is the amount of space the update will require to keep the previous level of service available to you in case you wish to reject this particular update. In order to recover this disk space, the **cleanup** command is provided. This command archives updates to tape or diskette media. Be sure to record the frame number supplied to you during the commit step of the update process. For more information on the **cleanup** command, refer to the *AIX Operating System Commands Reference*.

If you wish to keep all the committed service available on your system, you should make **/usr/lpp.save** a mounted filesystem. The **/usr/lpp.save** is the directory where all committed service is stored. This filesystem should be mounted on the primary site of your cluster. Service that is only in the applied state is stored in **/usr/lpp/lpp-name** for each LPP. Although this directory could also be made a mounted filesystem, many LPPs store important files there, and so you may be affecting the performance of those LPPs by removing them from the replicated root filesystem.

¦ Copyright IBM Corp. 1985, 1991

4.3.9 User Configurable Files

If you have made any updates to your system that have not been controlled by the AIX service process, you should make sure that you have backed up those changes before any service session. These include any changes that you have made to system libraries, or configuration files, or software that was not installed as an LPP.

¦ Copyright IBM Corp. 1985, 1991

4.3.10 Prerequisites

When updating your system, certain updates may have dependencies on other LPPs being updated at the same or known previous levels of service, although for the most part, this will be transparent to the user of the **updatep** command. You may become aware of this dependency if you do not have the prerequisite LPP installed. If this is the case, you are allowed to install ignoring prerequisites. For more information, see the "User Information" files associated with those updates.

¦ Copyright IBM Corp. 1985, 1991

4.3.11 Rejecting Service Past an LPP Installation

You should install your system completely before ever doing service on it. You should also have all LPPs installed in your cluster before trying to apply service. If this is not possible, you should back up your root filesystem for your primary site, and the locals for all your other sites in a cluster before installing any new LPP. In addition, if you intend to remove old service beyond the layer that was a new LPP installation, you should back up the system at that point also. If you wish to reject service beyond an update, you can do so by restoring the primary site to its original level before the LPP was installed. This must be done with the primary site removed from the network by using the **clusterstop** command. At this point, you can then reject any other service to the level you desired. You may also apply and commit any later service you wish to reapply. Use the latest backup of the system to retain any customized files.

Note: In order to complete this re-installation of your system, you need to perform the processes to make all sites in your network repropagate their root filesystems.

¦ Copyright IBM Corp. 1985, 1991

4.3.12 Statement of Serviceability

Software service for AIX (PS/2 or 370) is provided on media used by the program **updatep**. The **updatep** program also provides facilities to remove the service, restoring the system to its state prior to the installation of an update. Using this facility, an update can be removed by the system administrator if necessary.

If you install an AIX system and some number of service updates using **updatep**, special precautions must be taken if a new LPP (such as NFS) is going to be installed. A complete backup of the system filesystems should be made before installing an LPP so the system can be restored (if needed) to the state it was in before the LPP install. This system backup takes the place of the LPP **uninstall** capability. Service updates applied to the system after the LPP install can be removed using **updatep**.

If you need to remove service corrections installed *before* a new LPP was added to the system, you will need to back up key system files, restore the system backup saved before installation of the LPP, and then copy the saved key system files onto the restored system. All secondary sites in the cluster need to be re-propagated from the restored primary site at this point. You can now proceed with the **updatep** removal of service updates.

You should consider installing all of the LPPs you have licensed/ordered when you install the base system, even if you have no immediate desire to use the LPP. You can use your system and tailor the LPPs at your convenience. See the section below, "Applying Updates with the updatep Command" in topic 4.4 for more information.

¦ Copyright IBM Corp. 1985, 1991

4.4 Applying Updates with the updatep Command

You may have received an update tape or diskette with the other media in the AIX Operating System. And, in the future, you may periodically receive one or more media containing updates to the LPPs. The updates may consist of enhancements or changes to one or more programs provided on Program Temporary Fix (PTF) or interim Program Update Tapes (PUT) media.

Updated files are applied to a file system by using the **updatep** command. The **updatep** command lets you apply and commit numerous licensed programs with only one kernel rebuild. This means that you can complete updates faster and easier, but this method does not allow you to return to the previous level. These updates are stored along with the original system files which can be recovered later, if necessary. The original system files are saved in **/usr/lpp/***lpp_name***/inst_updt.save**.

Notes:

You may want to apply the updates to use the enhancements and changes You should install all AIX programs that you plan to update before you apply the update media.

You can only have one update in the applied state at any time. If yo have already applied an update to the system, that update can be committed by using the **-c** flag.

The update process is disk-space intensive. Before applying a update, you should determine whether or not you have enough space on your system to continue. The documentation provided with the update describes space requirements.

When the kernel is rebuilt, the old versions of the kernel are no deleted. It may be necessary to remove old versions if you run low on system space.

When you apply an update, you update the existing program on your system. You can use the updated program for some time to determine whether it is more acceptable to you than the previous level of the program. If you decide that the updated program is satisfactory, you can decide to commit the update. When you commit an update, you remove the previous level of the program from your system and permanently update the existing program with whatever enhancements and changes may be on the updated program. If you decide that the updated program is not satisfactory, you can decide to reject the update. In this case, you decide that you will bring your program back to its format state, before you applied the update.

Normally your cluster or system should be in a quiescent state while installing updates. You should expect some degradation in performance while the update process is occurring. Note that each of the files installed by the update are being automatically propagated to each site in your cluster. Some updates will require the local of each site to be updated (special functions, new kernels, etc). Although the update mechanism will allow you to start these local updates, you will have to start this process for any site in your cluster that is not currently up.

Consult the documentation packaged with your program to find out if it requires a quiescent system. If it does, restart the system, keeping it in single-user mode. You must be sure no processes are running on the system while performing the update.

¦ Copyright IBM Corp. 1985, 1991

Note: Log in as root or be the superuser.

To apply an update to your system, follow these steps:

1. Mount the distribution tape or insert your update diskette. Ensure that both the site with the file system **/usr/lpp** and the primary site are up.

2. At the AIX prompt (#), enter:

   ```
   updatep -a -d /dev/xxx
   ```

   The **xxx** is the input device name. Messages similar to the following are displayed as the system begins restoring files:

```
+-------------------------------------------------------------------------
¦
¦ File System Restore Utility
¦ x ./usr/sys/inst_updt/special
¦ x ./usr/sys/inst_updt/control
¦ x ./lpp_name
¦     files restored: 3
¦ 048-116 The system has completed the update validation of the programs.
¦  The system is checking to see whether any programs require special
¦  functions or must be updated together. Please wait.
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
+-------------------------------------------------------------------------
```

   Next the system displays the "Apply Updates Menu," which lists all licensed programs that have updates. It is similar to the following screen. From this menu, you can select a single licensed program, multiple licensed programs or all licensed programs you wish to update.

```
+-------------------------------------------------------------------------
¦
¦ 048-082 Apply Updates Menu
¦
¦        No Special Function Required
¦
¦
¦         ID UPDATE
¦
```

¦ Copyright IBM Corp. 1985, 1991

```
 1 PROGRAM1 Program

To cancel the updatep command enter "quit".

You may select one or more numbers from the list.

Type the ID numbers of the programs you wish to apply separated
by spaces (example: 1 3) and press Enter.

--->
```

On this menu, you need to select the number corresponding to the program(s) to update. If more than one program is to be updated, enter the numbers for the programs separated by spaces.

Notes:

a. The options listed on this menu depend on the programs currently installed on the system.

b. If you select more then one update at this point, those updates will be applied, committed, uncommitted, and rejected as a whole.

3. Enter the number corresponding to the program to be updated and press **ENTER**.

After you press **Enter**, the system begins to apply the updates. Messages similar to the following are displayed:

```
048-089 The system is now starting to apply the updates for program
        "PROGRAM1 Program".
File System Restore Utility
x ./usr/lpp/PROGRAM/lpp.loc/info
x ./usr/lpp/PROGRAM/lpp.loc/lpp.hist
x ./usr/lpp/PROGRAM/lpp.loc/inst_updt.loc
    files restored: 3
File System Restore Utility
x ./usr/lpp/PROGRAM/inst_updt/arp
    files restored: 1
```

¦ Copyright IBM Corp. 1985, 1991

Note: The exact content of these messages depends on the programs chosen to be updated.

After these messages, the system displays copyright information.

4. If the update being applied is an upgrade, you see messages indicating that the system is saving files written over by the "File System Restore Utility." You can retrieve these files later, if necessary.

```
File System Restore Utility
x ./usr/lpp/PROGRAM/lpp.loc/files/local/PROGRAM1
x ./usr/lpp/PROGRAM/lpp.loc/files/local/PROGRAM3
x ./usr/lpp/PROGRAM/lpp.loc/files/local/PROGRAM4
    files restored: 3
   -    File "/usr/bin/PROGRAM/PROGRAMx" is being saved.
   -    File "/usr/bin/PROGRAM1" is being saved.
   -    File "/usr/bin/PROGRAM3" is being saved.
   -    File "/usr/bin/PROGRAM4" is being saved.
   -    File "/usr/bin/PROGRAM5" is being saved.
File System Restore Utility
Please mount volume 1 on /dev/xxx
```

5. After you press **Enter**, you see messages similar to the menu above.

6. If there were updates to local file systems, the following messages are displayed:

```
x ./usr/bin/PROGRAM1
x ./usr/bin/PROGRAM3
x ./usr/bin/PROGRAM4
x ./usr/bin/PROGRAM5
x ./usr/bin/PROGRAM/PROGRAMx
```

¦ Copyright IBM Corp. 1985, 1991

```
¦     files restored: 5
¦ File System Backup Utility
¦ Done     at Fri Jun 17 14:48:05 1988
¦ 40 blocks on 1 volume(s)
¦ 048-088 All requested update processing is completed.
¦ The following sites are available:
¦ SITE1 SITE2 SITE3 SITE4 SITE5
¦ Please enter site name followed by the desired execution time.
¦ You can enter "all" to choose all sites and "now"
¦ for immediate execution.
¦ (STOP for break)
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
+------------------------------------------------------------------------
```

All the necessary files have been updated on the replicated root file system.

7. Specify the site to which you are updating files and the time you want the system to update the files on each of the local sites by entering:

   ```
   sitename time
   ```

   The **sitename** is the name of the site on which the local files that are to be updated reside. The **time** is the time when the update procedure is to begin on the specified site. For example, to update a single site called **SITE1** at 7:00 p.m., enter the following:

   ```
   SITE1 7:00 p.m.
   ```

   To update local files on all the listed sites, enter **all**. To begin the update immediately, enter **now**. If **now** is specified for the update time, the system performs the update immediately. If necessary, the system rebuilds the kernel and reboots the site.

   After the site name and time are entered, the system responds with possible sites.

8. Enter **STOP** in all uppercase letters (as shown) at the list of available sites. You are exited from the **updatep** program and the AIX prompt (#) returns.

Notes:

This update was performed with the **-a -c** options of the **updatep** command. See the *AIX Operating System Commands Reference* for other pertinent options.

For all other sites that were not available for update at this time you must manually run the **qproc** command. When you do this, it is suggested that you have the system nearly idle. If you are in a cluster environment, access to a primary or backbone site to update

¦ Copyright IBM Corp. 1985, 1991

your site's local is required. Should an error message display on the screen, see *IBM AIX Operating System Messages Reference* for details.

If updating local files that require rebuilding the kernel an rebooting the site, update local files on the machine that is running **updatep** by specifying that machine last or specifying a future update time.

Note: If you are updating several diskette programs at once, you will have to load the Update Diskettes more than once.

¦ Copyright IBM Corp. 1985, 1991

4.5 Committing Updates

Once an update has been applied, the update should be committed to make it a permanent part of the system. Committed updates are stored more permanently than applied updates.

When an update is committed, the files existing prior to the update are moved from the file **/usr/lpp/***lpp_name***/inst.updt.save** to another location. This makes the system ready for new updates to be applied.

Note: Log in as **root** or be the superuser.

To commit an update to the system, follow these steps:

1. At the AIX prompt (#), enter:

   ```
   # updatep -c
   ```

   The system displays the "Commit Updates Menu" which is similar to the following:

   ```
   +-------------------------------------------------------------------------------
   ¦
   ¦ 048-082 Commit Updates Menu
   ¦
   ¦ No Special Function Required
   ¦
   ¦     ID UPDATE
   ¦
   ¦ PROGRAM1 Program
   ¦
   ¦    To cancel the updatep command enter "quit".
   ¦
   ¦    You may select one or more numbers from the list.
   ¦
   ¦    Type the ID numbers of the programs you wish to commit separated
   ¦    by spaces (example: 1 3) and press Enter.
   ¦
   ¦    --->
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   +-------------------------------------------------------------------------------
   ```

   Note: The options listed on this menu depend on the programs for which updates are currently applied on the system.

   From this menu, you choose programs for which updates are to be committed.

2. Enter the number corresponding to the updated program to be committed. To commit updates for more than one program, enter the numbers for the programs separated by spaces.

¦ Copyright IBM Corp. 1985, 1991

After you press **Enter**, messages similar to the following example are displayed indicating that the system is saving the files that are being overwritten. These files can be retrieved later if necessary.

```
+-------------------------------------------------------------------------
¦
¦ Saving /usr/lpp.save/2/PROGRAM to /usr/lpp.save/2/PROGRAM
¦ File System Backup Utility
¦ Done     at Fri Jun 17 14:52:08 1988
¦ 40 blocks on 1 volume(s)
¦ Saving status /usr/sys/inst_updt to /usr/lpp.save/2/status
¦ File System Backup Utility
¦ Done     at Fri Jun 17 14:52:23 1988
¦ 40 blocks on 1 volume(s)
¦ 048-089 The system is now starting to commit the updates for program
¦         "PROGRAM1 Program".
¦ 048-088 All requested update processing is completed.
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
+-------------------------------------------------------------------------
```

Note: The exact content of these messages depends on the updates you have chosen to commit.

During the above operation, you will see the message:

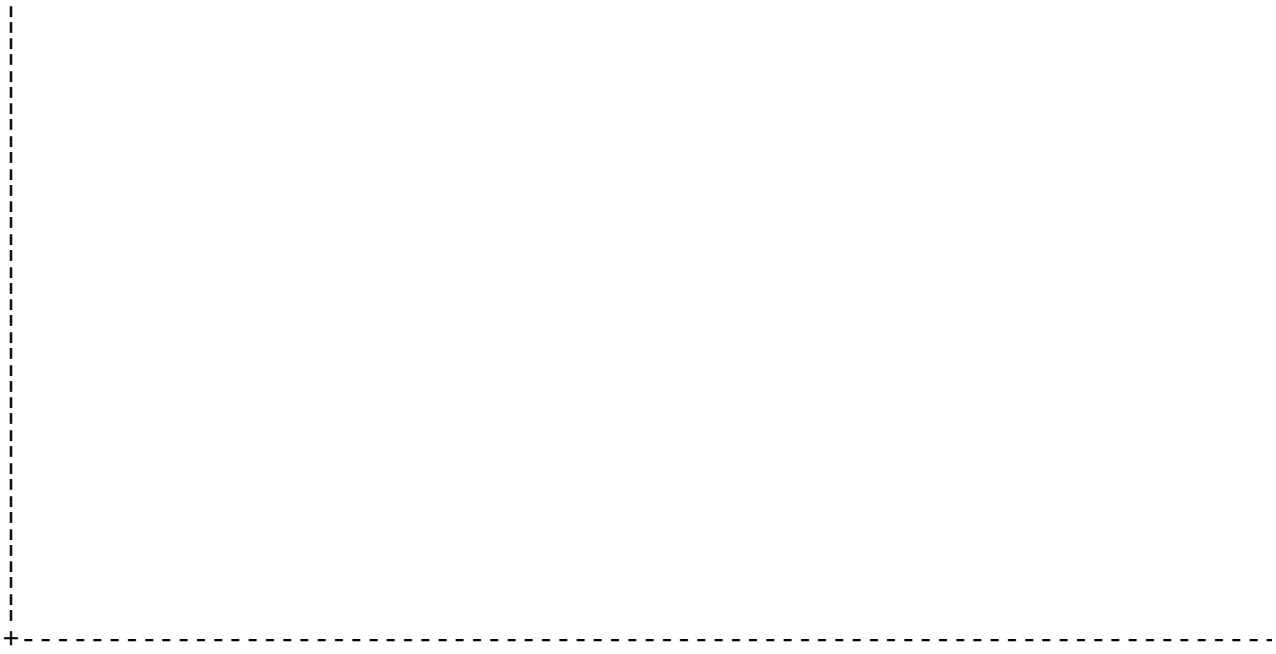
```
  Saving /usr/lpp/... to /usr/lpp.save/NN/...
```

You should record the specific frame number displayed in the message. This frame number is used when using the **cleanup** command described later. This command allows you to manage the amount of disk space used by the service system.

3. In this example, all the necessary files have been committed on the replicated root file system. If there were local updates, the system then displays the following message, which asks you to specify the sites to which you wish to commit updates and the time you wish the execution to begin.

```
+-------------------------------------------------------------------------
¦
¦ The following sites are available:
¦ SITE1 SITE2 SITE3 SITE4 SITE5
¦ Please enter site name followed by the desired execution time
¦ You can enter "all" to choose all sites and "now"
¦ for immediate execution
¦ (STOP for break)
¦
```

¦ Copyright IBM Corp. 1985, 1991



4. Specify the site to which you are committing updates and the time you want the system to commit the updates by entering:

   sitename time

   The **sitename** is the name of the site to which you are committing updates. To commit updates to local files on all the listed sites, enter **all**. The **time** indicates when the update procedure is to begin on the specified site. Enter times using the syntax of the **at** command. To begin the commit immediately, enter **now**.

   After you enter all of the site names and times, the system responds with possible sites.

5. Enter **STOP** in uppercase letters (as shown) at the list of available sites. The **updatep** program exits and the AIX prompt (#) returns.

Subtopics
4.5.1 Cleaning Up Updates

¦ Copyright IBM Corp. 1985, 1991

4.5.1 Cleaning Up Updates

During the apply or commit phases of service, you are using up disk space on your system in order to preserve the older versions of commands and library members. If you wish, you may use the **cleanup** command to manage some of this space.

During the commit phase, you were asked to record certain information in a save-stack frame. The **cleanup** command can be used to archive one or more of these to tape, or to restore them later if you need them in order to restore previous service levels.

If at a later time you wish to uncommit some update on which you have used the **cleanup** command, you will have to restore it to the system. Use **cleanup -r** *frame number* to retrieve this information.

Subtopics
4.5.1.1 Other Notes

¦ Copyright IBM Corp. 1985, 1991

4.5.1.1 Other Notes

If problems are found during the update process, that level of servic will automatically be rejected, and the system returns to its original state.

Service is applied in this system with a Last-In-First-Out strategy That is, the last service you applied is the only one that you can manipulate until that step is rejected. Hence, in order to remove service levels that were committed earlier, you will have to remove any subsequently applied service in the same order that it was applied. The update commands will help you with this, notifying you of previous service that must be removed before some particular level of service is applied.

When service is applied together for multiple Licensed Progra Products, that service is manipulated as a whole, and must then be committed, uncommitted, or rejected together.

See the *AIX Operating System Commands Reference* for more specific information.

¦ Copyright IBM Corp. 1985, 1991

4.6 Uncommitting Updates

Uncommitting an update reverses the process performed when the update was committed. When an update is uncommitted, the system returns to the same state as when the update was applied and the files existing prior to the update are moved back to **/usr/lpp/***lpp_name***/inst.updt.save**.

Note: Log in as **root** or be the superuser.

To uncommit an update on the system, follow these steps:

1. At the AIX prompt (#), enter:

   ```
   updatep -u lpp_name version_number
   ```

   The **lpp_name** is the name of the program to which the update or fix is to be uncommitted. The **version_number** is the number of the update you wish to uncommit.

   The system displays the following message:

   ```
   +-----------------------------------------------------------------------
   ¦
   ¦ 000-123 Before you continue, you must make sure there is no other activi
   ¦         on the system.  You should have just restarted the system, and r
   ¦         other terminals should be enabled.  Refer to your messages refer
   ¦         book for more information.
   ¦
   ¦         Do you want to continue with this command?  (y or n)
   ¦
   ¦         --->
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   +-----------------------------------------------------------------------
   ```

2. Enter **y**.

   Messages similar to the following are displayed as the system begins the uncommit procedure.

   ```
   +-----------------------------------------------------------------------
   ¦
   ¦     The following LPPs need to be uncommitted and rejected for this upda
   ¦     PROGRAM  00.00.0020.0000 use: updatep
   ¦ LPPs committed together, and will be all uncommitted
   ¦ Uncommit proceeds ....
   ¦ File System Restore Utility
   ```

¦ Copyright IBM Corp. 1985, 1991

```
¦ x .
¦ x ./lpp.loc
¦ x ./lpp.loc/info
¦ x ./lpp.loc/lpp.hist
¦ x ./lpp.loc/inst_updt.loc
¦ x ./lpp.loc/files
¦ x ./lpp.loc/files/local
¦ x ./lpp.loc/files/local/PROGRAM1
¦ x ./lpp.loc/files/local/PROGRAM3
¦ x ./lpp.loc/files/local/PROGRAM4
¦ x ./lpp.loc/al.loc
¦ x ./lpp.hist.bak
¦ x ./lpp.hist
¦ x ./sl
¦ x ./appscr.0005
¦ x ./lpp.uninst
¦ x ./rejscr.0005
¦


¦
¦ x ./inst_updt.save
¦ x ./inst_updt.save/update.list
¦ x ./inst_updt.save/update.1
¦ x ./inst_updt.save/update.2
¦ x ./inst_updt.save/update.3
¦ x ./inst_updt.save/update.4
¦ x ./inst_updt.save/update.5
¦ x ./al
¦     files restored: 25
¦ File System Restore Utility
¦ x .
¦ x ./inutemp.ndc
¦ x ./inutemp.dc
¦ x ./lpp_name
¦ x ./inutemp.vrmc
¦ x ./inutemp.mnuf
¦ x ./inutemp.max
¦ x ./updt_cntrl
¦ x ./inutemp.tmp3
¦ x ./inutemp.tmp
¦ x ./lppsize
¦ x ./inuapply.ok
¦ x ./inuname.tmp
¦     files restored: 13
¦
¦
+-----------------------------------------------------------------------
```

Note: The exact content of these messages depends on the updates that you have chosen to uncommit.

Updates to a program must be uncommitted in reverse order from the order in which they were committed. In other words, if updates were committed in the following order:

```
update1
update2
update3
```

Then, to uncommit **update1**, you must first uncommit **update3** followed by

¦ Copyright IBM Corp. 1985, 1991

update2.

If updates are uncommitted in the incorrect order, the following error message is displayed:

```
+----------------------------------------------------------------------
¦
¦     The following LPPs need to be uncommitted and rejected for this upda
¦     PROGRAM  00.00.0020.0000 use: updatep
¦ You must first uncommit/reject the above LPPs
¦
+----------------------------------------------------------------------
```

If you receive this message, uncommit the updates listed in the error message before continuing the procedure.

When the uncommit procedure is completed on the replicated root file system, the following message is displayed:

```
+----------------------------------------------------------------------
¦
¦ The following sites are available:
¦ SITE1 SITE2 SITE3 SITE4 SITE5
¦ Please enter site name followed by the desired execution time
¦ You can enter "all" to choose all sites and "now"
¦ for immediate execution
¦ (STOP for break)
¦
```

3. Specify the sites to which the update should be uncommitted and the times when the uncommit procedure should begin by entering:

   sitename time

   The **sitename** is the name of the sites on which the updates are to be uncommitted on the local files. To uncommit updates to local files on all the listed sites, enter **all**. The **time** is the time at which the uncommit procedure is to begin on the specified site. Enter times using the syntax of the **at** command. To begin the uncommit immediately, enter **now**.

   After entering the sitename and time, the system responds with available sites.

4. When finished specifying sites, enter **STOP** in all uppercase letters (as shown) at the list of available sites. You are exited from the **updatep** program and the AIX prompt returns.

Subtopics
4.6.1 Example

¦ Copyright IBM Corp. 1985, 1991

4.6.1 Example

Uncommitting updates on a single site called SITE1 at 7:00 p.m. may be similar to the following example:

```
The following sites are available:
SITE1 SITE2 SITE3 SITE4 SITE5
Please enter the site name followed by the desired execution time.
You can enter "all" to choose all sites and "now"
for immediate execution.
(STOP for break)
SITE1 7 p.m.
The following sites are available:
SITE1 SITE2 SITE3 SITE4 SITE5
Please enter the site name followed by the desired execution time.
You can enter "all" to choose all sites and "now"
for immediate execution.
(STOP for break)
STOP
```

¦ Copyright IBM Corp. 1985, 1991

4.7 Rejecting Updates

If an error occurs during the kernel rebuild, an error message and the order in which you must reject the licensed programs are displayed. After the system restarts, you will need to reject the programs in the order previously displayed with the error message. Those licensed programs that did not require a kernel rebuild will complete the apply/commit process and will not be displayed.

Rejecting an update reverses the process performed when the update was applied. When an update is rejected, the system returns to the same state as it was before the update was applied.

When an update is uncommitted, the files existing prior to the update are reconstructed using the information stored in the **/usr/lpp/***lpp_name***/inst_updt.save** file. This directory is then deleted.

When you reject an update that has already been committed, you must first uncommit that update.

To reject an update to the system, log in as **root** or be the superuser, and stop all running processes. Then, follow these steps:

1. At the AIX prompt (#), enter:

   ```
   updatep -r
   ```

   The system displays the following message:

   ```
   +-------------------------------------------------------------------------
   ¦
   ¦ 000-123 Before you continue, you must make sure there is no other activi
   ¦         on the system.  You should have just restarted the system, and r
   ¦         other terminals should be enabled.  Refer to your messages refer
   ¦         book for more information.
   ¦
   ¦         Do you want to continue with this command?  (y or n)
   ¦
   ¦         --->
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   ¦
   +-------------------------------------------------------------------------
   ```

2. Enter **y**.

   The system then displays the "Reject Updates Menu", which is similar to the following. The options listed on this example menu depend on

¦ Copyright IBM Corp. 1985, 1991

the updates currently applied on the system.

```
048-082 Reject Updates Menu

        No Special Function Required

         ID UPDATE

          1 PROGRAM1 Program

        To cancel the updatep command enter "quit".

        You may select one or more numbers from the list.

        Type the ID numbers of the programs you wish to reject separated
        by spaces (example: 1 3) and press Enter.

        --->
```

3. Enter the number corresponding to the update you want to reject. If more than one update is to be rejected, enter the numbers for the updates separated by spaces.

   After you press **Enter**, messages similar to the following are displayed:

```
048-089 The system is now starting to reject the updates for program
        "PROGRAM1 Program".
   -    File "/usr/bin/PROGRAM/PROGRAMx" is being recovered.
   -    File "/usr/bin/PROGRAM1" is being recovered.
   -    File "/usr/bin/PROGRAM3" is being recovered.
   -    File "/usr/bin/PROGRAM4" is being recovered.
   -    File "/usr/bin/PROGRAM5" is being recovered.
048-088 All requested update processing is completed.
```

¦ Copyright IBM Corp. 1985, 1991

Note: The exact content of the messages depends on the updates that are to be rejected.

When you see the message indicating that the processing is completed, the indicated updates have been rejected on the replicated root file system.

4. Specify the site on which you want updates rejected and the time you want to reject the updates by entering:

   sitename time

   The **sitename** is the name of the sites on which updates are to be rejected to local files. To reject updates to local files on all the listed sites, type **all**. The **time** is the time at which the reject procedure is to begin on the specified site. Enter time using the syntax of the **at** command. To begin the reject immediately, enter **now**.

5. After the sitename and time are entered, the system responds with available sites. When finished specifying sites, enter STOP in all uppercase letters (as shown) at the list of available sites. The **updatep** program exits and the prompt returns.

Subtopics
4.7.1 Example

¦ Copyright IBM Corp. 1985, 1991

4.7.1 Example

The following example shows how to reject updates on all the sites in your system.

```
+-------------------------------------------------------------------------------
¦
¦ The following sites are available:
¦ SITE1 SITE2 SITE3 SITE4 SITE5
¦ Please enter the site name followed by the desired execution time.
¦ You can enter "all" to choose all sites and "now"
¦ for immediate execution.
¦ (STOP for break)
¦ all now
¦ The following sites are available:
¦ SITE1 SITE2 SITE3 SITE4 SITE5
¦ Please enter the site name followed by the desired execution time.
¦ You can enter "all" to choose all sites and "now"
¦ for immediate execution.
¦ (STOP for break)
¦ STOP
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
+-------------------------------------------------------------------------------
```

Note that if the word, **now** is specified for the reject time, the system performs the reject immediately. If necessary, the system rebuilds the kernel and reboots the site when **ENTER** is pressed.

Note: If you are rejecting updates to local files that require rebuilding the kernel and rebooting the site, first reject updates to local files on the machine that is running the **updatep** command, then specify that machine last or specify a future reject time. If rejecting updates to local files that require rebuilding the kernel and rebooting the site immediately (by typing **now** as the reject time) on the machine on that is running **updatep**, the **updatep** procedure will be stopped when the kernel is rebuilt and the machine rebooted; therefore, STOP does not have to be entered.

¦ Copyright IBM Corp. 1985, 1991

4.8 Updating the System and Installing Local Programs

There are three ways in which you can alter the basic features of the computer system:

Install additional Licensed Product Programs (LPP), or components o licensed programs (using the **installp** command as is explained below, and in the *AIX Operating System PS/2 Installation and Customization Guide*).

Make updates to one or more licensed programs or components o licensed programs that are already installed.

Create your own programs

This section contains guidelines for updating your system and installing your own programs.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.8.1 Installing a Licensed Program Product on an AIX PS/2

This section describes how to use **installp** to install a Licensed Program Product (LPP) on an AIX PS/2 system. An LPP is a software program that remains the property of the manufacturer for which you, as the customer, pay a license fee.

For information on installing LPPs during initial operating system installation and on IBM-supplied LPP sizes, see *Installing and Customizing the AIX PS/2 Operating System*, *Installing and Customizing the AIX/370 Operating System*, and documentation accompanying the actual LPP.

To install an LPP from the distribution diskette on a PS/2 or PS/55, the following procedures are required:

1. Log into the system as root.

2. Insert the distribution diskette into the diskette drive and make sure it locks in place.

3. Type:

   ```
   # installp -d /dev/rfd0h
   ```

   The system responds:

   ```
   000-123  Before you continue, you must make sure there is no other
   activity on the system.  You should have just restarted the system and
   no other terminals should be enabled.  Refer to the AIX Operating Syste
   Messages Reference manual for more information.

   Do you want to continue with this command?  (y or n)
   ```

4. Type:

   ```
   y
   ```

   The system responds:

   ```
   Please mount volume 1 on /dev/rfd0
   ```

5. Confirm that the distribution diskette is in the diskette drive and press **Enter**.

   The system responds:

   ```
    files restored:  1
   _

   The program LPP_NAME will be installed

   Do you want to do this?  (y/n)
   ```

   where *LPP_NAME* is the name of the LPP you are installing.

6. Type:

   ```
   y
   ```

¦ Copyright IBM Corp. 1985, 1991

to install the LPP. The system responds with the number of files restored and a copyright message. The following message is displayed:

```
Please insert diskette
1 in diskette reader.
```

7. Confirm that the distribution diskette is in the diskette drive and press **Enter**.

   The system displays the names of the files as they are copied from the distribution diskette. When all the files are copied, the following messages are displayed:

```
 files restored:  nn

_
 Program LPP_NAME is now installed
```

At this point, the system displays messages specific to the LPP you are installing. The system may also display prompts requesting configuration information for the LPP. These prompts should be self-explanatory, and you should answer them as they appear. If you need more information about these prompts, refer to the documentation for the LPP you are installing.

When you finish entering configuration information, the system may need to rebuild the kernel. If necessary, this kernel rebuild will happen automatically.

Warning: Do not interrupt the rebuild process. When the kernel is rebuilt, the system reboots to enable the new kernel.

¦ Copyright IBM Corp. 1985, 1991

4.8.2 Updating the System

An ***update*** is an improvement for some part of the system. An update may apply to one or more licensed programs or components of licensed programs. If IBM supplies updates for your Operating System, you will receive a tape or diskette containing the update. You will use the **updatep** command to apply updates.

Updates for IBM-provided software products on your AIX Operating System are packaged together on the same tape or diskette. You select the products for which you want to apply updates. After the updates are applied, you can selectively ***commit*** or ***reject*** the complete set of updates for each product based on the results of a test period. You must be a member of the **system** group to run the **updatep** command.

You start an update by running the **updatep** command, which performs four functions:

updatep -a Applies selected product updates.

updatep -c Commits selected product updates that have been applied.

updatep -r Rejects selected product updates that have been applied.

updatep -s Presents status about all currently applied product updates.

Note: You can use the **-a** and **-c** flags together. This provides the fastest update of multiple licensed program products because it can minimize the number of kernel builds and reboots.

You must enter the appropriate action when you enter the **updatep** command.

A menu is presented if you choose one of the first three actions. If you choose **apply**, the menu contains every product on the diskette for which there is an earlier version on the system (that is, a version with a lower update level). If you choose either **commit** or **reject**, the menu contains all of the products on the system that have applied updates.

Select the appropriate products from the **apply**, **commit**, or **reject** menu. The action (**reject**, for example) is performed for all selected products.

For additional information on using **updatep**, see **updatep** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

4.8.3 Installing Applications and Installation-Specific Commands

If you create your own commands or programs (shell programs, for example), the following guidelines will make them convenient to use:

Create a directory for installation-specific commands. If you create commands that are to be available to all system users, place them in a single directory (for example, **/lbin**). If you create commands that are to be available only to members of certain groups, create a different command directory (for example, **/lbin/system**) and set its permissions appropriately.

By keeping installation-specific commands together in one (or two) directories, it becomes simple to:

- Remember the path name of the command.
- Determine what installation-specific commands are on the system (by listing the contents of the local command directory).

Create a directory for private commands. If you (or other system users) create commands for private use, keep them in a single subdirectory of the **$HOME** directory (for example, **/u/tom/bin**). This type of organization has the same advantages for an individual user that it does, on the larger scale, for all users of the system--it is easy to remember where the commands are and, thus, to distinguish the standard system from the parts of the system that you have added or created.

Note: Use ASCII characters for the path name to ensure that users of different locales can communicate successfully.

¦ Copyright IBM Corp. 1985, 1991

4.9 Setting the System Date and Time

This section covers setting time and date on your system with the **date** command, or the alternative **timed** command.

The computer system has an internal, battery-powered clock. You set the time and date when you install the system, and the clock maintains the time and date whether power to the system is on or off. To reset the time or date (for example, to synchronize system time with standard time or because of a battery failure) you can use the **date** command. You also can use the **date** command to find out what the current system date and time are.

To set the date and time, enter a command of the form:

```
date MMddhhmm.ssyy
```

where:

MM designates the month.
dd designates the day.
hh designates the hour, using a 24-hour clock (for example, 7 p.m. is represented as 19).
mm designates the minutes.
.ss designates the seconds.
yy designates the year.

Each time or date designation must be two digits long (for example, the date designation for July is 07).

Note: Do not set the date or time while other users are logged in to the system. You must have superuser authority to set the time and date.

In the following example, the **date** command sets the date and time to April 18, 3:15.32 (32 seconds after 3:15) p.m., 1985:

```
date 04181515.3285
```

To display the system's current date and time values, simply enter: **date**.

If you are part of a LAN, the system administrator may choose to use the **timed** command to keep your local systems synchronized with other systems on the LAN. See *AIX TCP/IP User's Guide* for more information regarding the **timed** command.

For more information about the **date** command, see **date** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

4.10 Running Commands at Pre-set Times

The **cron** command resembles a clock that can run shell commands at pre-set dates and times. You do not enter the **cron** command; rather, **cron** is included in the system startup scripts and starts during the system initialization process. Specify what commands are to run and when they are to run, with either the **at** command or the **crontab** command:

at Use **at** when a command is to be run only once.

crontab Use **crontab** when a command is to be run regularly.

Note: There is a difference between the command, **crontab** and the filename **/crontabs/**.

Note: To ensure that frequently used files and directories are open when needed by other commands, list them in the **/etc/openfiles** special file. This also ensures that other commands run faster. **cron** reads **openfiles** at periodic intervals to open (and keep open) directories and files listed in **/etc/openfiles**.

A default **openfiles** is provided with your system. When you edit this file you may include comments preceded by a pound sign (#) in the first column position; any blank lines in the file are ignored by **cron**. For more information on **cron** and **/etc/openfiles**, see the *AIX Commands Reference*, and the *AIX Technical Reference*.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.10.1 Using the at Command

You can use the **at** command if your name appears in the **/usr/adm/cron/at.allow** file. If this file exists, the superuser's login name must be included in it for the superuser to be able to use the **at** command. If this file does not exist, the **at** command checks the **/usr/adm/cron/at.deny** file to determine if you should be denied access to using the **at** command. If neither file exists, only the superuser can use the **at** command.

The **at** command takes its list of commands from standard input. Standard output and standard error from the commands are mailed to the user who ran the **at** command unless they are redirected.

To Use the at Command

1. Enter:

   at time date +increment

2. Enter the commands to be run (either one command per line or separated by semicolons [;]).

3. Press **END OF FILE** (**Ctrl-D**).

Following is an explanation of each **at** command parameter and how to set it:

time Required parameter. Specify **time** in one, two, or four digits. **at** interprets one- and two-digit numbers as hours, four-digit numbers as an hour and minutes (for example, **10** means 10 o'clock, **1043** means 10:43). Though not required, you can separate the hours and minutes with a colon (**10:43**).

To indicate whether the time is a.m. or p.m., do one of the following:

- Append an **am** or **pm** suffix to **time** (for example, **10:43pm**).
- Use a 24-hour clock (for example, 10:43 p.m. is **22:43**). If you do not supply an **am** or **pm** suffix, **at** interprets the time based on a 24-hour clock.

You also can use the following special names in the **at time** specification:

noon
midnight
now
next

Note: The special name **next** can be combined with the following terms to form compounds expressing times to run commands: **minute**, **hour**, **day**, **week**, **month**, **year**.

date Optional parameter. Specify **date** in one of the following ways:

¦ Copyright IBM Corp. 1985, 1991

monthname daynumber,yearnumber. For **monthname**, use a three-letter abbreviation (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, or Dec). For **daynumber**, use a one- or two-digit number. The **yearnumber** is an optional two-digit number preceded by a comma.

dayname. Specify the name of a day of the week, either fully spelled out or abbreviated to three letters.

The **at** command also recognizes the following special days:

today
tomorrow

If you do not specify **date**, **at** interprets the time to be:

today if **time** is later than the current time
tomorrow if **time** is earlier than the current time.

+increment Optional parameter. The **+increment** parameter, which is sometimes a more convenient way to specify time or date, is a number followed by one of these special words:

minute or **minutes**
hour or **hours**
day or **days**
week or **weeks**
month or **months**
year or **years**.

For example, to run a command at this same time tomorrow (rather than specifying the exact time) you could set up an **at** job like the one in the following example:

```
at now + 1 day
skulker
END OF FILE
```

Enter **at -l** to list the **at** jobs you have scheduled. Note that **at -l** lists **at** jobs by the numbers **at** assigns them. To see what commands a job contains, look at the file **/usr/spool/cron/atjobs/jobnumber**.

To cancel a scheduled command, use the **-r** flag:

```
at -r jobnumber
```

¦ Copyright IBM Corp. 1985, 1991

4.10.2 Using the crontab Command

The **crontab** command copies a specified file into the directory that contains all **crontab** files. The **cron** command (started by system initialization scripts) runs commands according to instructions in **crontab** files.

The **crontab** command has the following format

crontab **filename**

where **filename** is the name (or path name) of a file that contains information **cron** uses to schedule commands.

A **crontab** file can contain more than one entry (one entry corresponds to one command or command group). Each entry is on a separate line and consists of the following six fields, separated by spaces:

minute hour day/month month/year day/week command

Each field except **command** can contain:

A number in the specified rang

Minute (0-59)
Hour (0-23)
Day of the month (1-31)
Month of the year (1-12)
Day of the week (0-6 for Sunday-Saturday)

Two numbers separated by a **-** (hyphen) to indicate a range of values (for example, **4-15**)

A list of numbers separated by commas, which specifies each number i the list (for example, **4, 8, 17**)

An ***** (asterisk), which specifies all possible values. To specify days in only one field, the other field should contain an ***** (asterisk). See example below.

cron runs the command at the specified time. If you include a percent sign (**%**) in the sixth field, **cron** takes everything before the **%** as the command and everything after it as input for the command.

Following is a sample **crontab** file named **happy**, which you can create with an "vi" editor:

```
0 16 * 12 5 wall%HAPPY HOLIDAYS!
```

This **crontab** file writes a message to all logged-in users (**wall**) at 4:00 p.m. (**0** minutes, **16** hours), every day (*****), in December (**12)**, on Fridays (**5**). The **wall** command takes all the text following the **%** (percent sign) as its standard input.

Once you create **happy**, enter it into the **crontab** directory with the command: **crontab happy**.

You can modify the action of **crontab** with the following flags:

-l Lists the contents of your **crontab** file.

¦ Copyright IBM Corp. 1985, 1991

-r Removes your **crontab** file from the **crontab** directory.

Note: If you are logged in as the user **root**, you cannot run **crontab** on a file named **/usr/spool/cron/crontabs/su** unless **su** is the first entry in **/etc/passwd**. Similarly, if you are logged in as the user **su**, you cannot run **crontab** on a file named **/usr/spool/cron/crontabs/root** unless **root** is the first entry in **/etc/passwd.**

¦ Copyright IBM Corp. 1985, 1991

4.11 Monitoring Files and Directories That Get Larger Automatically

Even though the fixed disk on the computer system holds a large amount of data, it can become crowded, even full. Certain files on the system can grow automatically, using disk space to store data that are no longer useful. Following is a list of files that you may or may not have on your system. If you do have any of these files, you should check them periodically and remove all useless data.

Accounting files

/usr/adm/wtmp

/usr/adm/pacct

/usr/adm/acct/nite/*

Note: **/usr/adm/wtmp** is a symbolic link to **<LOCAL>/adm/wtmp**. The **wtmp** file actually resides in **<LOCAL>/adm/wtmp**, not **/usr/adm/wtmp**.

Other files

/usr/adm/cron/log

/usr/spool

/usr/adm/messages

$HOME/mbox

Also check the **/tmp** directory for files that you can delete. Ordinarily, processes should remove their temporary files from **/tmp**, but they may not always do so.

You can use the following shell procedure to locate files in specified directories (in this case, **/usr/adm**, **/usr/lib**, and **/usr/spool**) that are larger than a certain number of blocks (in this case, **50**):

```
for j in `ptn`
do
    on $j for i in /usr/adm /usr/lib /usr/spool
    do
       find $i -size +50 -exec ls -l {} \;
    done
done
```

¦ Copyright IBM Corp. 1985, 1991

4.12 Finding Files and Directories

The **find** command searches the file system for all file names that match a specified ***expression*** (characteristic or group of characteristics). For example, the following **find** command lists the complete path name of all files in the file system with the file name **.profile**:

```
find / -name .profile -print
```

This command may take a few minutes to complete depending upon the size of your directory structure. The **/** starts the **find** command at the root directory; the search continues through all subdirectories of **/**. The **-name** parameter specifies the name that **find** is to match. The **-print** parameter causes **find** to display the path name of each file that matches the **-name** expression.

The **-name** expression is just one of several expressions that you can use with the **find** command. For more information about **find** and its expressions, see **find** in *AIX Operating System Commands Reference*.

You can use the **find** command together with the **skulker** and **xargs** commands to periodically remove unwanted files from the file system. For more information, see **skulker** and **xargs** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

4.13 Managing Display Station Features

This section covers two topics:

- How to automatically set the characteristics of any display station o your system
- How to use the special features of the main display station

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.13.1 Setting Display Station Characteristics Automatically

A display station has standard characteristics, such as the length of the lines displayed on the screen, the number of lines per screen, and whether the special command line editing functions are available. However, you can use the **stty** command to change certain display station characteristics. (For more information about **stty**, see *Using the AIX Operating System* and **stty** in *AIX Operating System Commands Reference*.)

If you want the display station characteristics set in a particular (nonstandard) way every time you log in, you may find it convenient to have the system run **stty** automatically. Depending upon your requirements, you can do this in one of two ways:

Place the **stty** command in the **/etc/profile** file.

Each time you start the system, the system automatically runs the commands in **/etc/profile**. If you include **stty** in **/etc/profile**, every display station on the system starts with the same characteristics.

Place the **stty** command in your user profile file, **$HOME/.profile**.

Each time you log in, the system automatically runs the commands in **$HOME/.profile** (the individual user's profile). Commands in **$HOME/.profile** supersede those in **/etc/profile**. Therefore, if you include the **stty** command in your user profile, the display station always has the same characteristics when you first log in.

For more information about these two profiles, see "Setting Environment Variables in the Shell" in topic 1.2.6.3.

¦ Copyright IBM Corp. 1985, 1991

4.13.2 Managing Special Features of the Main Display Station

The **/dev/hft** device driver provides the special features of the PS/2 console. (**hft** stands for high function terminal.) Another device driver, **/dev/console**, also can run the main display station (or ***console***), but it does not provide the special features).

Note: You should open another virtual terminal with **open sh** to run commands and programs; thus, leaving **/dev/console** free to receive error messages. To ensure that error messages do not scroll off the screen, enter **stty page length 22**.

"Managing the PS/2 Virtual Terminal Feature" in topic 4.13.2.1 explains how to manage virtual terminals, a special feature you can use directly. "Additional PS/2 Main Display Station Features" in topic 4.13.2.2 lists some other special features provided by the **/dev/hft** device driver.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.13.2.1 Managing the PS/2 Virtual Terminal Feature

As explained in *Using the AIX Operating System*, a PS/2 virtual terminal is one of several equivalents of a display station, one or more of which can be available at the main display station concurrently. The virtual terminal feature is available on PS/2 systems only. The **actmngr** (activity manager) command monitors virtual terminal activity.

To Enable actmngr

1. Enter:

   adduser

2. Use the **c** (change) subcommand to place **/bin/actmngr** in the **program** field for the appropriate user.

3. End **adduser**.

The **actmngr** command creates the initial shell and then monitors the number of open virtual terminals until all are closed. If you try to end the initial shell before all virtual terminals are closed, **actmngr** restarts the initial shell.

If you do not have virtual terminal capabilities (that is, you are not logged in at the main display station), the system replaces **actmngr** with the initial shell program.

¦ Copyright IBM Corp. 1985, 1991

4.13.2.2 Additional PS/2 Main Display Station Features

In addition to virtual terminal capability, the device driver for the PS/2 main computer display station provides several other special features:

Sound contro

Keyboard contro

Locator (mouse) contro

Font choice

Note: The two fonts available on the PS/2 do not offer a style choice. Instead, they offer a choice between a fast hardware font which can display only the first 256 symbols out of the 616 available or a slower software font which can display all of the 616 symbols. In most cases, you do not need all the symbols, so you should consider using the **display** command to select the fast hardware font. For more information on the symbols available, see **display symbols** in *AIX Operating System Technical Reference*.

Controlling Sound: The **sound** command controls the volume of the sound output. The following flags control the volume of all sound output:

-h, -l, -m Turns the volume on.

-o Turns the volume off.

Note: The **h** (high), **m** (medium), and **l** (low) settings are for RT compatibility only.

The **sound** command in the following example sets the sound volume on:

```
sound -l
```

Controlling the Keyboard: The **keyboard** command controls the ***delay rate*** and ***repetition rate*** of the keyboard. The delay rate is the interval from when you press a key to when it begins to repeat. The repetition rate is the number of times the key repeats per second. These rates are set at system startup to 500 milliseconds and 14 characters per second, respectively.

The **keyboard** has two flags:

-drate Sets the delay rate to the specified value; **rate** can be **250**, **500**, **750**, or **1000**.

-rrate Sets the rate of repetition to the specified value. This **rate** can be an integer from **2** to **30**.

The following **keyboard** command sets the delay to 250 milliseconds and the repetition rate to 30 characters per second:

```
keyboard -d250 -r30
```

Controlling the Locator (Mouse): The **locator** command controls the rate at which the system checks the position of the locator (mouse). You can specify any of the following rates: **10**, **20**, **40**, **60**, **80**, or **100** times per

¦ Copyright IBM Corp. 1985, 1991

second. At system startup, the locator rate is set at **60**.

The following **locator** command sets the rate at **40** times per second:

```
locator -r40
```

¦ Copyright IBM Corp. 1985, 1991

4.13.2.3 Changing the PS/2 Physical Display

With the **display** command, you can change the PS/2 virtual terminal characteristics to be used when you open a PS/2 virtual terminal.

The flags to the **display** command are:

-b Changes the background color on the display

-f Changes the foreground color on the display

-p Changes the color palette to be used

-t Changes the font.

To see a menu of available options, enter **display** without flags or parameters. For example, when you enter **display -t**, you see a menu that contains a list of the available fonts. Choose one that is suitable for the work you will be doing. Not all fonts are available on all displays, nor do they all work with all programs.

You can use the **display** command in your **.profile** or **.cshrc** file to automatically set the desired colors and font when you open a new shell.

For more information, see **display** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

4.13.3 TERM Values for Different Displays, Adapters, and Terminals

Change the value of **TERM** in the **.profile** or **.cshrc** file and add **export TERM** to the **.profile** or **.cshrc** file. You can also use the **termdef** command, which returns the terminal name for your current logged-in terminal.

Use the following values for **TERM**:

Table 4-3. TERM Values for Different Displays, Adapters, and Terminals

Display/Terminal	**Adapter**	**Value**
IBM PS/2 8503, 8507, 8512, 8513, 8514, or 8604	IBM PS/2 VGA IBM PS/2 8514/A	ibm8503 ibm8507 ibm8512 ibm8513 ibm8514 ibm8604
IBM PS/55 5574-M06, 5574-W06, 5574-C06, 5574-C07, 5574-C09	Display Adapter II	ibm5574-M06 ibm5574-W06 ibm5574-C06 ibm5574-C07 ibm5574-C09
IBM 3161 ASCII Display Station	n/a	ibm3161
IBM 3163 ASCII Display Station	n/a	ibm3161
DEC VT100 (terminal)	n/a	vt100
DEC VT220 (terminal)	n/a	vt220
IBM 3161 ASCII Display Station with cartridge (1)	n/a	ibm3161-C
IBM 3162 ASCII Display Station	n/a	ibm3161
IBM 3162 ASCII Display Station with cartridge (1)	n/a	ibm3162
IBM 3278 et al.	n/a	3270 3270-24 3270-32 etc.

(1) International character support.

¦ Copyright IBM Corp. 1985, 1991

4.14 Managing Printers

There are two main management tasks associated with printers on the computer system:

Adding new printers to the system

For information about adding a printer that is supported by the computer system, consult the documentation that comes with the printer and the section of the AIX Operating System installation manual that discusses the **devices** command.

For information about adding a printer that is not supported, consult the documentation that comes with the printer.

Altering the way printers work (the subject of this section)

When you print a file, the system sends a byte-stream of characters to the printer. Some of the characters are codes that cause the printer to print particular characters (for example, a-z, A-Z, and 0-9). Other characters are codes that cause the printer to print characters or files in a particular way (for example, underscoring certain characters or making every page 60 lines long).

If you want to send different character codes to the printer (for example, to change the word **that** to the word **this**), you simply edit your file--you do not have to understand that codes are involved. However, altering the way a printer works is somewhat more complicated--you must understand what happens when you print a file, what options you have for sending control information to the printer, and what printer characteristics you can control.

For printing under VM, see your VM administrator, or check the VM document set.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.14.1 The Printing Process

You use the **print** command to send a file to a printer. Often you may do nothing more than enter a command of the form **print filename** and wait for the printer to complete the job.

A file does not go directly to the printer, however. First, the **qdaemon** command places the print request in a queue. The print request stays in the queue until the requested printer becomes available, at which point **qdaemon** runs the appropriate **backend** command. The **backend** command required for each printer is defined in the **/etc/qconfig** file. For more information on queues and the **qdaemon** command, see "Parts of the Queueing System" in topic 1.3.5.1.

For example, if the default printer's backend is **piobe** (printer input/output backend), the **piobe** command processes the file and sends it, along with control information, to the printer. Thus, the printer receives a data stream composed of the contents of the file and the control information supplied by the **piobe** command.

If your printer expects to receive data in a specific format (for example, postscript), you must provide a printing backend to process files and send them to the printer.

¦ Copyright IBM Corp. 1985, 1991

4.14.2 Controlling the Printing Process

You can add printer control information to the printer data stream in the following ways:

Include printer control codes in the file

All printer control information that is unique to a file should be included in that file. For example, if you want to underscore the title of a book or print a paragraph in bold type, you must use an editor to insert codes into your file that cause the printer to start and stop the special treatment at the correct places. Appendix A, "Printer Control Codes" in topic A.0 includes the printer control codes. Consult the documentation for your editor to learn how to use it to enter control codes into a file.

Supply **piobe** flags with the print command.

The **piobe** command recognizes a number of flags that control printer operations such as:

- Starting and stopping condensed, emphasized, double-wide, and double-strike printing
- Printing in specified colors
- Setting the left, right, top, and bottom margins
- Setting the number of lines per vertical inch
- Maintaining the horizontal position on the print line for a line feed or vertical tab control.

All of the **piobe** flags are listed and explained under **piobe** in *AIX Operating System Commands Reference*.

If you want to use particular **piobe** flags for a single print job, specify them along with the **print** command. For example, the **piobe** flag for setting form (page) length is **-fl=num**, where **num** is a number of lines. If the standard **piobe** setting is **-fl=66** and you want the file **printtest** to be printed with 45 lines per page, you can enter the command:

```
print -fl=45 printtest
```

The flag on the command line overrides the standard **piobe** setting for this one job. However, the standard **piobe** form length setting remains 66.

Change the standard **piobe** settings.

To modify some printer characteristics for all print jobs, modify the standard **piobe** flags; **piobe** flags are contained in the **/etc/qconfig** file. (The **piobe** flags are listed under **piobe** in *AIX Operating System Commands Reference*.)

For example, if you want the standard page length for all print jobs to be 45 lines, you can use an editor to modify the **/etc/qconfig** file, setting the value of **-fl** to **45**. Regardless of the page length set in **/etc/qconfig**, you always can specify a different page length for a particular print job by supplying a different value for **-fl** when you enter the **print** command.

¦ Copyright IBM Corp. 1985, 1991

4.15 Maintaining System Performance

System performance can be a highly specialized and complex subject, further complicated by the fact that a tactic that improves the performance of one system might actually impair the performance of a similar system used for different work. A discussion of improving system performance is beyond the scope of this book. However, there are several things you can do to maintain the performance level of your system:

Keep directory files small

Reorganize file systems

Reconstruct minidisks

This section covers each of these system maintenance tasks. (For a description of system configuration parameters that affect performance, see "Generating a New Kernel" in topic 1.3.7.)

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.15.1 Keeping Directory Files Small

Directories that are larger than 10 ***logical*** blocks are very inefficient. A 10-block directory can contain over 1200 entries. (Each logical block can contain 4096 bytes of data.) Thus, there usually is little reason to have larger directories. However, large directories (directories that occupy more than 10 blocks) can occur when, for example:

A user has a large number of closely-related small files

A user does not understand how to use a directory structure t organize files.

A directory (such as **/usr/news**) is shared by several users.

A command or program writes its output to a new file (or files) eac time it runs.

Your first tactic for keeping directory files small should be to inform all users that their directories should contain no more than a certain number of files (a number well below the 1200 that could be possible). At the same time, you should make certain that all users understand how to use directory structures to organize files. (For information on how to use directory structures to organize files, see *Using the AIX Operating System*.)

Your second tactic should be to periodically search all file systems for directories larger than 10 blocks. If any large directories exist, the following **find** command displays a list of them:

```
find / -type d -size +10 -print
```

Simply removing files from a directory does not make that directory smaller; rather, it leaves the directory entries for the removed files vacant and ready to be assigned to other files.

To make a directory smaller, you must:

1. Make sure the directory has an acceptable number of entries. (For convenience, you might establish a 100-file directory size guideline.)

   If you need to remove files from the directory, you can use either the **rm** command (to remove files from the system) or the **mv** command (to move files to another directory).

2. Create a compacted directory or directory structure (one that does not contain the vacant entries).

   To compact a single directory:

   a. Create a new directory with the **mkdir** command.

   b. Move (**mv**) or copy (**cp**) the files from the old directory into the new directory.

   c. Remove all files from the old directory (**rm**) and then remove the old directory (**rmdir**).

   To compact a complete file system, use the **dcopy** command to copy the entire old file system to a new file system. For information

¦ Copyright IBM Corp. 1985, 1991

about **dcopy**, see **dcopy** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

4.15.2 Reorganizing File Systems

When you create a file system, its free list is organized for maximum efficiency. However, as you create, delete, link, unlink, copy, and move files on the file system, the free list becomes less and less organized. At the same time, the physical location of data on the disk becomes less organized. The less organized the free list and data are, the longer average time it takes for the system to access files and directories in the file system. Reorganizing the free list and data of a file system often can improve system response time.

Subtopics
4.15.2.1 Reorganizing Only the Free List
4.15.2.2 Reorganizing Data and the Free List

¦ Copyright IBM Corp. 1985, 1991

4.15.2.1 Reorganizing Only the Free List

Two flags cause the **fsck** command to reorganize free lists: **-s** and **-S**.

The **-s** flag causes **fsck** to reorganize the free list regardless of the condition of the file system. The **-S** flag causes **fsck** to reorganize the free list only if **fsck** finds no inconsistencies in the file system.

If you only want to reorganize the free list, **fsck -S** is the quickest way to do so. You should run **fsck -S** regularly, perhaps once per week or twice per month, depending upon how heavily your system is used.

Note: Always run **fsck** on unmounted file systems.

For more information about the **fsck** command, see "Checking and Repairing File Systems - The fsck Command" in topic 1.3.3.3 and **fsck** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

4.15.2.2 Reorganizing Data and the Free List

There are two ways to reorganize both the data and the free list of a file system:

backup, **mkfs**, and **restore**

1. Back up the complete file system (with the **backup** command).

2. Make a new file system on the minidisk that held the file system (with the **mkfs** command).

3. Restore the backed-up file system to the new file system (with the **restore** command).

This procedure for reorganizing free lists is most appropriate when done in conjunction with routine level 0 system backups. For more information on **backup**, backup levels, and **restore**, see "Backing up Files and File Systems on AIX PS/2" in topic 1.2.9. For more information on **mkfs**, see "Creating and Mounting File Systems" in topic 1.2.8. Also see **backup**, **restore**, and **mkfs** in *AIX Operating System Commands Reference*.

dcopy

Besides compacting directories (as is discussed under "Keeping Directory Files Small" in topic 4.15.1), the **dcopy** command also reorganizes a file system's free list. The main purpose of **dcopy** is to reorganize the file system for faster access times. It is also useful for copying an existing file system into a new file system (for example, when you move a file system from one minidisk to another, perhaps to balance the load among several fixed disks). In such cases, the effect that **dcopy** has on directories and free lists is of secondary importance.

However, if you want to use **dcopy** to reorganize the free list and compact directories, but do not want to move the file system, first use **dcopy** to copy the file system to an available minidisk, and then use either **dcopy** or **dd** to copy the file system back to its original minidisk. (For more information on the **dd** command, see **dd** in *AIX Operating System Commands Reference*.)

¦ Copyright IBM Corp. 1985, 1991

4.15.3 Handling the Minidisk

Your system consists of a certain number of separate file systems, each of which resides on its own minidisk. At the very least, your AIX/370 system has three minidisks for the operating system and one for page space. The AIX PS/2 system may have as few as two minidisks. The exact number of minidisks on your system depends upon the options you have installed and the number of additional minidisks you have created.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.15.3.1 Minidisk Full Condition

The size of each minidisk is set when the minidisk is created. A minidisk that is too small for its purpose can significantly reduce your system's performance. A minidisk can be too small either because it was not made large enough initially (the problem shows up immediately) or because it has become full with use (the problem may appear gradually). If you suspect that one or more minidisks are becoming too full, enter the **df** command to determine, for each mounted file system:

Note: Minidisks do not necessarily need to be 100% full to cause problems. A safe maximum to use in your calculations is 90% full.

Total number of disk block

Number of disk blocks fre

Number of disk blocks used

Note: The **df** command (without arguments) only reports information about file systems with stanzas in **/etc/filesystems** that contain the line **df=true**.

In some cases, you may be able to correct a minidisk size problem by removing files from the file system. Often, however, you will need to reconstruct one or more minidisks, increasing their size as necessary. The remainder of this section describes how to recover from a **minidisk full** condition, whether it results from the installation of a program or from free space being used up over time.

Note: If you are not familiar with the **minidisks** command, please see the information on creating minidisks in the appropriate AIX installation manual before you continue with this section.

To Recover from a minidisk full Condition

1. Use the **minidisks** command to determine the amount and location of existing free space.
2. Determine the amount of free space required.
3. Determine whether sufficient free space exists to meet your requirements.
4. Follow an appropriate recovery procedure.

If you are installing a licensed program, the documentation supplied with the program should state how many blocks the program requires and which file systems (minidisks) must contain those blocks. Thus, you know how much free space such a program requires. However, if a minidisk has simply filled up over time, it is not so easy to determine how much of the available free space you will allocate to it; you will have to make this decision based on your overall understanding of how your system is used.

If there is not sufficient free space to meet your requirements, you have two alternatives when faced with the minidisk full condition:

¦ Copyright IBM Corp. 1985, 1991

Remove a sufficient number of existing files to create free space

Add additional storage to your system (that is, install another fixe disk).

Replace an existing fixed disk with a larger one

If there is sufficient free space available to meet your requirements, you can use one of the following four procedures to recover from a minidisk full condition:

Expand a subdirectory into a file system

Create and mount a new licensed program file system

Expand a file system

Reconstruct existing minidisks

¦ Copyright IBM Corp. 1985, 1991

4.15.3.2 Expanding a Subdirectory into a File System on the Minidisk

This procedure involves creating a new minidisk, moving all data from an existing subdirectory into its own file system on the new minidisk, and then mounting the new minidisk onto the original file system subdirectory.

Note: For this procedure to be successful, the subdirectory to be expanded must have no mounted subdirectories; and there must be sufficient contiguous free space for a copy of the complete subdirectory plus the new data.

Following are the steps for expanding a subdirectory into a file system:

1. Use the **add** subcommand of **minidisks** to create a new minidisk large enough to hold the old subdirectory plus the new requirements. For the mount directory, specify the original subdirectory name.

2. Mount the new minidisk on a temporary directory, for example:

   ```
   mount /dev/hdn /tmp/lpp
   ```

3. Copy the original subdirectory to the new file system. For example, to copy everything under the current subdirectory to the new file system, enter:

   ```
   tar crf | (cd dirname; tar xfv -)
   ```

4. Compare the original subdirectory to the new one (use the **dircmp** command).

5. Unmount the new file system from its temporary directory, for example:

   ```
   umount /dev/hdn
   ```

6. Use **fsck** to check the consistency of the new file system.

7. Delete files from the original subdirectory. (Enter **pwd** to verify that the original subdirectory is your current directory, and then enter **rm -r *** to delete all files under that subdirectory.)

8. Use the **cd** command to move to the parent directory.

9. Mount the new file system onto the original subdirectory.

¦ Copyright IBM Corp. 1985, 1991

4.15.3.3 Creating and Mounting a New Licensed Program File System on a Minidis

This procedure involves creating a minidisk large enough for the new requirements, mounting that minidisk onto a new subdirectory of an existing file system, and installing the new files onto the new file system.

Note: For this procedure to be successful, the disk on which the minidisk is to be created must contain enough contiguous free space to meet the new requirements.

Following are the steps for creating and mounting a new licensed program file system:

1. Use the **add** subcommand of **minidisks** to create a new minidisk large enough to meet the requirements of the new licensed program and specify the minidisk mount directory (for example, **/usr/lpp/lppname**).

2. Create the minidisk mount directory (for example, **mkdir /usr/lpp/lppname**).

3. Mount the new file system onto the minidisk mount directory (for example: **mount /usr/lpp/lppname)**

Note: If you create a separate minidisk for each licensed program, you could eventually reach the maximum number of minidisks (28) and perhaps experience problems due to fragmentation of disk space. Therefore, it is good practice to merge minidisks periodically, if possible.

¦ Copyright IBM Corp. 1985, 1991

4.15.3.4 Expanding a File System on a Minidisk

This procedure involves transferring all data from a mounted file system to a file system on a new, larger minidisk. There must be enough contiguous free space to hold the old file system and the new data. Following are the steps for expanding a file system:

1. Start the maintenance system. (For information on starting and using the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.)

2. Select **Use Maintenance Commands**.

3. Select **Backup commands**.

4. Select **Backup a file system** and back up the file system that you are expanding.

   Note: Do not use the **Backup a minidisk image** option.

5. Select **Delete a fixed disk minidisk**. Delete the minidisk.

6. Select **Create a fixed disk minidisk**:

   a. Select **No preference** to cause the minidisk to be created in the first disk area large enough to accommodate it. (Do not use the **beginning, middle, end** allocation suboptions.)

   b. Specify the number of blocks necessary to meet your new requirements.

7. Select **Make a file system** to create a file system on the new minidisk.

8. Select **Restore commands**.

   a. Select **Restore a file system**.

   b. Restore the old file system to the new minidisk.

9. Select **Check a file system** and check the structure of the new file system.

10. Exit System Maintenance and boot normally.

¦ Copyright IBM Corp. 1985, 1991

4.15.3.5 Rearranging Existing Minidisks

With this procedure, you can physically rearrange the minidisks on the fixed disk(s). Use this procedure if:

There is enough total free space to meet your requirements

and

You cannot use any of the previous procedures because the free spac is not contiguous.

Following are the steps for rearranging existing minidisks:

1. Start the maintenance system from the AIX Operating System Maintenance disk. (For information on starting and using the maintenance system, see "Running the PS/2 and System/370 Maintenance System" in topic 1.2.3.2.)

2. Select **Use Maintenance Commands**.

3. Select **Backup commands**.

4. Select **Backup a file system** and back up the file systems that you are expanding. (If you want to arrange minidisks so that all of the free space is in one area, back up all minidisks between the first and last areas of free space on a particular disk.)

   Note: Use the **Backup a minidisk image** option only for minidisks that are not AIX file systems.

5. Select **Delete a fixed disk minidisk**. Delete the minidisk.

6. Select **Create a fixed disk minidisk** to create a new minidisk to replace each of the minidisks that you deleted:

   a. Select **No preference** to cause the minidisk to be created in the first disk area large enough to accommodate it. (Do not use the **beginning, middle, end** allocation suboptions.)

   b. Specify the number of blocks that you noted in the previous step.

   c. Repeat these steps for each of the new minidisks.

7. Select **Make a file system** and create a file system on each new minidisk that requires one (that is, the ones that originally had file systems).

8. Select **Restore commands**.

   a. Select **Restore a file system**.

   b. Restore the old file system to the new minidisk.

   c. Repeat these steps for each of the new file systems.

   Note: To restore minidisks that do not have AIX file systems (the ones that were backed up with the **Backup a minidisk image** option), select the **Restore a minidisk image** option.

¦ Copyright IBM Corp. 1985, 1991

9. For all minidisks that contain AIX file systems, select **Check a file system** and check the structure of the new file system.

10. Exit System Maintenance and reboot normally.

¦ Copyright IBM Corp. 1985, 1991

4.15.4 Hidden Directories and Incomplete Paths in a TCF Environment

Hidden directories are a key system function for managing heterogeneous environments. Currently, their primary use within the system is the maintenance of multiple binaries for a given program. Via hidden directories, a single name may be mapped to multiple objects, the resolution depending on the context in which the name translation is being performed. (See the *AIX Operating System Technical Reference, Volume 1* for more information on hidden directories.) AIX (and Unix) command shells also have a "path" construct that allows different command names to be resolved to different pathnames depending on the order of the order of the "path". The following discussion outlines how to avoid confusion in the interactions between hidden directories and shell "paths".

Occasionally, the same command may exist in multiple components within the "path". It is possible to alter the order of directories in the "path" and get a different default. It is recommended that when the same command name is duplicated within different directories of the "path" each hidden directory be fully populated for all machine types in the cluster. If this is not done then you may run a command locally and have it behave differently when run remotely (using **onsite**). The hidden directories can be populated with scripts like the following (this script assumes the **i386** component exists and the **i370** component will be in this script:

```
#! /bin/sh
$0@/i386 $*
```

More complex scripts can be assembled to determine the execution site or a C program can be used which will reduce startup costs if the program is short and frequently used.

If a hidden directory is not fully populated then the typical error can occur: you run the command locally but when you input **onsite** it may fail. In particular, it will fail when there is no component in the hidden directory for the specified machine. **onsite** does not change the sitepath for the new process. It simply tells the system to start the process there. The sitepath remains the same for the command, but, while the on command is initiating the task on the destination machine, only the destination machine type is effectively in the site path. The kernel design specifies a particular site for users to run the command, or fail. As an administrator setting up hidden directories and porting programs, it may be that you want to make the system provide the service, even if the task is not running on the prescribed machine.

¦ Copyright IBM Corp. 1985, 1991

4.16 Logging In Automatically

Generally, it is best to use the standard login and password system to protect your system from unauthorized access. However, if your situation does not require this security, you can modify your system so that it logs you in automatically each time you start it. To make your system log you in automatically, create a file named **/etc/autolog** that contains a valid user name (that is, a user name that is in the password file). For information about users and the password file, see "Managing User Accounts" in topic 1.2.5. For more information about **/etc/autolog**, see **autolog** in *AIX Operating System Technical Reference*.

Note: You can use the automatic login only for the main display station (the console).

¦ Copyright IBM Corp. 1985, 1991

4.17 Introduction to International Character Support

The following terms are defined for use with international character support:

Code page An ordered set of characters grouped together to support a particular language or function. Within the range of the code, each character is assigned to a unique binary valued called a *code point*.

Code point A conceptual character. The single-byte or multibyte value that identifies a single character of a code page. The same code point may be associated with different characters in different code pages. AIX uses the code points or code page to support the characters of various languages. This allows the system to support many languages at once, while remaining compatible with systems and devices that support only ASCII.

Collating order The sequence in which characters and character strings are sorted.

Collation table A table that defines the collating order.

Conversion subroutine Provides locale-specific formats and/or strings that are used by the system to translate dates, times and currencies.

Environment A memory area used for storing data items that are not part of a program but are relevant to its operation. They are stored in an array of string values with the form *name=value*. When an internationalized process begins, the **setlocale** library routine uses these strings to reference files from which it obtains tokens. These values modify the operation of routines called by the program. For details see the entry for *environment* in *AIX Operating System Technical Reference*.

Equivalence class A set of characters that are considered equal for the purposes of string search and comparison. Examples are *a, A* and ^ *A*. For instance, the user might want to search for all uses of the word *any,* whether that word appears at the beginning of a sentence or not. In this case the first letter may be *A* or *a*. Any foreign words beginning with ^ A would also be found. Provided they have been so defined in the collation table, these letters are considered to lie in the same *equivalence class*.

Extended character Any character other than a 7-bit ASCII character. An extended character may be either a single-byte character with the high order bit set or a multibyte character.

Flattened character A singlebyte extended character drawn from the pc850 code set which has been translated to a 7-bit ASCII character of similar appearance. For example, the flattened form of 'e is e. Such characters are used

¦ Copyright IBM Corp. 1985, 1991

to send pc850 extended characters to a standard ASCII printer.

Locale A term given to all the features of the process *environment* that must change to correspond to different cultural traditions. Some of these features are encoded as *environment variables*; others may be encoded as *tokens* found within an environment file. The *locale* of a process determines the character set recognized by that process and how the process handles features like the printing of dates, times and currencies.

The AIX Operating System provides international character support for European languages and Japanese. European languages are supported by the pc850 code set. Japanese is supported by the pc 932 code set.

Both of these character sets include the full range of ASCII characters. pc850 offers an additional group of accented Roman-based characters to support European languages. pc932 offers the Katakana, Hiragana, and Kanji characters used to represent Japanese.

For each of these file codes, a configuration file establishes the classifications within the character set and the sorting order for the characters. This table is configurable and is altered by the **ctab** program. Environment variables and environment files let the user or the system administrator select alternate forms of expression for items which vary with language and culture. Such culture-dependent items include date, time, and numbering formats, the characters used for currency representation, and other location-specific symbols. Appropriate defaults can be selected by nationality.

Most of this is entirely transparent to the user. Users typically attach to the same port every session and an entry in the **/etc/ports** file causes the terminal to initialize in the proper locale. Most users will spend every computing session working in the same language and with the same date, time, and language needs. The few who need to switch languages or run processes in another locale can do so by resetting a single environment variable **(LANG)**.

Every program written for AIX calls a **C** library subroutine names **setlocale** as one of its earliest activities. This sentence consults environment variables and/or environment files to determine the values for locale-specific features. If specific variables have been set in the environment, those values are used. If not, the **LANG** environment variable is consulted. **LANG** gives the name of a specific environment file in **/usr/lib/mbcs**. This file is then opened and the tokens within it provide the needed values for the setting of the *locale*.

If **LANG** is not set, or a valid environment file cannot be found, default values are used, derived from U.S. English. This default is called the **C locale**. It provides the minimum values for the operation of a program. For full details see the **setlocale** entry in the *AIX Operating System Technical Reference*.

Programs written for AIX Version 1.2 use environment variables and processing routines that begin with NL. The entire contents of the ***/usr/lib/nls*** directory has been preserved for backward compatibility. Version 1.2 NLS binaries may be used on an AIX Version 1.2.1 system, but they cannot switch locale, or process Japanese characters.

¦ Copyright IBM Corp. 1985, 1991

Subtopics
4.17.1 Features

¦ Copyright IBM Corp. 1985, 1991

4.17.1 Features

The international character support provides these features:

Extended characters
The extended characters from both pc850 and pc932 are supported as file contents, file names, and names of directories. The names of sites, users, groups and queues must always be ASCII. All AIX string processing routines support extended characters. Any program that uses them can be used to process either European or Japanese extended characters. All the standard AIX editing programs (**vi**, **ex**, **ed**, **sed**) allow Japanese multibyte characters to be entered and manipulated. The **nroff** and **troff** programs, however, cannot process any multibyte characters from any file code.

Regular expression support
Regular expression support is provided for both ASCII and extended characters. Metacharacters retain their traditional use but now can match extended as well as ASCII characters. In addition, AIX provides a new method for collating extended characters. (See **regex** in the *AIX Technical Reference*.)

Configurable collation
A collating table defines an alphabetical (sort) order, designates character case, and sets up equivalence classes. You can alter the collation order, case, and equivalence classes to support a language, dialect, site, or user.

String formats
Utilities that produce time and date information use a format selected from the environment. The **at** utility accepts input dates in the current format as well.

Numeric symbols
The currency symbol, decimal point character, and thousands-divider character can be set for the site. An example figure using the (U.S.) default is **$9,999.99**.

¦ Copyright IBM Corp. 1985, 1991

4.18 Code Point

Subtopics
4.18.1 Code Point Support for pc850
4.18.2 Code Point Support for pc932
4.18.3 Wide Code
4.18.4 Character Entry
4.18.5 File Name Length
4.18.6 Intersystems Compatibility

¦ Copyright IBM Corp. 1985, 1991

4.18.1 Code Point Support for pc850

AIX supports European languages through the pc850 file code character set. It offers ASCII characters plus single-byte extended characters.

The 7-bit ASCII characters are represented by the code points below 127. The high-order bit of these code points is always zero. Code points from 128 to 255 are devoted to single-byte extended characters. These offer most of the characters needed for Roman-based European languages. They are Roman characters with special accent marks. The high-order bit of these single-byte extended characters is always set to one. Thus there can be no confusion between the ASCII characters and the extended Latin-based characters.

Table 4-4. Code Points for pc850

Character Encoding	**Code Points**	**Description**
000xxxxx	0-31	Controls
00100000	32	Space
0xxxxxxx	33-126	7-bit ASCII Characters
01111111	127	Delete
1xxxxxxx	128-255	Extended Roman Characters

¦ Copyright IBM Corp. 1985, 1991

4.18.2 Code Point Support for pc932

In addition to the single-byte extended characters, AIX provides *multibyte character set* (MBCS) support through the pc932 code page. This is a mixed single- and double-byte character set.

When certain bit positions are on in the first byte, only a single byte is read; all bits are considered character data. The first byte offers the standard array of ASCII characters which differ by one character and are thus renamed JISCII characters. The ASCII backslash is replaced by a JISCII yen sign, which is used for the same purpose in most applications. Also offered within the first byte are a set of single-byte Katakana.

When certain other bit patterns are present in the first byte, the remaining bits of the first byte are interpreted as character data, and a second byte is read. All the bits of the second byte are interpreted as character data. By combining bits from the first and second bytes in this way, very large sets of characters can be referenced. The double-byte characters within pc932 offer Katakana, Hiragana, Kanji, and double-byte Roman characters. Altogether there are thousands of characters represented. Refer to Table 4-5, for more information.

Table 4-5. Character Bit Values for PC932

Character Encoding	Range of first byte	Range of second byte	Description	# Characters
000xxxxx	0 - 31	N/A	Controls	32
00100000	32	N/A	Space	1
0xxxxxxx	33-126	N/A	7-bit JISCII	94
01111111	127	N/A	Delete	1
10000000	128	N/A	Undefined	
100xxxxx 01xxxxxx	129 - 159	64 - 126	Double Byte	1953
100xxxxx 1xxxxxxx	129 - 159	128 - 252	Double Byte	3844
10100000	160	N/A	Unused	
1xxxxxxx	161 - 223	N/A	8-bit Katakana	63
111xxxxx 01xxxxxx	224 - 252	64 - 126	Double Byte	1827
111xxxxx 1xxxxxxx	224 - 252	128 - 252	Double Byte	3596
11111101	253	N/A	Undefined	
11111110	254	N/A	Undefined	

Copyright IBM Corp. 1985, 1991

11111111	255	N/A	Undefined	

¦ Copyright IBM Corp. 1985, 1991

4.18.3 Wide Code

pc850 and *pc932* are *file codes*. They are used to store characters and to transport text between systems. Some AIX programs also process characters internally in this file code format, but programs that do extensive manipulation of characters convert each character to a *wide code* format.

Wide code expresses every character as a uniform, four-byte object of **wchar_t** type. For backward compatibility with files created on earlier versions of the AIX system, an **NLchar** data type is also offered. It too occupies four bytes and is handled exactly the same way.

The system administrator need not be concerned with the details of **wide code**. Normally, it is only used internally. Programs read there input and produce their output only in file code format.

Normally, no files are created in wide code format. The exception is temporary files used during pipe operations. These are usually erased upon completion of the pipe.

Occasionally a pipe operation may be interrupted before completion, leaving these temporary files in existence. If the system administrator discovers temporary files with contents that cannot be read by any locale offered by the system, this is probably their source. They should be deleted.

¦ Copyright IBM Corp. 1985, 1991

4.18.4 Character Entry

Any character in any code page can be entered directly from the keyboard. Techniques for entering European characters that are not supported by the standard keyboard map are fully described in the *AIX PS/2 Keyboard Description and Character Reference*.

Entirely different techniques are used for the Japanese language. Entering Japanese characters that are not part of the keyboard map is done by special conversion routines. These allow the user to spell out Japanese words in any of several phonetic character sets and then convert the words to a pictographic character set called Kanji. This process is covered in *Using the Operating System*.

¦ Copyright IBM Corp. 1985, 1991

4.18.5 File Name Length

File and directory names have a length limit of 255 bytes. If single-byte characters (like ASCII) are used exclusively, the file name can be as long as 255 characters. If double-byte characters are used, the limit may be as low as 127 characters.

¦ Copyright IBM Corp. 1985, 1991

4.18.6 Intersystems Compatibility

The AIX Operating System is compatible with environments that do not support extended characters. It provides support for workstations, intersystem mail, networking, and program development.

AIX can communicate with other systems and clusters which use standard ASCII workstations. Site names used in intersite communications must conform to the requirements of the communication protocol (such as that used by the **uucp** command) running on the host system. They must be expressed in ASCII characters only. Sometimes a specific subset of ASCII is required. See Chapter 8, "Managing the Basic Networking Utilities" in topic 6.8 for further information.

¦ Copyright IBM Corp. 1985, 1991

4.19 Configuring the Environment

Setting up the environment for International Character Support requires that local environment variables to control locale-specific functions. This includes, at a minimum, settings for date, time, and numbering formats and a collation table. For most sites and most users, this requires setting only a single variable, **LANG**. Under normal circumstances one or more environment variable settings are placed in the user's **.profile** file.

LC_COLLATE provides a collation table that is used to sort the characters of that language. **LC_TIME** determines the data and time formats. **LC_MESSAGE** dictates the language in which program messages will be displayed. A full list of these variables can be found under "Environment" and the values to which they must be set can be found under the separate listings for each variable in *AIX Operating System Technical Reference* It is necessary to set these environment variables separately only when some special combination of locale-dependent features is needed. Ordinarily the single variable **LANG** is sufficient. Its value tells **setlocale** which environment file in **/usr/lib/mbcs** to open. These files contain tokens which are used to adjust the functioning of all the features which would ordinarily be used together in each locale. Environment files have names of the form

language_territory.codeset.en

The **setlocale** subroutine can obtain the values it seeks from three sources:

1. **LANG**. The values for all locale-dependent functions can be derived from the tokens within the environment file indicated by **LANG** Any of these values can be overwritten by the following steps.

2. **LC_MESSAGE**, or **LC_TIME**. Values for specific locale-dependent functions can be derived from the tokens within other environment files. Which files to use are indicated by the environment variables, **LC_MESSAGE** or **LC_TIME**. When these environment variables are set, **setlocale** retrieves from indicated environment files only the tokens for that category. These values take precedence over those derived in step 1.

3. Values passed to **setlocale** by the calling program. This mechanism allows a program to be "hardcoded." When set values are passed, the program always handles its locale-specific functions in the same way, irrespective of the locale in which it is run.

No matter where it gets the values, **setlocale** records the tokens in data structures within the calling program. These structures are the locale which is being set. Their contents will thereafter dictate the way the calling program handles functions determined by the culture and location in which the program is run. These structures within the program are consulted by the library routines which the program calls.

The **setlocale** subroutine also loads the appropriate collation and conversion tables at the start of each process which requires them.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

4.19.1 System Administrator's Responsibilities

The system administrator must perform a few simple tasks to manage an AIX system where users may work with international character support.

1) See that the contents of ***/usr/lib/nls*** and ***/usr/lib/mbcs*** are properly loaded on the system and are not erased.

2) Set the **TZ** variable in **/etc/environment** to establish a system-wide Time Zone setting. For details, see "Environment" in the *AIX Operating System Technical Reference*.

3) Establish an appropriate **LANG** token in ***/etc/ports*** for each user terminal. This causes the terminal to come up in an appropriate default locale.

4) Establish an appropriate **LANG** variable in the **.profile** of each user. This is especially necessary for users who will connect over the LAN, because the port to which they connect is unpredictable.

5) Set the **NLIN** and **NLOUT** variables in the **.profile** of each European language user to establish the input and output translation tables. These handle ASCII-to-EBCDIC and EBCDIC-to-ASCII translation. (Japanese characters are also converted to an EBCDIC format for printing on System/370 printers, but this is handled by the **iconv** utility.)

6) Set up special variables in the **.profile** of those users who need unusual collation sequences or date or time handling.

7) Create new collation tables (where necessary) using the **ctab** command.

¦ Copyright IBM Corp. 1985, 1991

4.19.2 Creating New Collation Tables

You can use the **ctab** program to establish a new collating sequence and/or new case conversions. The input and output files are stored in **/usr/lib/mbcs**.

A number of collation tables are shipped with the system. You can use, without change, one of the predefined table files in this directory for a language. You can also modify an existing table, or build a customized table.

The collation table files are in both source and binary format. An appropriate source file can be chosen which is close to the desired collation order. This can be modified with **ctab** to produce a new binary with the desired order. File names should generally reflect their contents; for example, **uk** for British characters, symbols and lexicographical collating sequence. An example collation table is provided for you in **/usr/lib/mbcs/example.ctab**.

For each character in a collating sequence (each **mbchar_t**), the table input file provides the following information:

For alphabetical characters, the corresponding uppercase or lowercas version of the given character. (You can assign case to any character.)

Collating sequence

The range of an equivalence class. All the characters in this clas are counted in the character range, along with this character, whenever a character is named as an end point of a character range.

You can assign a string of characters to a character equivalenc class.

The following **ctab** input conventions are used for setting up a table file:

Input starts from a standard template file containing the entir supported character set in the general order of an ISO collating sequence. Collating sequence follows the ordering of the per-character input lines. Except for ASCII characters and the characters in all supported languages, lines in the file are commented out.

Escape sequences (in C language style) are permitted in the inpu table; if a backslash does not form part of a valid escape sequence, it strips the following character of any special meaning it could have had.

One line of information is present for each character explicitl named.

Each character information line in a collation table file contains four data fields:

1. ***Subject Character***
   Identifies the character to appear in the collating order at that point. The treatment of the subject character is dependent on the type of character and its use. The subject character can be:

¦ Copyright IBM Corp. 1985, 1991

A (nonprinting) control character or a space character.

An alphabetical character (having a case equivalent character) and (optional) equivalent characters. If a string of characters is given, they are collated as a single unit.

A character to be translated using a defined translation string. The field contains a subject character followed by a **¦** (vertical bar) and a single or multiple-character string; this is a ***translation string*** that the subject character is translated to before collation. For example, to treat the character oe as equivalent to the string **oe**, the line must contain oe¦oe as its first field. However, the subject character cannot be used in the string of characters into which it is to translate. For example, **o¦oe** is an illegal construction.

2. ***Case Match***
   Identifies the matching case character for the character in field 1 if field 1 is an alphabetic character. If the field 1 character is lowercase, field 2 holds its uppercase equivalent, and visa-versa. If the character in field 1 has no equivalent, this is an empty (null) field.

3. ***Type Identifier***
   Identifies the type of character in field 1:

   If the subject character is to be treated as an alphabetic character (even if field 2 is null) this field must identify the character as lowercase or uppercase. An **l** or **L** placed in the type identifier field selects a lowercase subject character or characters. A **U** or **u** in this field selects an uppercase character (or characters).
   If the subject character is a (nonprinting) control character, a value of **C** or **c** must be placed in this field.
   If the subject character is a space (blank or tab) character, a value of **S** or **s** must be placed in this field.

4. ***Equivalence Class***
   Specifies the first character in the equivalence class of the subject character. Members of an equivalence class must be listed in consecutive lines. If this field is not specified, the group of consecutive subject characters having blank fourth fields are placed in the same equivalence class (only) if they are based on the same Roman alphabetic character.

A line beginning with the word **option** serves to change one or more of the default conditions or metacharacters built into **ctab**. This option line contains a set of **name-value** pairs; each pair is delimited by space or tab characters. Recognized **names** are:

eclass Turns equivalence class function on or off globally. The **value** must be **on** or **off**; the default is **on**.

sep Uses the **value** as the separator character between fields on an input line; the default is a colon (**:**). Tab or space characters may surround fields and separators.

trans Uses the **value** as an indicator of translation in subject character fields; the default is **1**.

¦ Copyright IBM Corp. 1985, 1991

repeat Uses the **value** as a repeat indicator meaning *same as previous character* in subject character fields; the default is a circumflex (^). A repeat character cannot be used following a translation string because the character to be translated has no inherent collating value.

comment Uses the **value** as a comment character; the default is a pound sign (#). That portion of a line to the right of a comment character is ignored.

¦ Copyright IBM Corp. 1985, 1991

4.19.3 Terminal Mapping

You can set the terminal maps for your system with the **stty** command. The **imap** parameter sets a terminal **.in filename** and the **omap** parameter sets a terminal **.out**. You must select the terminal **.in** and terminal **.out** names from the files in the **/etc/nls/termmap**.

You can establish terminal map assignments in the **/etc/ports** file. The **getty** utility uses this file at system initialization.

¦ Copyright IBM Corp. 1985, 1991

5.0 Chapter 5. Managing Multi-User Systems

Subtopics

¦ Copyright IBM Corp. 1985, 1991

5.1 Contents

¦ Copyright IBM Corp. 1985, 1991

5.2 About This Chapter

This chapter describes the structure, setup, and management of the AIX accounting system, as well as the reports it generates. This accounting system is designed to provide not only a way to bill for computer use but also a way to monitor various aspects of system operations.

Since you need system data combined and summarized by system users for billing purposes, and you need data combined and summarized by system resources for monitoring system operations, the accounting system collects detailed data on each transaction and provides tools for processing the data to produce different kinds of reports.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System,* and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

5.3 Running System Accounting

Subtopics

¦ Copyright IBM Corp. 1985, 1991

5.3.1 An Introduction to System Accounting

AIX system accounting provides for collecting and processing the following types of system data:

1. The amount of time each user spends logged into the system (***connect-time accounting***).

2. The use by each process of the processing unit, memory, and I/O resources (***process accounting***).

3. The amount of disk space occupied by each user's files.

4. The amount of printer use by each user.

5. Fees charged for materials and services.

Subtopics
5.3.1.1 Connect-Time Accounting
5.3.1.2 Process Accounting
5.3.1.3 Disk-Usage Data
5.3.1.4 Printer-Usage Data
5.3.1.5 Fees for Services and Materials
5.3.1.6 Files and Directories

¦ Copyright IBM Corp. 1985, 1991

5.3.1.1 Connect-Time Accounting

Data collection: Connect-time data are collected through cooperation between the **init** and **login** commands. When you log in, the **login** program writes a login record in the file **/etc/utmp** (actually, **<LOCAL>/utmp**).

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 in and onto which display stations. If the connect accounting file **/usr/adm/wtmp** exists, **login** adds a copy of this login record to it. The file **/usr/adm/wtmp** is actually **<LOCAL>/adm/wtmp**.

Note: **/etc/wtmp** is a symbolic link to **/usr/adm/wtmp**. **/usr/adm/wtmp** is a symbolic link to **<LOCAL>/adm/wtmp**. The **wtmp** file actually resides in **<LOCAL>/adm/wtmp**, not **/usr/adm/wtmp**.

The official AIX name for this file is **/usr/adm/wtmp**. It has been set up as a symbolic link to achieve compatibility with UNIX SVID.

When your login program ends (normally when you log out), the **init** process records the end of the session by writing another record in **/usr/adm/wtmp**. Both the login and logout records have the form described in the header file **utmp.h**. Logout records differ from login records in that they have a blank user name.

The **acctwtmp** command also writes special entries in **/usr/adm/wtmp** having to do with system shutdown and startup.

Reports: The accounting commands concerned with keeping track of each user's share of system resources write standardized ***total accounting records***, whose format is given in the header file **acct.h**. All billable accounting data originates as, or can be converted to, either the total accounting record format or a readable and editable ASCII version thereof.

With the **acctmerg** command, you can convert between the ASCII and binary formats and merge the records from different sources into single records, one for each user.

You can use the **prtacct** command to display any total accounting file. You can also use **awk** scripts to produce site-specific reports from the ASCII representation of the total accounting records. (For a brief description of the **awk** command, see *AIX Operating System Commands Reference*. For a more detailed description, see *AIX Operating System Programming Tools and Interfaces*.)

The **acctcon1** command produces connect-time and line-use records (***session records***), from login and logout records. From these session records, you can produce reports that show each individual login session, that show a summary of login sessions for each port, or that show an overall summary of login sessions for the system.

You can use the **prctmp** command to display session records with appropriate headings. The **acctcon2** command converts a collection of session records, sorted by user name, to total accounting records. You usually merge these records with the total accounting records produced by other parts of the accounting system to produce reports for each user.

You can also find out the last date on which each user logged in. The **lastlogin** command produces this report, normally as a part of the **runacct** procedure.

¦ Copyright IBM Corp. 1985, 1991

5.3.1.2 Process Accounting

Data collection: The AIX Operating System collects resource usage data for each process as it runs. The data include the user and group numbers under which the process runs, the first eight characters of the name of the command, the elapsed time and processor time used by the process, memory use, the number of characters transferred, and the number of disk blocks read or written on behalf of the process. The **accton** command causes the kernel to record this data in a specified file, usually **/local/adm/pacct**. If called without parameters, **accton** turns on the recording of data.

Other commands relating to the recording of process accounting data are **startup**, **shutacct**, **dodisk**, **ckpacct**, and **turnacct**. These commands are simple shell files that build on the other commands discussed in this section.

Reports: In addition to providing billing information, process accounting data also provides a rich source of information that you can use to monitor the use of system resources.

The **acctcms** command summarizes resource use by command name. It provides information on how many times each command was run, how much processor time and memory it used, and how intensive that use was while the command was running (its ***hog factor***). It can also produce long-term statistics on system utilization. Thus, it provides answers to questions about total system usage, what commands are used frequently enough to be worth optimizing, and so on.

The **acctcom** command handles the same data as **acctcms**, but for a different purpose. Whereas **acctcms** summarizes the data by command, **acctcom** provides detailed information about each running of each process. With it, 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 during which the process ended, the name of the command, the user or group that ran the process, and the port from which the process was run.

The **acctprc1** and **acctprc2** commands produce billing information in two forms. **acctprc1** translates the user ID into a user name and writes ASCII records containing the chargeable items (***prime*** and ***nonprime CPU time***, ***mean memory size***, and ***I/O*** data). **acctprc2** transforms these records into total accounting records.

¦ Copyright IBM Corp. 1985, 1991

5.3.1.3 Disk-Usage Data

Most accounting information on AIX system is collected continuously as the resources are consumed. Disk usage data is an exception. Rather than maintain a running record of disk space consumption, you use the **dodisk** command to collect this data periodically. **dodisk** examines file systems and summarizes space use by user name. You then use **acctmerg** to produce total accounting records from the output of **dodisk**.

The **dodisk** command charges a user name for links to files found under that user's login directory. It divides the charge for a file evenly among links to that file. The decision to charge for links, rather than for files owned, is motivated by two considerations:

The cost of a file should be borne by all who use it. The mos reasonable way to determine who uses a file is to see who has links to it.

You can create a file, allow another user to link to it, and the remove that person's own link to it. Once the user has relinquished access to the file, it seems unfair to charge for it.

¦ Copyright IBM Corp. 1985, 1991

5.3.1.4 Printer-Usage Data

Collection of printer usage data is a cooperative effort between the **print** command that enqueues files to be printed and the queuing daemon that schedules the printing. The **print** command enqueues the user name and number of the user doing the printing, along with the name of the file to be printed and other user-specified flags. If print accounting is turned on, the system keeps a record of what has been printed. After each file is printed, **qdaemon** writes an ASCII record to a file. The name of this file is specified in the **/etc/qconfig** file for each printer and is usually **/usr/adm/qacct**. Add the field, **acctfile=/usr/adm/qacct** to the printer queue stanza in the **/etc/qconfig** file, and create the file **/local/adm/qacct** at the site where printer is located.

Note: **/usr/adm/qacct** is a symbolic link to **<LOCAL>/adm/qacct**.

This ASCII record contains the user name, user number, and number of pages printed. You can sort these records, convert them to total accounting records, and use **acctmerg** to combine them with data from other sources.

For more information about this accounting file, see **qconfig** in the *AIX Operating System Technical Reference*.

¦ Copyright IBM Corp. 1985, 1991

5.3.1.5 Fees for Services and Materials

You use the **chargefee** command to charge users for services; such as, file restores, or consulting, or for certain materials. **chargefee** writes an ASCII total accounting record in the file **/usr/adm/fee**.

Note: **/usr/adm/fee** is a symbolic link to **<LOCAL>/adm/fee**.

¦ Copyright IBM Corp. 1985, 1991

5.3.1.6 Files and Directories

The directory **/usr/lib/acct** contains all of the C Language programs and shell procedures necessary to run AIX system accounting. The accounting system data files belong to user **adm** (currently user ID 4), and all active data files (**wtmp**, **pacct**, etc.), reside in this user's home directory, **/usr/adm**. All reports and summary files are stored in the subdirectories of **/usr/adm/acct**: **nite**, **sum**, and **fiscal** (see Figure 5-1). Note that in Figure 5-1, **/usr/adm/acct** is a symbolic link to **<LOCAL>/adm/acct**.

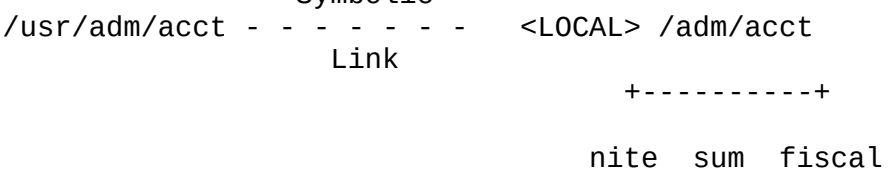


Figure 5-1. Accounting Directory Structure

The **nite** directory contains files that **runacct** reuses daily. The **sum** directory contains the cumulative summary files that **runacct** updates. Finally, the **fiscal** directory contains the summary files that **monacct** creates.

¦ Copyright IBM Corp. 1985, 1991

5.3.2 Setting Up the Accounting System

In order to automate the operation of this accounting system, you need to do several things. System directory and file names must be in ASCII for successful communication across locales.

To Set up System Accounting

1. Enter: **/usr/lib/acct/nulladm wtmp pacct**.
2. Update the file **/usr/lib/acct/holidays**.
3. Turn on process accounting through the file **/etc/rc**.
4. In the file **/etc/filesystems**, identify file systems to include in disk accounting.
5. Specify the data file for printer usage data in the file **/etc/qconfig**.
6. Schedule daily accounting in the file **/usr/spool/cron/crontabs/adm**.
7. Schedule monthly or fiscal summaries in the file **/usr/spool/cron/crontabs/adm**.
8. Set the PATH shell variable in **/.profile** to include **/usr/lib/acct**.

Subtopics
5.3.2.1 Accounting Setup Procedure

¦ Copyright IBM Corp. 1985, 1991

5.3.2.1 Accounting Setup Procedure

1. Run the **nulladm** procedure as follows:

   a. Log in as user **root**.

   b. Change to the proper directory. Enter:

      ```
      cd /usr/adm
      ```

   c. Enter:

      ```
      /usr/lib/acct/nulladm wtmp pacct
      ```

   The **nulladm** procedure ensures that each file has the proper access permission code (664).

2. Update the file **/usr/lib/acct/holidays**, if necessary. This file contains the time table that the accounting system uses to distinguish prime time and nonprime time. Edit this table to reflect your holiday schedule for the year and to reflect the hours during each day that you want to designate as prime time.

   This file contains three types of entries:

   a. Comment lines:

      Comment lines may appear anywhere in the file as long as the first character in the line is an asterisk (*****).

   b. Year Designation Line:

      This line must be the first data line (that is, the first line that is not a comment) and may appear only once. It consists of three fields of four digits each (leading blanks ignored). These three fields contains:

      1) The year

      2) The time (**hhmm**) at which prime time begins

      3) The time (**hhmm**) at which prime time ends

      Use a 24-hour clock to specify the time. You can enter the hour of midnight as either 0000 or 2400.

      For example, to specify the year as 1984, prime time as beginning at 8:00 a.m., and nonprime time as beginning at 5:00 p.m., enter the following line:

      ```
      1984    0800    1700
      ```

   c. Company Holiday Lines: These entries come after the year designation line. Each of these lines, composed of four fields, has the following general format:

      ```
      Year-day  Month  Day  Description of Holiday
      ```

      The Year-day field contains a number from 1 through 366 that indicates the day of the year on which the holiday falls (leading

¦ Copyright IBM Corp. 1985, 1991

blanks ignored). The other three fields--Month, Day, and Description of Holiday--are treated as comments by other accounting programs. These fields identify to other users what holiday is designated by the number in the Year-Day field. Figure 5-2 shows a sample **holidays** file.

```
* Curr  Prime  Non-Prime
* Year  Start  Start
*
  1985  0830   1700
*
* Day of    Calendar    Company
* Year      Date        Holiday
*
    1       Jan 1       New Year's
    2       Jan 2       day after
   46       Feb 15      Wash. Birthday
  152       May 31      Memorial Day
  186       Jul 5       Indep. Day
  249       Sep 6       Labor Day
  332       Nov 28      Thanksgiving
  333       Nov 29      day after
  358       Dec 24      Christmas Eve
```

Figure 5-2. Sample Holidays File

3. To turn on process accounting, add the following line to **/etc/rc**, if it is not already there (or delete the comment symbol if it is present):

   ```
   /usr/lib/acct/startup
   ```

   The **startup** shell procedure calls **acctwtmp** to record in **/usr/adm/wtmp** the time that accounting was turned on. It calls **turnacct on**, which cleans up the previous day's accounting files by calling the **remove** shell procedure.

   The **remove** procedure deletes all **/usr/adm/acct/sum/wtmp***, **/usr/adm/acct/sum/pacct***, and **/usr/adm/acct/nite/lock*** files.

4. In **/etc/filesystems**, add the following line to each stanza that defines a file system you want to include in system accounting:

   ```
   account = true
   ```

5. Enable printer usage accounting by adding the following line to the queue stanza in the file **/etc/qconfig**:

   ```
   acctfile = /usr/adm/qacct
   ```

6. To automatically run daily accounting through **cron**, add the following lines to the file **/usr/spool/cron/crontabs/adm** (**/usr/spool** is actually a symbolic link to **/machinename/spool**). (See the descriptions of **cron** and **crontab** in *AIX Operating System Commands Reference* for details about submitting jobs to **cron**.) If these times do not fit the hours that your system is operational, adjust your entries accordingly.

¦ Copyright IBM Corp. 1985, 1991

```
0 2 * * 4 /usr/lib/acct/dodisk
5 * * * * /usr/lib/acct/ckpacct
0 4 * * 1-6 /usr/lib/acct/runacct
              2>/usr/adm/acct/nite/fd2log
```

The first entry schedules the running of **dodisk** for 2:00 a.m. (**0 2**), each Thursday (**4**). The second entry schedules **ckpacct** for 5 minutes past every hour (**5 ***) of every day (*). The third entry schedules **runacct** for 4:00 a.m. (**0 4**) every Monday through Saturday (**1-6**).

The **dodisk** procedure calls **diskusg** and **acctdisk** to write disk usage records to the file **dacct**. It stores this file in **/usr/adm/acct/nite**.

The **ckpacct** procedure monitors the size of **/local/adm/pacct**. If the file is larger than 500 blocks, **ckpacct** calls **turnacct switch** to copy the current **pacct** file to **pacctx**, where **x** is an integer that is increased each time **turnacct switch** is called. The advantage of having several smaller **pacct** files becomes apparent when you must restart **runacct** after a failure in processing these records.

The **runacct** procedure processes the active data files (including **wtmp**, **pacct**) to produce command summaries and usage summaries sorted by user name. For a detailed description of **runacct**, see "Running Daily Accounting - The runacct Command" in topic 5.3.3.

7. To automatically perform monthly merging of accounting data, add the following line to the file **/usr/spool/cron/crontabs/adm**:

   ```
   15 5 1 * * /usr/lib/acct monacct
   ```

   Be sure to schedule this procedure at a time that allows **runacct** sufficient time to finish when ordinary user activity is absent or minimal. This will, on the first day of each month, create monthly accounting files with the entire month's data. For a detailed description of **monacct** reports, see "Daily Command Summary and Monthly Total Command Summary Reports" in topic 5.3.4.2.

8. Set the PATH shell variable in **/.profile** as follows:

   ```
   PATH=/usr/lib/acct:/bin:/etc:/usr/bin:
   export PATH
   ```

   This simplifies the running of individual accounting commands when you are logged in as **adm**.

¦ Copyright IBM Corp. 1985, 1991

5.3.3 Running Daily Accounting - The runacct Command

The **runacct** command is the main daily accounting shell procedure. Normally initiated by **cron** during nonprime hours, **runacct** processes connect, fee, disk, and process accounting files. It also prepares daily and cumulative summary files for use by the **prdaily** command or for billing purposes.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

5.3.3.1 Output Files

The following files produced by **runacct** are of particular interest. See Table 5-1 in topic 5.3.5 for a description of Accounting file formats.

/usr/adm/acct/nite/lineuse
This file 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.

/usr/adm/acct/nite/daytacct
This is the total accounting file for the previous day.

/usr/adm/acct/sum/tacct
This file contains the accumulation of each day's **nite/daytacct** file and can be used for billing purposes. **monacct** restarts it each month or fiscal period.

/usr/adm/acct/sum/cms
This file contains the accumulation of each day's command summaries. **monacct** uses this binary version of the file and restarts it. The ASCII version is **nite/cms**.

/usr/adm/acct/sum/daycms
This file contains the daily command summary. An ASCII version is stored in **nite/daycms**.

/usr/adm/acct/sum/loginlog
This file contains a record of the last time each user ID was used.

/usr/adm/acct/sum/rprtmmdd
This file contains a copy of the daily report saved by **runacct**.

¦ Copyright IBM Corp. 1985, 1991

5.3.3.2 Operational States

The **runacct** procedure takes care to protect active data files by monitoring for operational errors. If it detects one, it writes a message to **/dev/console** (the command line in **crontabs/adm** should redirect these messages to the file **/usr/adm/acct/nite/accterr**), mails messages to the users root and **adm**, removes locks, saves diagnostic files, and exits.

To make **runacct** easier to restart after an error, its operation is divided into clearly defined stages, or ***operational states***. It records its progress through these states by writing descriptive messages in the file **/usr/adm/acct/nite/active**. It also writes its current state to the file **/usr/adm/acct/nite/statefile**. The last operation in each state is to write the next state to **statefile**. After it completes each state, **runacct** reads **statefile** for the next state to process.

These states are processed as follows:

State	Actions
SETUP	This state calls **turnacct switch**. It moves the process accounting files **/local/adm/pacct?** to **/local/adm/Spacct?.mmdd**. It also calls **acctwtmp** to write the current time as the last record in **/usr/adm/wtmp** and then moves the file to **/usr/adm/acct/nite/wtmpmmdd**.
WTMPFIX	This state calls **wtmpfix** to check **nite/wtmp** for accuracy. Some date changes cause problems with the **acctcon1** command, so **wtmpfix** attempts to adjust the time stamps in the **wtmp** file if a date change record appears.
CONNECT1	This state calls **acctcon1** to write session records from **wtmp** to the **ctmp** file (see Table 5-1 in topic 5.3.5). It also creates the **lineuse** file and the **reboots** file. Together, these three files contain all of the login and logout records found in the **wtmp** file.
CONNECT2	This state calls **acctcon2** to convert **ctmp** files to total accounting files. It sends these to **acctmerg** to produce the **ctacctmmdd** file. This file contains all the connect accounting records (see Table 5-1 in topic 5.3.5).
PROCESS	This state calls **acctprc1** and **acctprc2** to convert the process accounting files **/local/adm/Spacct?.mmdd** into total accounting records in **ptacct?.mmdd** (see Table 5-1 in topic 5.3.5). The **Spacct** and **ptacct** files are correlated by number so that if **runacct** fails in this state, you do not need to reprocess all **Spacct** files. **Note:** When restarting **runacct** in the PROCESS state, remove the last **ptacct** file as it will not be complete.
MERGE	This state calls **acctmerg** to merge all process accounting records with the connect accounting records, writing these to the **daytacct** file (see Table 5-1 in topic 5.3.5).
FEES	This state calls **acctmerg** to merge any records from the

¦ Copyright IBM Corp. 1985, 1991

file **fee** into **daytacct**.

DISK — This state calls **acctmerg** to merge **dacct** with **daytacct**.

QUEUEACCT — This state sorts the queue (printer) usage records, converts them into total accounting records, and calls **acctmerg** to merge them with the other total accounting records in **daytacct**.

MERGETACCT — This state calls **acctmerg** to merge **daytacct** with **sum/tacct**, the cumulative total accounting file. Each day, **daytacct** is saved as **sum/tacctmmdd**, so that **sum/tacct** can be recreated in the event it becomes damaged or lost.

CMS — This state merges the current day's command summary with the cumulative command summary file **sum/cms**, producing both ASCII and binary summary files (see Table 5-1 in topic 5.3.5).

USEREXIT — If the shell file **/local/adm/siteacct** exists, this state calls it to perform site-dependent processing.

CLEANUP — This state cleans up temporary files, runs **prdaily** and saves its output in **sum/rprtmmdd**, removes the locks, and exits.

¦ Copyright IBM Corp. 1985, 1991

5.3.3.3 Recovering from Failure

The **runacct** procedure can fail for a variety of reasons. The most common reasons are: the system goes down, the file system **/local/adm** runs out of space, or the **wtmp** file has records with inconsistent date stamps. If **runacct** fails, do the following:

1. Check the file **/usr/adm/acct/nite/activemmdd** first for error messages.

2. If both the **active** file and lock files exist in the **acct/nite/** directory, check the file **accterr** (the file you redirected error messages to when **cron** called **runacct**).

3. Perform any actions needed to eliminate errors (see "Fixing Damaged Files" in topic 5.3.3.5).

4. Restart **runacct** (see the following section, "Restarting runacct" in topic 5.3.3.4).

¦ Copyright IBM Corp. 1985, 1991

5.3.3.4 Restarting runacct

If you call **runacct** with no command line parameters, it assumes that this is the first invocation of the day. You need to include the parameter **mmdd** if you are restarting **runacct**. This parameter specifies the month and day for which **runacct** is to rerun accounting. If you do not specify a **state**, **runacct** determines the entry point for processing by reading **statefile**. To override **statefile**, specify the desired **state** on the command line.

Note: When you perform the following tasks, you may need to use the full path name of **runacct** (**/usr/lib/acct/runacct**) rather than the simple command name.

1. To start **runacct**, enter:

   ```
   nohup runacct 2>/usr/adm/acct/nite/accterr &
   ```

   This entry causes **runacct** to ignore all **INTERRUPT** and **QUIT WITH DUMP** signals while it performs its processing in the background. It also redirects all standard error output (file descriptor 2) to the file **/usr/adm/acct/nite/accterr**.

2. To restart **runacct**:

   ```
   nohup runacct 0601 2>>/usr/adm/acct/nite/accterr &
   ```

   This restarts **runacct** for day of June 1 (**0601**). **runacct** reads the file **/usr/adm/acct/nite/statefile** to find out which state it should begin with. All standard error output is appended to the file **/usr/adm/acct/nite/accterr**.

3. To restart **runacct** at a specified state, in this case the MERGE state, enter the following:

   ```
   nohup runacct 0601 MERGE 2>>/usr/adm/acct/nite/accterr &
   ```

Note: Use ASCII characters for file names to ensure that users of different locales can access these files.

¦ Copyright IBM Corp. 1985, 1991

5.3.3.5 Fixing Damaged Files

Unfortunately, a file occasionally becomes damaged or lost. Certain files must be fixed in order to maintain the integrity of the accounting system.

Fixing wtmp Errors: The **wtmp** files seem to cause the most problems in the day-to-day operation of the accounting system. When the date is changed and the AIX Operating System is in multiuser mode, a set of date change records is written to **/usr/adm/wtmp**.

The **wtmpfix** command is designed to adjust the time stamps in the **wtmp** records when a date change is encountered. However, some combinations of date changes and system restarts will slip through **wtmpfix** and cause **acctcon1** to fail. The following steps show how to patch up a **wtmp** file:

```
cd  /usr/adm/acct/nite
fwtmp  <wtmp.mmdd  >wtmp.new
ed  wtmp.new

            (Delete damaged records or
            delete all records from beginning
            up to the date change)

fwtmp  -ic   <wtmp.new  >wtmp.mmdd
```

The **fwtmp** command without a flag converts a binary **utmp** file to an ASCII file that you can edit. **fwtmp** with the **-ic** flag converts the ASCII file back to the binary **utmp** format.

If the **wtmp** file is beyond repair, use the **nulladm** command to create an empty **wtmp** file. This prevents any charging of connect time.

Fixing tacct Errors: If you are using the accounting system to charge users for system resources, the validity of **/usr/adm/acct/sum/tacct** is important. Occasionally, invalid **tacct** records appear that contain negative numbers, duplicate user numbers, or a user number of 65,535. If this happens, first use the **prtacct** command to check the **sum/tacctprev** file. If it looks all right, the latest **/usr/adm/acct/sum/tacct.mmdd** should be patched up, then **/usr/adm/acct/sum/tacct** recreated. A simple patchup procedure would be:

```
cd  /usr/adm/acct/sum
acctmerg -v  <tacct.mmdd  >tacct.new
ed tacct.new

(Remove the bad records.  Write duplicate user number records to another
file.)

acctmerg -i  <tacct.new  >tacct.mmdd
acctmerg tacctprev  <tacct.mmdd  >tacct
```

Remember that the **monacct** procedure removes all the **tacct.mmdd** files; therefore, **/usr/adm/acct/sum/tacct** can be recreated by merging these files together.

¦ Copyright IBM Corp. 1985, 1991

5.3.4 Accounting Reports

The **runacct** procedure produces five basic reports. They cover the areas of connect accounting, daily use of system resources by each user, command usage reported by daily and monthly totals, and a report of the last time users were logged in.

The following sections describe the reports and the meaning of their tabulated data.

Subtopics
5.3.4.1 Daily Report
5.3.4.2 Daily Command Summary and Monthly Total Command Summary Reports
5.3.4.3 Last Login

¦ Copyright IBM Corp. 1985, 1991

5.3.4.1 Daily Report

The Daily Report has two main sections, which can be divided into Part 1 and Part 2.

Part 1: Data for Part 1 is stored in **/usr/adm/acct/sum/rpt***mmdd*. The first line identifies the period reported on. The beginning time for each report is the time the last accounting report was produced. The ending time is the time the current accounting report was produced. This is followed by a log of system boots, shutdowns, power fail recoveries, and any other record written to **/usr/admwtmp** by **acctwtmp**.

Then the report breaks down line use. The **TOTAL DURATION** entry tells how long the system was in the multiuser state (that is, able to be accessed through terminal lines). The report fields are:

LINE The terminal line or access port.

MINUTES The total number of minutes that line was in use during the accounting period.

PERCENT The total number of **MINUTES** the line was in use divided into the **TOTAL DURATION**.

 # SESS The number of times this port was accessed for a login session.

 # ON The same as **# SESS**.

 # OFF The total number of both logouts and interrupts that occur on a line.

During real time, you should monitor **/usr/adm/wtmp**. If it increases rapidly, run **acctcon1** to see which line is the noisiest. System performance is affected as the interrupt rate increases.

Part 2: The second part of the Daily Report gives a breakdown of daily system resource use by user. The data for this report is kept in **/usr/adm/acct/sum/nite/lineuse**. Its data consists of:

UID The user's identification.

LOGNAME The id name of the user. There can be more than one user name for a single user ID; LOGNAME identifies which one is referred to.

PRI_MEM An abbreviation for "prime memory." A cumulative measure of the amount of memory used during prime time (as defined in **/usr/lib/acct/holidays**). Prime time is usually the hours between 8:00 am and 5:00 pm, except for certain holidays, which are non-prime all day long. The amounts shown reflect kilobyte segments of memory used per minute.

NPRI_MEM An abbreviation for "non-prime memory." A cumulative measure of the amount of memory used during **non** prime time (as defined in **/usr/lib/acct/holidays**.) Non-Prime time is usually the hours before 8:00 am and after 5:00 pm, except for certain holidays, which are non-prime all day long. The amounts shown reflect kilobyte segments of memory used per minute.

¦ Copyright IBM Corp. 1985, 1991

PRI_CONNECT The amount of time that the user was logged onto the system during prime time.

NPRI_CONNECT The amount of time that the user was logged onto the system during **non**-prime time.

DSK BLOCKS The number of disk blocks used.

PRINT Represents charges accrued by printing.

FEES The total charged against the user by the **chargefee** command.

¦ Copyright IBM Corp. 1985, 1991

5.3.4.2 Daily Command Summary and Monthly Total Command Summary Reports

These two reports, **/usr/adm/acct/nite/cms** and **/usr/adm/acct/nite/daycms**, are virtually the same except that the Daily Command Summary only reports on the current accounting period while the Monthly Total Command Summary tells the story for the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last running of **monacct**.

These reports are sorted by **TOTAL KCOREMIN**, which is an arbitrary yardstick but often a good one for calculating drain on a system.

COMMAND NAME The name of the command. Unfortunately, all shell procedures are lumped together under the name **sh** since only object modules are reported by the process accounting system. You should monitor the frequency of programs called **a.out** or **core** or any other name that does not seem quite right. **acctcom** is also a good tool to use to determine who ran a specific command and to determine if superuser authority was used.

NUMBER CMDS The total number of invocations of this particular command.

TOTAL KCOREMIN The total cumulative measurement of the amount of kilobyte segments of memory used by a process per minute of run time.

TOTAL CPU-MIN The total processing time this program has accumulated.

TOTAL REAL-MIN The total ***real-time*** (wall clock) minutes this program has accumulated. This total is the actual time that you must wait for a system prompt, as opposed to starting a process in the background.

MEAN CPU-MIN Total processing time over elapsed time in minutes.

MEAN SIZE-K This is the mean of the **TOTAL KCOREMIN** over the number of invocations reflected by **NUMBER CMDS**.

HOG FACTOR A relative measurement of the ratio of system availability to system utilization. It is computed by the formula:

(total processor time) / (elapsed time)

This gives a relative measure of the total available processor time consumed by the process during its execution.

CHARS TRNSFD The total count of the number of characters transferred by the **read** and **write** system calls. (This total may be a negative number.)

BLOCKS READ A total count of the physical block reads and writes that a process performed.

¦ Copyright IBM Corp. 1985, 1991

5.3.4.3 Last Login

This report, stored in **/usr/adm/acct/sum/loginlog**, simply gives the date when a particular user name was last used. This could be a good source for finding likely candidates for the archives or for getting rid of unused logins and login directories.

¦ Copyright IBM Corp. 1985, 1991

5.3.5 Accounting File Formats

Table 5-1 provides a description of the formats of the major accounting data and summary data files.

Table 5-1. Accounting File Formats

File Name	Record Format	Contents
wtmp	utmp.h	Login records: 1. user number 2. user name 3. device name 4. process ID 5. entry type 6. exit status 7. date / time
ctmp	ASCII	Connect records: 1. user number 2. user name 3. device name 4. prime connect time 5. nonprime connect time 6. session starting time 7. starting date
pacct* **Spacct***	acct.h	Process records: 1. beginning time 2. user time 3. system time 4. elapsed time 5. memory used 6. chars transferred 7. blocks read / written 8. command name
daytacct **sum/tacct**	tacct (acct.h)	Total accounting records: 1. user number 2. user name 3. total kcore minutes 4. total connect time 5. total disk usage 6. number of processes 7. number of login sessions 8. number of disk samples 9. fees for special services 10. total chars transferred 11. total blocks read / written 12. queuing system charges
ptacct	tacct	Process total accounting records: 1. user number 2. user name

Copyright IBM Corp. 1985, 1991

		3. total kcore minutes 4. number of processes 5. total chars processed 6. total blocks read / written
ctacct	tacct	Connect total accounting records: 1. user number 2. user name 3. total connect time
cms **daycms**	binary & ASCII	Command summary records: 1. command name 2. number of times called 3. total kcore minutes 4. total processor minutes 5. total read minutes 6. mean memory size 7. mean processor size 8. Hog factor

¦ Copyright IBM Corp. 1985, 1991

5.3.6 Accounting System Files

¦ Copyright IBM Corp. 1985, 1991

5.3.7 Files in the /local/adm Directory

diskdiag	Diagnostic output during the execution of disk accounting programs.
dtmp	Output from the **acctdusg** command.
fee	Output from the **chargefee** command, in ASCII **tacct** records.
pacct	Active process accounting file.
Spacct?.MMDD	Process accounting files for **MMDD** during the execution of **runacct**.

¦ Copyright IBM Corp. 1985, 1991

5.3.8 Files in the /usr/adm/acct/nite Directory

active	Used by **runacct** to record progress and print warning and error messages. The file **activeMMDD** is a copy of **active** made by **runacct** after it detects an error.
cms	ASCII total command summary used by the **prdaily** command.
ctacct.MMDD	Connect total accounting records.
ctmp	Connect session records.
daycms	ASCII daily command summary used by the **prdaily** command.
daytacct	Total accounting records for one day.
dacct	Disk total accounting records created by **dodisk**.
accterr	Diagnostic output produced during the execution of **runacct**.
lastdate	The last day **runacct** executed in **date +%m%d** format.
lock **lock1**	Used to control serial use of **runacct**.
lineuse	tty line usage report used by **prdaily**.
log	Diagnostic output from **acctcon1**.
logmmdd	Same as **log** after **runacct** 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 **runacct**.
tmpwtmp	wtmp file corrected by **wtmpfix**.
wtmperror	Contains **wtmpfix** error messages.
wtmperrmmdd	Same as **wtmperror** after **runacct** detects an error.
wtmp.mmdd	The previous day's **wtmp** file.

¦ Copyright IBM Corp. 1985, 1991

5.3.9 Files in the /usr/adm/acct/sum Directory

cms	Total command summary file for the current fiscal period in binary format.
cmsprev	The command summary file without the latest update.
daycms	The command summary file for the previous day in binary format.
loginlog	The file created by **loginlog**.
pacct.mmdd	Concatenated version of all **pacct** files for **mmdd**. This file is removed after system startup by the **remove** procedure.
rprtmmdd	The saved output of **prdaily**.
tacct	The cumulative total accounting file for the current fiscal period.
tacctprev	The same as **tacct** without the latest update.
tacctmmdd	The total accounting file for **mmdd**.
wtmp.mmdd	The saved copy of the **wtmp** file for **mmdd**. This file is removed after system startup by the **remove** procedure.

¦ Copyright IBM Corp. 1985, 1991

5.3.10 Files in the /usr/adm/acct/fiscal Directory

cms?	The total command summary file for fiscal period **?** in binary format.
fiscrpt?	A report similar to that of **prdaily** for fiscal period **?**.
tacct?	The total accounting file for fiscal period **?**.

¦ Copyright IBM Corp. 1985, 1991

5.4 Using the System Activity Package

The AIX Operating System contains a number of counters that are incremented as various system actions occur. The system activity package reports system-wide measurements, including central processing unit utilization, disk and tape I/O activities, terminal device activity, buffer usage, system calls, system switching, file-access activity, queue activity, and message and semaphore activities.

The package provides three commands that generate various types of reports. Procedures that automatically generate daily reports are also included. The four functions of the activity package are:

The **sar** command
: This command allows you to generate system activity reports in real time and to save system activities in a file for later use.

The **sag** command
: This command displays system activity in a graphical form.

The **timex** command
: This command is a modified **time** command that times a process and also optionally reports concurrent system activity and process accounting activity.

Daily Reports
: Procedures are provided for sampling and saving system activities in a data file periodically and for generating the daily report from the data file.

The system activity information reported by this package is derived from a set of system counters located in the AIX Operating System kernel. These system counters are described under "System Activity Counters" in topic 5.4.1. "System Activity Commands" in topic 5.4.2 describes the commands provided by this package. The procedure for producing daily reports is described under "System Activity Daily Reports" in topic 5.4.3.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

5.4.1 System Activity Counters

The AIX Operating System manages a number of counters that record various activities and provide the basis for the system activity reporting system. The data structure for most of these counters is defined in the **sysinfo** structure in **/usr/include/sys/sysinfo.h** (see "System Activity Data Structures and File Formats" in topic 5.4.4). The system table overflow counters are kept in the **_syserr** structure. The device activity counters are extracted from the device status tables.

The following is a list of the system activity counters sampled by the system activity package:

CPU time counters
At each clock interrupt, the system increments one of four time counters (**cpu[]**), depending on the mode the CPU is in at the interrupt (idle, user, kernel, and wait for I/O completion).

lread and lwrite
The **lread** and **lwrite** counters contain the count of logical read and write requests issued by the system to block devices.

bread and bwrite
The **bread** and **bwrite** counters contain a count of the number of times data are transferred between the system buffers and the block devices. The ratio of block I/O to logical I/O is a common measure of the effectiveness of system buffering.

phread and phwrite
The **phread** and **phwrite** contains a count of the read and write requests issued by the system to raw devices.

pswitch and syscall
These counters are related to the management of multiprogramming. **syscall** is incremented every time a system call is invoked. The numbers of invocations of **read**, **write**, **fork**, and **exec** system calls are kept in counters **sysread**, **syswrite**, **sysfork**, and **sysexec**. **pswitch** counts the times the **switcher** was invoked, which occurs when:

1. A system call resulted in a road block.

2. An interrupt occurred resulting in awakening a higher priority process.

3. A one-second clock interrupt occurs.

iget, namei, and dirblk
These counters apply to file-access operations. **iget** and **namei**, in particular, are the names of AIX Operating System routines. The counters record the number of times the respective routines are called.

The **namei** routine performs file system path searches. It searches the various directory files to get the associated i-number of a file corresponding to a special file.

The **iget** routine locates the inode entry of a file. It first searches the inode table in memory. If the inode entry is not in the table, the **iget** routine gets the inode from the file

¦ Copyright IBM Corp. 1985, 1991

system where the file resides and enters it in the inode table in memory. **iget** returns a pointer to this entry.

The **namei** routine calls **iget**, but other file access routines also call **iget**. Therefore, counter **iget** is always greater than counter **namei**.

Counter **dirblk** records the number of directory block reads issued by the system.

runque and runocc

These counters record queue activities. They are implemented in the **clock.c** routine. The clock routine periodically examines the process table to see whether any processes are in memory and in ready state. If so, the system increments the counter **runocc** and adds the number of such processes to counter **runque**.

readch and writech

The **readch** and **writech** counters record the total number of bytes (characters) transferred by the **read** and **write** system calls.

Monitoring terminal device activities

There are six bytes monitoring terminal device activities. **rcvint**, **xmtint**, and **mdmint** are counters measuring hardware interrupt occurrences for receiver, transmitter, and modem individually. **rawch**, **canch**, and **outch** count the number of characters in the raw queue, canonical queue, and output queue. Characters generated by devices operating in the "cooked" mode, such as terminals, are counted in both **rawch** and (as edited) in **canch**; but characters from raw devices, such as communication processors, are counted only in **rawch**.

msg and sema counters

These counters record message sending and receiving and semaphore operations.

Monitoring I/O activities

Four counters are kept for each disk or tape drive in the device status table. Counter **io_ops** is incremented when an I/O operation has occurred on the device. It includes block I/O, and physical I/O. **io_bcnt** counts the amount of data transferred between the device and memory in 512-byte units. **io_act** and **io_resp** measure the active time and response time for a device in time ticks summed over all I/O requests that have completed for each device. The device active time includes the device seeking, rotating, and data transferring times, while the response time of an I/O operation is the time the I/O request is queued to the device to the time when the I/O completes.

inodeovf, fileovf, textovf, and procovf

These counters are extracted from the **_syserr** structure. When an overflow occurs in any of the inode, file, text, and process tables, the corresponding overflow counter is incremented.

¦ Copyright IBM Corp. 1985, 1991

5.4.2 System Activity Commands

The system activity package provides three commands for generating various system activity reports and one command for profiling disk activities. These tools facilitate observation of system activity during:

A controlled standalone test of a large syste

An uncontrolled run of a program to observe the operating environmen

Normal production operation

The **sar** and **sag** commands permit you to specify a sampling interval and number of intervals for examining system activity and then to display the observed level of activity in tabular or graphical form. The **timex** command reports the amount of system activity that occurred during the precise period of execution of a timed command.

Subtopics
5.4.2.1 The sar Command
5.4.2.3 The timex Command

¦ Copyright IBM Corp. 1985, 1991

5.4.2.1 The sar Command

The **sar** command can be used in the following two ways:

When you specify the frequency parameters **interval** and **number**, **sar** invokes the data collection program **sadc** to sample the system activity counters in the operating system every **interval** seconds for **number** intervals and generates system activity reports in real-time. Generally, it is desirable to include the option to save the sampled data in a file for later examination. For the format of this file, see "System Activity Data Structures and File Formats" in topic 5.4.4. In addition to the system counters, a time stamp is also included.

If you do not supply frequency parameters, **sar** generates system activity reports for a specified time interval from an existing data file that was created by **sar** at an earlier time.

A convenient usage is to run **sar** as a background process, saving its samples in a temporary file but sending its standard output to **/dev/null**. Then, an experiment is conducted after which the system activity is extracted from the temporary file.

See **sar** in *AIX Operating System Commands Reference* for a description of all flags and usage.

¦ Copyright IBM Corp. 1985, 1991

5.4.2.2 The sag Command

The **sag** command displays system activity data graphically. It relies on the data file produced by a prior run of **sar**, from which any column of data or the combination of columns of data can be plotted. A fairly simple but powerful command syntax allows the specification of cross plots or time plots. Data items are selected using the **sar** column header names.

See **sag** in *AIX Operating System Commands Reference* for a description of all flags and usage.

¦ Copyright IBM Corp. 1985, 1991

5.4.2.3 The timex Command

The **timex** command is an extension of the **time** command. Without flags, **timex** behaves like **time**. In addition to giving the time information, it can also display a system activity report and a process accounting report. For all the flags available, see *AIX Operating System Commands Reference*.

In the report, the **user** and **sys** times reported in the second and third lines apply to the measured process itself, including all its children. The remaining data, including **cpu user %** and **cpu sys %**, apply to the entire system.

While the normal use of **timex** will probably be to measure a single command, multiple commands can also be timed, either by combining them in an executable file and timing it or by entering:

```
timex sh -c "cmd1; cmd2;...;"
```

This establishes the necessary parent-child relationships to correctly extract the user and system consumed by **cmd1**, **cmd2**,... (and the shell).

¦ Copyright IBM Corp. 1985, 1991

5.4.3 System Activity Daily Reports

The previous section described the commands available to users to initiate activity observations. It is probably desirable for each installation to routinely monitor and record system activity in a standard way for historical analysis. This section describes the steps that you may follow to automatically produce a standard daily report of system activity.

Subtopics
5.4.3.1 Facilities
5.4.3.2 Suggested Operational Setup

¦ Copyright IBM Corp. 1985, 1991

5.4.3.1 Facilities

sadc This command reads system counters from **/dev/kmem** and records them in a file. In addition to the file parameter, two frequency parameters are usually specified to indicate the sampling interval and number of samples to be taken. If frequency parameters are not given, it writes a dummy record in the file to indicate system restart.

sa1 The shell procedure that invokes **sadc** to write system counters in the daily data file **/usr/adm/sa/sadd**, where **dd** represents the day of the month. It may be invoked with sampling interval and iterations as parameters.

Use ASCII characters for file names to ensure that users of different locales can access these files.

sa2 The shell procedure that invokes **sar** to generate daily report **/usr/adm/sa/sardd** from the daily data file **/usr/adm/sa/sadd**. It also removes the daily data files and report files after seven days. The starting and ending times and all report options of **sar** are applicable to **sa2**.

¦ Copyright IBM Corp. 1985, 1991

5.4.3.2 Suggested Operational Setup

You should use **cron** to control the normal data collection and report generation operations. For example, the sample entries in **/usr/spool/cron/crontabs/adm** are:

```
0 * * * 0,6 /usr/lib/sa/sa1
0 18-7 * * 1-5 /usr/lib/sa/sa1
0 8-17 * * 1-5 /usr/lib/sa sa1 1200 3
```

would cause the data collection program **sadc** to be invoked every hour on the hour. Moreover, depending on the arguments presented, it writes data to the data file one to three times at every 20 minutes. Therefore, under the control of **cron**, the data file is written every 20 minutes between 8:00 and 18:00 on weekdays and hourly at other times.

Note that data samples are taken more frequently during prime time on week days to make them available for a finer and more detailed graphical display. You should invoke **sa1** hourly rather than invoking it once every day, as this ensures that if the system crashes, data collection will be resumed within an hour after the system is restarted.

Because system activity counters restart from zero when the system is restarted, a special record is written on the data file to reflect this situation. This process is accomplished by invoking **sadc** in **/etc/rc**, without frequency parameters, when going to multiuser state:

```
/usr/lib/sa/sadc /usr/adm/sa/sa` date +%d`
```

cron also controls the invocation of **sar** to generate the daily report via shell procedure **sa2**. You may choose the time period the daily report is to cover and the groups of system activity to be reported. For instance, if:

```
0 20 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:00 -i 3600 -uybd
```

is an entry in **/usr/spool/cron/crontabs/sys**, **cron** will execute the **sar** command to generate daily reports from the daily data file at 20:00 on week days. The daily report reports the processor utilization, terminal device activity, buffer usage, and device activity every hour from 8:00 to 18:00.

In case of a shortage of the disk space or for any other reason, these data files and report files can be removed by a user with superuser authority.

¦ Copyright IBM Corp. 1985, 1991

5.4.4 System Activity Data Structures and File Formats

The data structures that contain the extracted system activity statistics is described in the **include** file **<sys/sysinfo.h>** found in the file **/usr/include/sys/sysinfo.h**.

¦ Copyright IBM Corp. 1985, 1991

5.4.5 sar Data File Structure

The structure of the binary daily data file is:

```
struct sa {
struct sysinfo si;
struct pageinfo pi;
int szin/*ecurrent size of inode table */
int szfi/*;current size of file table */
int szte/*;current size of text table */
int szpr/*;current size of proc table */
int mszi/*dmaximum size of inode table */
int mszf/*emaximum size of file table */
int mszt/*tmaximum size of text table */
int mszp/*cmaximum size of proc table */
long ino/*ocum. overflows of inode table */
long fil/*fcum. overflows of file table */
long tex/*fcum. overflows of text table */
long pro/*fcum. overflows of proc table */
time_t t/* time stamp, seconds */
long dev/*[device[info for up to NDEVS units */
#def0ne I/*Onumber I/O requests */
#def1ne /*_number blocks transferred */
#def2ne /*_number drive busy time in ticks */
#def3ne /*_number I/O resp time in ticks */
};
```

¦ Copyright IBM Corp. 1985, 1991

5.5 Communicating with System Users

If your system has more than one user, you may find it convenient to use the system to communicate about such things as changes to the system, work schedules, unexpected system shutdowns, or new information. The services described in this section provide several convenient means for communication among users.

If non-ASCII characters are used in your messages, users of different locales may not be able to successfully access the files. See the ***Guide to Multibyte Character Set (MBCS) Support*** for more information.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

5.5.1 Communicating with Another User - The write Command

The **write** command sends a message to another user. Often, the **write** command is used to converse with another user (that is, each user alternately sends and receives a short message).

If you do not want to be interrupted by messages, enter **mesg n** to deny message permission. Enter **mesg y** to permit messages again.

The **write** command sends a sound signal (in ASCII) to the display station of the person receiving the message, and then displays the following message on that display station:

```
Message from username (ttynn)
[date] . . .
```

The format of how **[date]** is displayed depends on how it was previously set up.

Use ASCII characters to ensure that users of different locales can access these files.

After successful connection with the other user's display station, **write** sends two sound signals to your display station.

When you try to send a message to a user who is not logged in, **write** displays the message: **user not logged in**. If you send a message to a user who has refused message permission (with the **mesg** command), **write** displays the message: **write: permission denied**.

To Send a Message with write

1. Enter:

   write username

2. After the two sound signals, enter your message.

3. At the end of your message, press:

   END OF FILE (usually **Ctrl-D**).

After you send a **write** message, the other user can respond by sending a message back to you by using the **write** command also. You may find the following conventions for carrying out such an exchange of messages useful: When you first enter the **write username** command, wait for the other user to send a message back before you send any text. End each message with a signal such as **o** (over) to alert the other person to reply. Use **oo** (over and out) when you complete the exchange.

For more information about **write**, see **write** in *AIX Operating System Commands Reference*. For more information about communicating with different cluster sites in MBCS environments, see the *Guide to Multibyte Character Set (MBCS) Support*.

¦ Copyright IBM Corp. 1985, 1991

5.5.2 Sending a Message to all Logged-In Users - The wall Command

The **wall** (write all users) command sends a message to all logged-in users. If you realize that you will have to shut down the system unexpectedly, the **wall** command is a good way to tell everyone who logged in to bring their work to a stopping place and log out. You must have superuser authority to use the **wall** command.

To send a message with **wall**, enter **wall message**:

```
# wall System shutdown at 14:30
# _
```

The **wall** command sends the message, preceded by a heading, to every user logged in to the system, for example:

```
Broadcast Message from root
System shutdown at 14:30
```

Use ASCII characters in the **wall** command to ensure that users of different locales can access the files successfully.

¦ Copyright IBM Corp. 1985, 1991

5.5.3 Creating a Message of the Day

To communicate with all users who will log in on a given day (not just those currently logged in), you can create a ***message of the day***. If you have TCF installed in your cluster, you can have both a system-wide and a local-site-only message of the day.

A system-wide message is displayed (in ASCII) on all sites in the cluster with homogeneous locales. To create or change a system-wide message of the day, edit the **/etc/MOTD** file. After entering the text of your message into the **/etc/MOTD** file and saving the file, run the **makemotd** command. The **makemotd** command recreates the **/etc/motd** file, which is what the user actually sees on his terminal. The message of the day usually contains a banner giving the name of the site the user is logged in to as well as the text of the message.

To implement a local-site-only message of the day, edit **<LOCAL>/MOTD** (in ASCII) to create the text of your message. After you have saved this file, run the **makemotd** command. The **makemotd** command recreates the **/etc/motd** file, which now contains three items: the banner, the system-wide message, and the local-site-only message.

To create a temporary message of the day, edit the **/etc/motd** file, adding the text of your message. You do not run the **makemotd** command. This temporary message is displayed to each user who logs on until the next time the system is rebooted. Each reboot runs the **makemotd** command and, thus, recreates the **/etc/motd** file.

Use ASCII characters for all information, including directory and file names.

¦ Copyright IBM Corp. 1985, 1991

5.5.4 Creating and Reading News Items - The news Command

Use the **news** command to keep system users informed of news about the system or other topics of general interest. There are two tasks involved in using the **news** command:

Posting news item

Informing system users about how **news** works.

To post a news item, create a file in the **/usr/news** directory. This file and file name can contain non-ASCII characters. You can either use an editor to create a file in **/usr/news** or copy a file from another directory into **/usr/news**. For example, if you create a file named **schedule** in your current directory, you can use the **cp** (copy) command to enter **schedule** as a news item:

```
cp schedule /usr/news
```

To create a news item, you must have write permission for the **/usr/news** directory. You should periodically remove all the files that contain outdated news items.

The **news** command gives you access to the items stored in **/usr/news**. To avoid reporting old news, the **news** command stores a ***currency time*** each time a user reads news items. The **news** command considers only the items posted after this time to be current for that user.

You can use the **news** command alone, with the name of one or more news items, or with one of three flags, depending upon the type of information you need. The following list explains the type of information returned by each version of the **news** command:

news Displays all items posted since you last read the news. To display the news items one page at a time, pipe the output of **news** to the **pg** command:

```
news ¦ pg
```

news -a Displays all news items, regardless of the currency time. The currency time does not change.

news -n Reports the names of current news items without displaying their contents. The currency time does not change.

news -s Reports the number of current news items without displaying their names or contents. The currency time does not change.

news item Displays the contents of a particular **item**.

Typically, users enter the **news -n** command first. If **news -n** shows items that may be of interest, then the user can use the **news item** command to display the contents of the individual news items.

The most convenient way to use **news -n** is to set it up to run automatically each time a user logs in, which you can do in one of two ways:

Add **news -n** to each user's **$HOME/.profile** file.

¦ Copyright IBM Corp. 1985, 1991

Add **news -n** to the system's **/etc/profile** file.

¦ Copyright IBM Corp. 1985, 1991

5.5.5 Sending and Reading Messages - The mail Command

Use ASCII characters for all commands to ensure that users of different locales can communicate successfully. The content of mail messages may be non-ASCII.

The **mail** command allows you to:

Send messages or files to specific users

Receive and process mail sent to you

To Send Mail

1. Enter:

   mail username

   where **username** is a list of one or more user names.

2. Enter your message.

3. Press **END OF FILE** (usually **Ctrl-D**).

You can also use **mail** to send a file to other users with a command of the form:

mail **username** <**filename**

The **mail** command prefixes each message with the sender's name and the date and time of the message (its ***postmark***). **mail** then places messages in **$HOME/.newmail** (for example, **/u/tom/.newmail**). When users read their mail and then exits from **mail**, **mail** saves messages that have already been read in the user's mailbox--that user's **$HOME/mbox** file (unless the user specifies a different file name or specifies that the messages should stay in the **.newmail** file).

To Read Mail

1. Enter:

   mail

2. At the **?** prompt following each message, press **Enter** to read the next message.

3. (Optional) To display the previous message, enter:

   - (hyphen)

4. (Optional) To delete the current message and display the next message, enter:

   d

5. (Optional) To forward a message to another user, enter:

   m username

¦ Copyright IBM Corp. 1985, 1991

6. To end the **mail** program:

   Enter **q** or press **END OF FILE** (usually **Ctrl-D**).

Messages that you have already read are saved in **$HOME/mbox**.

For information about other **mail** flags and subcommands, see **mail** in *AIX Operating System Commands Reference*. For more information about communicating with different cluster sites in MBCS environments, see the *Guide to Multibyte Character Set (MBCS) Support*.

¦ Copyright IBM Corp. 1985, 1991

5.5.6 Identifying Logged-In Users - The who Command

Use the **who** command to identify all users currently logged-in to the system. Besides showing the user name of each logged in user, **who** also reports which display station each user is using and the date and time that each user logged in. In the following example, **who** reports that there are three users logged in to the system and the TCF site where each user is logged in:

```
$ who
sam       console       Apr   4  09:19   <site1>
pat       tty0          Apr   4  13:31   <site2>
mark      tty2          Apr   4  15:04   <site3>
$ _
```

To identify only the users on your local site, enter **who -L**.

For information about other ways to use the **who** command, see **who** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

5.6 Managing Ports, Cables, and Modems on a PS/2

This section provides an overview of the role of ports, cables, and modems in establishing communication between a local and a remote computer.

The section includes the following information:

- Configuring ports, including types of ports, using port commands, an setting up a port
- Connecting terminals and modems, including using adapters, cables, an null modem cables
- Configuring modems, including modem connections for both call-out an call-in modems, and modem switch settings.

Two sample modem connections show the relationship between configuring modems and configuring the Basic Networking Utilities (BNU) communications facility for use with modem connections. For information about BNU, see Chapter 8, "Managing the Basic Networking Utilities" in topic 6.8.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

5.6.1 Ports

This section provides an overview of ports and tells you how to perform the following operations:

- Use the port commands **pstart**, **penable**, **pdelay**, **pshare**, **pdisable**, and **phold** to change the port configurations at your site.

- Use the **devices** command to customize port configurations to work with the communications facility used at your site.

As used in this section, a ***port*** is the logical connection between your computer and another piece of equipment such as a computer, a printer, or a terminal. A port is associated with certain programs and files.

Physically, a port is supported by an adapter in the computer. The adapter connects the computer to another piece of equipment using one of the following:

- A direct cable or a null modem cable that permanently connects terminal to the adapter. This is called a ***hardwired connection***.

- A modem and telephone line that connect temporarily with remot systems for both local and long-distance calls.

- A short-haul modem, which is a special type of hardwired connectio that functions like a modem for short distances.

Subtopics
5.6.1.1 Types of Ports
5.6.1.2 Using the Port Commands
5.6.1.3 How to Set Up a Port (devices)

¦ Copyright IBM Corp. 1985, 1991

5.6.1.1 Types of Ports

A port is set up either for an outgoing or an incoming call. Some ports are always set to a ***call-in*** state, while others are always set to a ***call-out*** state:

Call-out State In this state, a local system initiates a connection to a remote system. In order to make an outgoing call, the port must be ***disabled*** using the **pdisable** command. Executing this command prevents another system from attempting to log in on the local system.

A call-out port is also referred to as a ***primary site***.

Note: This use of the term, primary site, is completely different from its use in the context of AIX as a network of interconnected machines.

Call-in State In this state, a local system receives a login from a remote system. The port must be ***enabled***, using the **penable** command, to receive information from a remote system.

If the only port on a system is a call-in port, that port is referred to as a ***secondary site***.

Still other ports are set to switch functions automatically in response to a port command. These are called ***bidirectional ports***. For example, when a user on a local system issues a BNU command to send a file to a remote system, the port used to make the connection is disabled. On the other hand, a port is enabled when the local system receives a file sent by BNU from another system.

¦ Copyright IBM Corp. 1985, 1991

5.6.1.2 Using the Port Commands

There are six commands that you use to enable and disable ports. You must enter these commands in ASCII characters.

pstart port_name Enables all ports (normal, shared, and delayed) that are enabled in the **/etc/ports** file. If you do not specify a device to enable, **pstart** reports the names of all enabled ports and tells whether they are currently enabled as normal, shared, or delayed. Usually the command is run in the form **pstart -a -i -w** from **/etc/rc** to enable all ports on a multi-user system. To enable a port, the stanza for the port in **/etc/ports** must have enabled=true and the stanza for the port in **/etc/inittab** must have action=respawn.

penable port_name Enables normal ports that are enabled in the **/etc/ports** file. Normal ports are ports that are asynchronous and only allow users to log in to those ports. No outgoing use of the port is allowed while it is enabled. This command is equivalent to the statement **penable enabled=true**. If you do not specify a device, **penable** reports the names of the currently enabled normal ports. To enable a port, the stanza for the port in **/etc/ports** must have enabled=true and the stanza for the port in **/etc/inittab** must have action=respawn.

pshare port_name Enables shared ports that are enabled in the **/etc/ports** file. Shared ports are bidirectional. This command is equivalent to the statement **pshare enabled=share**. If you do not specify a device, **pshare** reports the names of the currently enabled shared ports. To enable shared ports, **getty** attempts to create a lock file in **/etc/locks** that contains the ASCII process ID of the **getty** process. If the port is already in use by some other process, **getty** waits until the port is available and tries again.

pdelay port_name Enables delayed ports that are enabled in the **/etc/ports** file. Delayed ports are ports that are enabled like shared ports except that the login herald is not displayed until the user types one or more characters (usually carriage returns). If the port is directly connected to a remote system or connected to an intelligent modem, the port is usually enabled as a delayed port to prevent the **getty** from talking to a **getty** on the remote side, or to the modem on a local connection, thereby consuming system resources. This statement is equivalent to **pdelay enabled=delay**. If you do not specify a device, **pdelay** reports the name of the currently enabled delayed ports.

pdisable port_name
Disables the port. The local system cannot accept a remote login request.

phold port_name Specifies that an enabled port should be disabled as soon as the current user logs out.

These commands work by changing the status of the port that is specified

¦ Copyright IBM Corp. 1985, 1991

in the **/etc/inittab** file. When the port status is changed with one of the port commands, that change is only temporary. Once the system is shut down and then restarted, the port status reverts to the default value, which is specified with the **devices** command.

You can use the port commands to change a single port or a class of ports. Following are some examples of using the port commands.

The following command disables all ports configured for call-ins

```
pdisable term = dialin
```

This next example command enables all normal, shared, and delaye ports that are enabled in the **/etc/ports** file, reinitializes the existing **/etc/ports**, and makes the command return immediately rather than wait for **init** to confirm port status:

```
pstart -a i -w
```

This example enables the system attached to the **/dev/tty0** port as a shared port:

```
pshare /dev/tty0
```

The next example enables all 9600 baud ports as delayed

```
pdelay speed=9600
```

The following command disables logins from remote computers using specified device:

```
pdisable tty2
```

¦ Copyright IBM Corp. 1985, 1991

5.6.1.3 How to Set Up a Port (devices)

To set up a port, you must perform the following actions:

Specify the adapter port and the device to which you want to connec

Select the device settings that create a call-out, call-in, o bidirectional port.

To start both tasks, issue the **devices** command and then select the **add** subcommand from the Device Customizing Commands menu.

You must know the following information when setting up a port:

Device clas

Device typ

Port number, if requeste

Any changes required in the default device settings

Note: If you change the default settings at a later time, disable the device before issuing the **devices** command.

For information on devices and the **devices** command, see the AIX installation manual for your workstation. For examples of specifying the two types of modem connections, see "Example 1: BNU Modem Connection for Call-Out Port" in topic 5.6.3 and "Example 2: Configuring BNU for a Call-In Port" in topic 5.6.3.

¦ Copyright IBM Corp. 1985, 1991

5.6.2 Connecting Cables

This section briefly describes the adapters and cables that you need to connect terminals and modems to a system through asynchronous ports. For more detailed information, see the *AIX/370 Planning Guide*.

Subtopics
5.6.2.1 Using Adapters on the PS/2
5.6.2.2 Using Direct Cables
5.6.2.3 Using a Null Modem Cable

¦ Copyright IBM Corp. 1985, 1991

5.6.2.1 Using Adapters on the PS/2

You can use the serial adapter card or one of the serial ports on the system board for communications between a local and a remote system:

A dual-port RS-232 card (for a modem attachment or a hardwire connection)

The 8580 system board for a modem attachment or a hardwire connection.

¦ Copyright IBM Corp. 1985, 1991

5.6.2.2 Using Direct Cables

Consult the *PS/2 Quick Reference* for drawings of the various direct cable connections on PS/2s.

¦ Copyright IBM Corp. 1985, 1991

5.6.2.3 Using a Null Modem Cable

When two devices that both function as DTEs (data terminal equipment) are directly connected, the cable that connects them must transpose send/receive signals. A particular type of device called a ***null modem cable*** is typically required.

The null modem cable connects a terminal directly to a computer port through a wired connector with a specific pin arrangement.

¦ Copyright IBM Corp. 1985, 1991

5.6.3 Modems

There are two types of modems: ***internal*** and ***external***.

An internal modem is an adapter that fits into the computer. An external modem is a box outside the system unit that plugs into a single-port or dual-port adapter.

You must configure the modem connection for the specific modem that you use. The instructions that come with your modem provide a description of its characteristics.

The following examples show how to customize and configure Basic Networking Utilities (BNU) to use a call-out port and call-in port with a Hayes 1200 modem for the PS/2. (Other modems may require a different setting.) This connection enables the local system to call out to a remote system.

> For detailed information about customizing the BNU files fo connecting two computers over a telephone line, using modems, see "Setting Up Remote Communication" in topic 6.8.5.2.
>
> For information about the **devices** command, see "How to Set Up a Port (devices)" in topic 5.6.1.3.

Example 1: BNU Modem Connection for Call-Out Port:

Following are the characteristics of the modem for which you want to set up a BNU entry on the local system:

Type of Modem Hayes, 1200 baud

Name of the Calling System zeus (uucp node)

Name of the Device tty1

Destination Name odin (remote system)

Destination Phone Number 9-1-555-123-4567 (remote system)

Destination Login ID uucp (remote system)

Destination Password biguucp (remote system)

The following configuration enables the local system **zeus** (for example) to call out from the local modem to the modem connected to the remote system named **odin**.

+--- **Configuring a Call-out Port for a Local Modem** ----------+
¦ ¦
¦ 1. Add the following entry to the **/usr/adm/uucp/Devices** file: ¦
¦ ¦
¦ ACU tty1 - 1200 hayes \T ¦
¦ ¦
¦ This specifies the device that the local system **zeus** will use to ¦
¦ communicate with the remote system **odin**. ¦

¦ Copyright IBM Corp. 1985, 1991

2. Add the following entry to the **/usr/adm/uucp/Dialers** file:

   ```
   hayes =,-, "" \dAT\r\c OK \pATDT\T\r\c CONNECT
   ```

   This specifies the handshaking data used by the modem on **zeus** to connect to the modem on **odin**.

3. Add the following entry to the file **/usr/adm/uucp/Systems**:

   ```
   odin Any ACU 1200 local4567 in:--in: uucp word: biguucp
   ```

   This entry specifies that **zeus** can call **odin** at any time on any day.

4. Add the following entry to the **/usr/adm/uucp/Dialcodes** file:

   ```
   local 9-1-555-123
   ```

   This specifies the dialing code prefix used in the **Systems** file.

5. Add the following lines to the **/usr/adm/uucp/Permissions** file:

   ```
   LOGNAME=uucp VALIDATE=odin REQUEST=yes SENDFILES=yes READ=/ WRITE=/
   MACHINE=zeus:odin REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
   ```

   This specifies the access that **odin** has to **zeus**.

6. Add device **tty1** by entering the information in the next quick reference box.

For information about configuring the BNU files mentioned above, see the following:

For information about the **Devices** file, see "Setting Up Devices" in topic 6.8.5.2.1.

For information about the **Dialers** file, see "Setting Up Dialers" in topic 6.8.5.2.2.

For information about the **Systems** file, see "Customizing the Systems File" in topic 6.8.5.2.3.

For information about the **Dialcodes** file, see "Setting Up Dialing Codes" in topic 6.8.5.2.4.

For information about the **Permissions** file, see "Customizing the Permissions File" in topic 6.8.5.2.5.

¦ Copyright IBM Corp. 1985, 1991

Adding a Device for a Call-out Port

1. Enter the **devices** command:

   devices

2. When the Device Customizing Menu appears, enter the **add** option, the device class, and the device type:

   a ttydev tty

3. In response to the prompts, select serial port 1 and press the **Enter** key:

   s1

4. When the configuration menu appears, set the following and then press **Enter**:

   ae false (Automatic enable is false)

   Note: Do **not** allow users to log in through this port.

5. Indicate that you want to change another option.

6. Now set the device options as listed in the next quick reference box.

Setting Device Options for a Call-out Port

1. Set the options for the device specified in the above example as follows:

   ```
   bpc   8 (data sent 8 bits per second)
   pt    none  (no parity)
   rts   1200 (receive and transmit speed 1200 bps)
   dvam  1 (device attached by modem)
   nosb  1 (data has one stop bit)
   ixp   true (include Xon/Xoff protocol)
   ```

2. Use the default for all other options.

3. Press **Enter**.

¦ Copyright IBM Corp. 1985, 1991

Note: If you change a port configuration rather than create a new port, enter **penable** and then **pdisable** to put the new settings for the call-out port into effect.

Example 2: Configuring BNU for a Call-In Port:

This example shows how to customize and configure BNU to use a call-in port with a Hayes 1200 modem attached to a serial port on the PS/2. (Other devices may require a different setting.) The call-in port is on the local computer.

For detailed information about customizing the BNU files fo connecting two computers over a telephone line, using modems, see "Setting Up Remote Communication" in topic 6.8.5.2.

For information about the **devices** command, see "How to Set Up a Port (devices)" in topic 5.6.1.3.

Following are the characteristics of the modem for which you want to set up a BNU entry on the remote system, using the same modem in Example 1, above.

Type of Modem	Hayes, 1200 baud
Name of the Calling System	odin (remote system)
Name of the Device	tty1
Destination Name	zeus (local system)
Destination Phone Number	9-1-555-765-4321 (local system)
Destination Login ID	uucp (local system)
Destination Password	alluucp (local system)

Configuring a Call-in Port for a Remote Modem

1. Add the following entry to the file **/usr/adm/uucp/Systems**:

   ```
   zeus Any ACU 1200 local4321 in:--in: uucp word: alluucp
   ```

   This entry specifies that **odin** can call **zeus** at any time on any day.

2. Add the following lines to the **/usr/adm/uucp/Permissions** file:

   ```
   LOGNAME=uucp VALIDATE=zeus REQUEST=yes SENDFILES=yes READ=/ WRITE=/

   MACHINE=odin:zeus REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
   ```

3. Add device **tty1** by entering the information in the next quick reference box.

Copyright IBM Corp. 1985, 1991

For information about configuring the BNU files mentioned above, see the following:

For information about the **Devices** file, see "Setting Up Devices" in topic 6.8.5.2.1.

For information about the **Dialers** file, see "Setting Up Dialers" in topic 6.8.5.2.2.

For information about the **Systems** file, see "Customizing the Systems File" in topic 6.8.5.2.3.

For information about the **Dialcodes** file, see "Setting Up Dialing Codes" in topic 6.8.5.2.4.

For information about the **Permissions** file, see "Customizing the Permissions File" in topic 6.8.5.2.5.

For information about the **Myfile** file, see "Files in the Supporting Data Base" in topic 6.8.4.2.

For information about the **Spools** file, see "Files in the Supporting Data Base" in topic 6.8.4.2.

Adding a Device for a Call-in Port

1. Enter the **devices** command:

   ```
   devices
   ```

2. When the Device Customizing Menu appears enter the **add** option, the device class and the device type:

   ```
   a ttydev tty
   ```

3. A screen appears showing you which ports are available. In response to prompts type the following:

   ```
   s1
   ```

   This stands for Serial Port 1.

4. Now press **Enter**. A screen appears which allows you to change information which does not match your system configuration.

   Notes:

   a. The modification options you see on your screen may not match those shown. In many cases, the list you see will be smaller.

Copyright IBM Corp. 1985, 1991

After you press **Enter** you will see a screen which asks if you want to make any further changes. If you respond **yes** to this question you will be shown a screen that represents the whole list shown.

Which modification options appear on the first screen is controlled by:

/etc/ddi/tty.kaf for built in and dual async adapters

/etc/ddi/rtyt.kaf for the Artic adapter.

The options listed as *required = true* will show on the first screen. Those listed as *required = false* will only show on the second screen. You can reconfigure these selections to suit your needs.

b. In using these two screens, keep in mind that when you are finished the device settings must match the way the terminal is actually set (parity, bits per second, receive/transmit speed, and so on). See the **ddi** file format page in the *AIX Operating System Technical Reference, Volume 2* for further explanation of terminal parameters.

c. The **devices** command only allows one speed setting to be specified for each terminal line. To specify multiple values to be cycled through (e.g., to support a variable speed modem), you should enter one speed now and then later edit the **/etc/ports** file (see the **ports** file format page in the *AIX Operating System Technical Reference, Volume 2*).

5. Type **ae true**
   Note that automatic enable is now highlighted in the true state.

6. Press return (with **ae true** still highlighted).

7. A screen appears telling you that all device information for this device has been set for you.

Adding a Device for a Call-in Port (continued)

8. Type in yes, and then press **Enter**. Note that the following screen appears to let you set devices options.

Name	Description	Current Choice	Possible Choices
tt	Terminal Type	ibm3151	vt100, vt220, ibm3161
ae	Automatic Enable	true	true, false, share, delay
bpc	Bits Per Character	8	5, 6, 7, 8
pt	Parity Type	none	even, odd, mark, space, none
ixp	Include Xon/Xoff Protocol	true	true, false
xscan	Enable scan code terminal	true	true, false

¦ Copyright IBM Corp. 1985, 1991

dvam	**Device Attachment Method**	**0**	**0 = local, 1 = remote (modem)**
nosb	**Number of Stop Bits**	**1**	**1, 1.5, 2**
elevel	**Enable level for terminal**	**1, 4**	**one or more of 0-6, a, b, c**
rts	**Receive/Transmit Speed**	**9600**	**300, 1200, 4800, 9600, etc.**
cycle	**Cyclic Speed Setting Support**	**true**	
aa	**Automatic Answering**	**false**	**true, false**
pro	**Protocol**	**dtr**	**dtr, dc**

To CANCEL and return to the list of commands, press F3.
To CHANGE a current choice, type the Name followed by your new choice (example: lpi 6) and press Enter.
To ADD this device with the current choices, press Enter.

9. Continue with the instructions below for setting device options for a call-in port.

Setting Device Options for a Call-in Port

Note: For the following instructions, use the screen from step number 8, above, in "Adding a Device for a Call-in Port."

1. Set automatic answer (aa) for call-in by entering the following:

```
aa true
```

   Note that aa will be highlighted as true.

2. All other settings are the same as for the call-out port.

3. Press **Enter**.

4. Enter the following to complete the configuration for the call-in port:

```
penable tty1
```

Note: The steps described for configuring a call-in port are an example of the general procedure you follow to perform this task. You may need to set other parameters to match those of the remote system, such as transmission speed, parity, and bit length/character.

For more information, see the file format section for the **ports** file, in the *AIX Operating System Technical Reference (Volume 2)*.

The procedure for adding a device for a bidirectional port is exactly the same as that for adding a device for a call-in port, as described in the

¦ Copyright IBM Corp. 1985, 1991

quick reference box above.

The procedure for setting the device options for a bidirectional port is almost the same as for a call-in port, with one exception:

Instead of setting **ae true** (as in {AUTENB}), set the following:

```
ae delay
```

Subtopics
5.6.3.1 Switch Settings for a Hayes Smartmodem 1200

¦ Copyright IBM Corp. 1985, 1991

5.6.3.1 Switch Settings for a Hayes Smartmodem 1200

In the settings that follow, **Up** means "open", and **Down** means "closed" on the modem switch. The numbers represent the modem switches--that is, **1** represents switch number 1, **2** represents switch number 2, and so on.

Switch	Setting	Description
1:	**Up**	Supports RS232C DTR lead.
2:	**Up**	Word result codes displayed.
3:	**Down**	Result codes are sent.
4:	**Down**	No echo unless half-duplex is selected and modem is on line.
5:	**Up/Down**	Up for autoanswer on first ring. This setting should be **Up** for a ports configured for dialin or bi-directional dialin/dialout.
6:	**Down**	RS-232C Carrier Detect on (always).
7:	**Up/Down**	Single line/multiline. This setting should be **Up** for a single line telephone installation (RJ11 phone jack). This setting should be **Down** for a multiline telephone installation (RJ12 or RJ13 phone jack).
8:	**Down**	Enable modem command recognition.
9:	**Up**	Compatible with Bell 103/212A modem.
10:	**Up**	Modem reset when turned on.

Configure the associated terminal port with modem control.

Note: The Settings used in the example above comply to U.S. standards. If you have a modem other than this type, follow the specific instructions that come with that modem.

¦ Copyright IBM Corp. 1985, 1991

6.0 Part 2. Managing AIX Data Communication

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.6 Chapter 6. Managing the Electronic Mail System

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.6.1 Contents

¦ Copyright IBM Corp. 1985, 1991

6.6.2 About This Chapter

This chapter describes the general structure of the electronic mail system and how to set it up to route mail in local and networking configurations. To use the mail system to route mail in a network, you must have already installed the network and have it working for other network functions. Types of networks over which you can route mail include:

tcp/ip - Refer to *AIX TCP/IP User's Guide* for information on how to configure and use this network.

uucp - Refer to Chapter 8, "Managing the Basic Networking Utilities" in topic 6.8 for information to configure this network. Refer to *Using the AIX Operating System* for information to use this program on an asynchronous network.

This chapter also includes information to help you manage the mail system including information about:

Managing the mail queu

Keeping a record of mail system activitie

Changing the configuration file to meet unusual circumstances

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System*, and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

6.6.3 Understanding the Electronic Mail System

The mail system is a general purpose, inter-network mail routing facility. It is not tied to any one transport protocol. It relays messages from one user to another across system and network domain boundaries. While processing the messages, the mail system can do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain.

The mail system consists of the following main components as shown in Figure 6-1:

A user interface program for handling message
The **sendmail** program for routing messages
One or more mailer programs to transmit messages to thei destinations.

Figure 6-1. Mail System Overview

You must use ASCII characters for all user IDs, site names, host names, and domains.

The following paragraphs outline the functions of the main components of the mail system.

Subtopics
6.6.3.1 The User Interface
6.6.3.2 Mail Routing
6.6.3.3 Mailer Programs
6.6.3.4 International Character Support

¦ Copyright IBM Corp. 1985, 1991

6.6.3.1 The User Interface

The user interface to the mail system is the **mail** program. This program allows the user to:

Compose messages to send to other user

Receive and read messages sent by other user

Manage a set of mail box and folder files to keep track of message sent and received.

Refer to *Using the AIX Operating System* for information about using **mail**. This program is provided as the standard interface to the mail system. MH can also be used together with **mail**. Refer to Chapter 9, "Overview of the Message Handling Package" in topic 6.9 and *AIX Operating System Commands Reference* for information about MH.

¦ Copyright IBM Corp. 1985, 1991

6.6.3.2 Mail Routing

The **sendmail** program provides mail routing functions for the mail system. This program interfaces with several user application programs to receive and deliver messages to users. In addition, you can configure it to route mail across network connections to users at remote systems. The **sendmail** program can route mail to users in the following network situations:

Users on the local syste
Users connected to the local system with a TCP/IP protoco
Users connected to the local system with a **uucp** protocol.
Users on your local VM system or any VM node that communicates wit it.

Note: You must use ASCII characters for site and login IDs.

Refer to *Using the AIX Operating System* for a description of the addressing format used to deliver mail in any of these situations.

¦ Copyright IBM Corp. 1985, 1991

6.6.3.3 Mailer Programs

Mailer programs are those programs that either deliver the message to local users or transmit the message to a remote system. The programs that perform these functions include:

bellmail This program delivers mail to local users.

sendmail This program uses SMTP to transmit mail across a TCP/IP network.

uucp This program transmits mail across an asynchronous connection.

rscsmail This program writes a VM spool file to **/dev/pun**. The file is then forwarded to the destination user via the RSCS virtual machine.

¦ Copyright IBM Corp. 1985, 1991

6.6.3.4 International Character Support

The standard **mail** program provided by AIX is modified to handle Japanese characters both as message text and as names of files. The text of mail messages sent to users on the same cluster can contain Japanese characters, and incoming Japanese text from other users is also handled properly. Users can write incoming messages to files with Japanese names or include such files in outgoing messages. As with all AIX commands, the name of the **mail** program and its options and subcommands are entered in ASCII characters. Kanji text in a mail message is received correctly by another user if the recipient is running in a Japanese local on the same cluster.

The transfer of Kanji text in mail messages may not work reliably between AIX and other operating systems. Most **mail** programs limit messages to 7-bit ASCII, and the remote transfer protocols allow only a 7-bit data path. These programs assume the the high-order bit is zero; some even zero it out. Such programs are at best unable to interpret or display Kanji text properly and at worst may destroy it. Therefore, Japanese users can feel free to create mail in Japanese text for local users but should use only ASCII characters in mail to users on remote systems.

The following is an alternative to **mail** for sending messages to remote users:

1. Each user creates a **/mail** subdirectory in his or her **HOME** directory and secures it with standard permissions. All incoming messages are transferred into this directory.

2. The sender uses an editor to create a message with Japanese text and gives the message an ASCII name.

3. The sender uses either **uucp** or FTP to transfer the message into the recipient's **/mail** directory.

4. The sender uses the standard **mail** program to send the recipient a mail message that contains the file name of the message that was transferred.

5. The recipient retrieves the message from his or her **/mail** subdirectory.

¦ Copyright IBM Corp. 1985, 1991

6.6.4 Understanding Mail System Files

When you install **sendmail** and set up a mail system for your computer, you must install, create or change some files in the system. Figure 6-2 shows files associated with the mail system. The information following the diagram summarizes the purpose of the files that are affected by setting up a mail system.

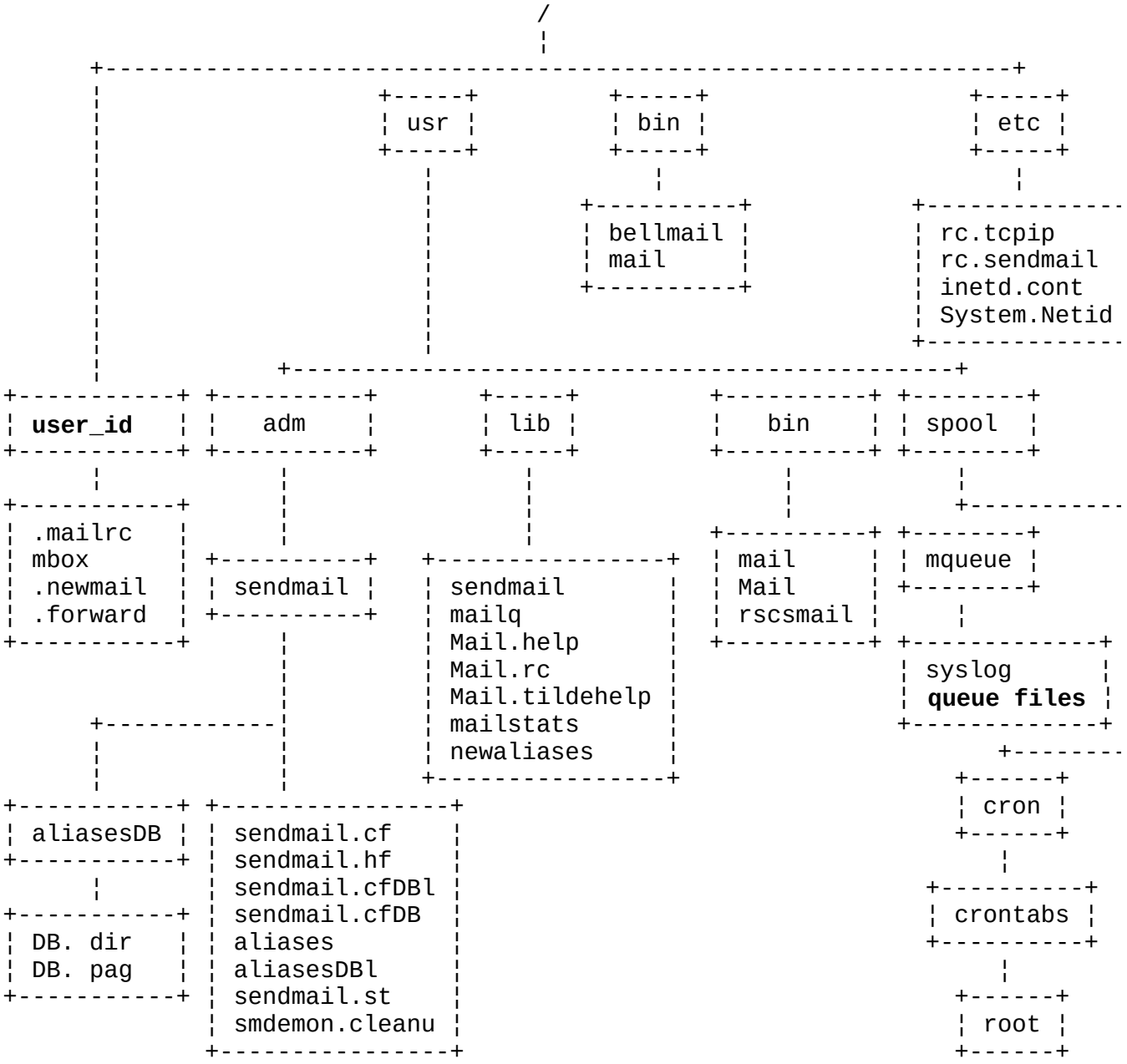


Figure 6-2. Mail System Files

Subtopics
6.6.4.1 Understanding Files for the mail Command
6.6.4.2 Understanding Files for the sendmail Program

¦ Copyright IBM Corp. 1985, 1991

6.6.4.1 Understanding Files for the mail Command

The **mail** command is the user interface to the mail system. Refer to *Using the AIX Operating System* for a description of how to use this program. The following files are associated with the **mail** program:

Table 6-1. Mail Files

Directory or File	Use
/usr/lib:	
Mail.help	This file is a text file containing help information for using the mailbox handling function of the **mail** program.
Mail.tildehelp	This file is a text file containing help information for using the **mail** editor to create messages.
Mail.rc	This file is a text file that you can modify to set the default characteristics of the **mail** program (see "Defining the Characteristics of the Mail Program" in topic 6.6.6).
/u/user_id:	
.mailrc	This file is a text file that each user creates in the **$HOME** directory. It allows the user to change the system defaults for the **mail** program when that user runs the program. See *Using the AIX Operating System* for information about this file.
.newmail	This file is a text file into which the system delivers new mail. Incoming mail is found here when the user runs the **mail** program.
mbox	This file is a text file that stores processed mail for the individual user. See *Using the AIX Operating System* for more information about this file.
.forward	This file is the user's local mail forwarding file. This file must have permissions set so

Copyright IBM Corp. 1985, 1991

	that it is owned, readable, and writable by the user. For more information about forwarding mail, see *Using the AIX Operating System*.
/usr/bin:	
Mail or **mail**	These two names are linked to the same program. The **mail** program is the user interface to the mail system. The **mail** program is the Berkeley 4.2 based **mail** program.
rscsmail	This program is the interface to communicating with or sending mail to VM users. It is called from the **sendmail** program. It should not be invoked directly by the user.
/bin:	
bellmail	This file contains the original mail program that comes with the operating system and is the original UNIX 5.2 based **mail** program. This program performs local mail delivery.
rmail	This program is the interface to **sendmail** that **uucp** uses to deliver mail. It is not the same **rmail** program that existed in earlier versions of the operating system.

Copyright IBM Corp. 1985, 1991

6.6.4.2 Understanding Files for the sendmail Program

Although the **sendmail** program can be used directly to send mail to other users, it is normally used as a background process to control mail routing. The **sendmail** program uses the following files. You can use different file names for many of these files by specifying a flag or configuration option to **sendmail**. See *AIX Operating System Commands Reference* for complete information about **sendmail** command flags.

Table 6-2. sendmail Files

Directory or File	Use
/usr/lib:	
sendmail	This file is the **sendmail** program.
mailq	This file is a link to **/usr/lib/sendmail** that executes **/usr/lib/sendmail -bp** to print a listing of the mail queue.
newaliases	This file is a link to **/usr/lib/sendmail** that executes **/usr/lib/sendmail -bi** to build an alias data base from the text file **/usr/adm/sendmail/aliases**.
mailstats	This command formats and prints the **sendmail** statistics as found in **/usr/adm/sendmail/sendmail.st** if that file exists (see "Logging Mailer Statistics" in topic 6.6.8).
edconfig	This program provides a menu interface for editing many parameters in the **sendmail** configuration file.
edconfig.hf	This file contains the help text that the **edconfig** program displays.
/usr/adm/sendmail:	
aliases	This file is a text version of the aliases file for **sendmail**. You can edit this file to create new aliases for your system.

Copyright IBM Corp. 1985, 1991

sendmail.hf	This file contains help information used by the SMTP **HELP** command.
sendmail.cf	This file contains the **sendmail** configuration information in text form. You can edit this file to change this information. See "Changing the Sendmail Configuration File" in topic 6.6.10.
sendmail.fc	This file contains the frozen version of the **sendmail.cf** file. This file is created from **sendmail.cf** when you run **/usr/lib/sendmail -bz**.
smdemon.cleanu	This file is a shell program file that runs the mail queue and maintains the **sendmail** log files in the directory **/usr/spool/mqueue**. See "Managing the Log and Mail Queue" in topic 6.6.7.3 for more information about this program.
sendmail.st	This file collects statistics about mail traffic. The file does not grow. Use the **/usr/lib/mailstats** command to display the contents of the file. You do not need to have this file if you do not want to collect this information. See "Logging Mailer Statistics" in topic 6.6.8 for more information.
/usr/spool:	
mqueue	This directory contains the **syslog** file and temporary files associated with each message in the queue. Refer to "Managing the Mail Queue" in topic 6.6.9 for information about the temporary queue files. Refer to "Logging Mail System Activities" in topic 6.6.7 for information about the mail log file.
cron/crontabs	This directory contains files that **cron** reads to determine jobs that it is to start. The **root** file contains a line to start **smdemon.cleanu** (see "Managing the Log and Mail Queue" in topic 6.6.7.3).
/etc:	

Copyright IBM Corp. 1985, 1991

rc.tcpip	This file contains shell scripts to start up many functions associated with TCP/IP. If you install TCP/IP, you must uncomment certain lines in this file to start **sendmail** running as a daemon on each site. See "Starting sendmail with TCP/IP Installed" in topic 6.6.5.4.1 for information about changing this file. You also have the option to use the **inetd** daemon to start **sendmail** (see the following).
inetd.conf **rc.sendmail**	This file contains an entry which when enabled (uncommented) causes the **inetd** daemon to listen for incoming mail calls. If a **sendmail** daemon is not required on each site in the cluster, you must uncomment certain lines in this file to start **sendmail**. When data comes in, the **inetd** daemon executes **sendmail** with a specific flag to indicate that incoming data has already been received. See "Starting sendmail Through the inetd Daemon" in topic 6.6.5.4.2 for information about changing this file.
System.Netid	This file contains a shell script to start the **sendmail** daemon. On 370 systems, **/etc/System.Netid** is a symbolic link to **<LOCAL>/System.Netid**. There is no corresponding file on PS/2 or PS/55 systems. This file describes the connection between AIX/370 and the RSCS virtual machine.

Refer to *AIX Operating System Technical Reference* for additional information about the configuration file **sendmail.cf**.

¦ Copyright IBM Corp. 1985, 1991

6.6.5 Setting Up Mail Delivery

You can set up the mail system to deliver mail to users on a local computer, to users connected by a local area network, to users connected by a **uucp link**, or a combination of these connections. Setting up the mail system for any of these environments requires similar procedures for each type of configuration. Only the information provided to **sendmail**, and the type of addressing required for each network are different.

+--- **Setting Up Mail Delivery** ----------+

Note: You must be a member of the system group to configure the mail system.

1. Install the **sendmail** program from the Extended Services Program (see *Installing and Customizing the AIX Operating System* for your workstation.

2. Ensure that the configuration information in **/usr/adm/sendmail/sendmail.cf** accurately reflects your system's needs. See "Defining Mail Requirements" in topic 6.6.5.1 for information about this file. Then, build a data base version of the configuration file:

```
/usr/lib/sendmail -bz
```

3. Define the addressing and routing information required for your mail delivery system. See "Defining Addressing and Routing Information" in topic 6.6.5.2 for information about addressing requirements.

4. Ensure that the alias information in **/usr/adm/sendmail/aliases** accurately reflects your system's needs. See "Defining Aliases" in topic 6.6.5.3 for information about this file. Then, create data base files from this file:

```
/usr/lib/sendmail -bi
        or
/usr/lib/newaliases
```

5. Select which daemon will run on each site to start the **sendmail** program. Your choices are:

   Having **sendmail** run constantly as a daemon process.
   Having **inetd** run as a daemon that starts the **sendmail** program in response to incoming calls.

   The default is the **inetd** daemon. If you want **inetd** to run, you need to take no further action.

6. If you want to run **sendmail** as your daemon, you must:

   Uncomment the lines in the **rc.tcpip** file that invoke **/etc/rc.sendmail**. (As delivered, these lines are commented out.)

¦ Copyright IBM Corp. 1985, 1991

Modify the **inetd.conf** file so that it does not try to start **sendmail** since **sendmail** will already be running.

Subtopics
6.6.5.1 Defining Mail Requirements
6.6.5.2 Defining Addressing and Routing Information
6.6.5.3 Defining Aliases
6.6.5.4 Starting the sendmail Daemon

¦ Copyright IBM Corp. 1985, 1991

6.6.5.1 Defining Mail Requirements

The **sendmail** program reads the file **/usr/adm/sendmail/sendmail.cf** for information about the network addressing, the type of mailer programs being used and information about how it should format messages. This file is supplied with the **sendmail** programs and contains a set of information that allows **sendmail** to operate in the following environments with little or no changes required to the file:

Local mail deliver

Local area network delivery using TCP/IP addressing formats

- **user_id**
- **user_id@host**
- **user_id@host.domain**

  You must use ASCII characters for all user IDs, site names, and host names for successful communication.

Remote delivery using **uucp**

Delivery to VM users using **rscsmail**.

If your environment includes one or all of these types of mail delivery, you may be able to use the supplied configuration file with only a few changes:

Host name macro (see "Changing the Host Name Macro" in topic 6.6.10.1.1)

Host name class (see "Changing the Host Name Class" in topic 6.6.10.1.2)

Domain name macro (see "Changing the Domain Name Macro" in topic 6.6.10.1.3)

Domain name classes (see "Changing the Domain Name Part Macros" in topic 6.6.10.1.4).

Only unique situations should require any other changes to this file. For information about changing this file, refer to "Changing the Sendmail Configuration File" in topic 6.6.10.

Subtopics
6.6.5.1.1 Building the Configuration File

¦ Copyright IBM Corp. 1985, 1991

6.6.5.1.1 Building the Configuration File

The first time you install **sendmail** and each time you change the configuration file, you must build the data base version of the configuration file:

```
/usr/lib/sendmail -bz
```

This operation creates the file **/usr/adm/sendmail/sendmail.cfDB** which contains the data base version of the configuration information. Because the **sendmail** daemon reads the data base version of its configuration file each time it is invoked, it is not sufficient simply to edit the text version of the configuration file as the text version is not automatically built into a data base each time it is edited. Therefore, if the **sendmail** daemon is running, you must kill that process, build the configuration file with the above command, and then start the daemon again for the new configuration file to take effect.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.2 Defining Addressing and Routing Information

The type of routing and addressing information needed depends upon how complex your mail delivery system is. Mail can be delivered:

Locall

To users on a local area networ

To users on a **uucp** network

To users on the VM system

To deliver mail in any of these modes, you must define the addressing and routing information needed for the selected mode. To deliver mail in more than one mode, you must define the addressing and routing information for each mode that you want to use.

Subtopics
6.6.5.2.1 Defining Local Information
6.6.5.2.2 Defining Local Area Network Information
6.6.5.2.3 Defining uucp Information
6.6.5.2.4 Defining RSCS Information

¦ Copyright IBM Corp. 1985, 1991

6.6.5.2.1 Defining Local Information

To deliver mail in a local environment (where all senders and receivers are users on the same system), no additional addressing and routing information is required. The **sendmail** program uses the information already available in the local system files to determine the correct routing for a local message.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.2.2 Defining Local Area Network Information

To deliver mail in a local area network (where senders and receivers may be on different systems connected by local area network links), the network must be using TCP/IP as one available network protocol. For networks with many systems, you may want to set up one or more systems on the network as the name server for the network. The name server controls the addressing information for the network and for connections to systems beyond the immediate network. Using a name server simplifies the task of keeping the address information up to date. See *AIX TCP/IP User's Guide* for information about using this program and for instructions to set up a system as a name server on a local area network.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.2.3 Defining uucp Information

To deliver mail to another system connected to the local system with a **uucp** link, you must define the addressing and routing information required for that link (see Chapter 8, "Managing the Basic Networking Utilities" in topic 6.8). The **sendmail** program transfers mail to the **uucp** daemons (through the **uux** command) for delivery across a **uucp** link. The **uusched** program handles the queued routing of mail for **uucp** links.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.2.4 Defining RSCS Information

To deliver mail via the VM RSCS virtual machine, use the **sendmail** program with a special address. The command has the following form:

sendmail **userid@nodeid**.RSCS

where:

userid identifies the user on the VM system.
nodeid identifies the computer on which the user is connected.
.RSCS indicates that the user must be reached through the VM system.

Note: You must use ASCII characters for all user id's and node id's to ensure successful communication.

The **sendmail** initiates **rscsmail** to deliver the message. For more information about **rscsmail**, see *AIX Operating System Commands Reference*.

By using the **sendmail** program in this way, you can deliver mail to:

CMS users on the local VM syste

AIX users on another AIX/370 system that is running on your local V system but not part of your cluster

Users on any VM system that is not part of your local VM system but i accessible via RSCS.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.3 Defining Aliases

A mail system ***alias*** is a name that represents an address or set of addresses. Purposes of defining aliases include:

Convenience: Using a short name, such as **mark** to represent a long address such as **mark@zeus.aus.IBM.COM**.

Accuracy: Using a short name for a complex address to help reduce the chance of a keying error in the address.

Distribution Lists: Using a short name for a list of addresses to allow you to send the same message to many people using only one address when you compose the message.

Rerouting Mail: Causing mail for one user ID to be delivered to another user ID with an alias.

Sending Mail to a File: Causing mail for a user ID to be delivered to a file instead of the system mailbox by using an alias.

You can define two types of aliases: personal aliases and local-system aliases. Personal aliases are aliases that operate on mail produced by the individual user that defined the alias. These aliases are defined in the user's **.mailrc** file as described in *Using the AIX Operating System*. Local-system aliases are aliases that apply to all mail handled by **sendmail** on the local system. Define these aliases in **/usr/adm/sendmail/aliases**, which is an editable text file that you can change if you are a member of system group.

The **aliases** file is included with the files that you install with the **sendmail** program. This file has all required aliases defined so that **sendmail** will work without changing this file in many cases. You may have to change this file if the predefined aliases do not match your system configuration, or to add new local-system aliases. After you change the **aliases** file, you must rebuild the alias data base that **sendmail** actually uses. If you do not rebuild the data base, **sendmail** does not recognize your changes.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.6.5.3.1 Building the Alias Data Base

The **sendmail** program does not use the alias definitions in the **aliases** file directly. Instead, you must process this file to produce data base files that allow **sendmail** to process them more quickly. When you are satisfied that the **aliases** file contains all the information that you need for your system, use the following command to create the alias data base:

```
/usr/lib/newaliases
        or
/usr/lib/sendmail -bi
```

This command causes **sendmail** to read the **aliases** file and create two additional files containing the alias data base information. These new alias files are:

/usr/adm/sendmail/aliases.dir

/usr/adm/sendmail/aliases.pag

If these files do not exist, **sendmail** cannot start and **sendmail** generates an error message when it tries to start.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.3.2 Creating a New Alias

To create a new local-system alias, edit the file **/usr/adm/sendmail/aliases**. You must be a member of system group to edit this file. Aliases in this file are defined with an entry in the following format:

a_name: **name1**, **name2**, ...**namex**

In this format **a_name** can be any alphanumeric string that you choose (not including any special characters, such as @ or !), and **name1** through **namex** is a series of one or more addresses. To continue the list of names on one or more lines, begin each continued line with a space or a tab. Blank lines and lines beginning with a number sign (#) are comment lines that are not included in the alias data base.

For example, the following entry defines an alias **writers** to be a set of addresses of people in that group:

```
writers:  geo, mark@zeus, ctw@athena, brian
```

This definition could be contained on several lines also, as long as each added line begins with a space or a tab:

```
writers:  geo,
      mark@zeus,
      ctw@athena,
      brian
```

¦ Copyright IBM Corp. 1985, 1991

6.6.5.3.3 Required Aliases

The **aliases** file must contain alias definitions for the following names:

MAILER-DAEMON alias
Assign this name to the ID of the user that is to receive mailer daemon messages. This name is initially assigned to **root**:

```
MAILER-DAEMON:  root
```

postmaster
Assign this name to the ID of the user responsible for the operation of the local mail system. The alias **Postmaster** (in any combination of uppercase or lowercase letters) defines a single mailbox address that is valid at each system in a network. This address allows users to send inquiries to the postmaster at any system without knowing the correct address of any user at that system. This name is initially assigned to **root**:

```
postmaster:  root
```

nobody Assign this name to the ID that is to receive messages directed to programs such as **news** and **msgs**. This name is initially assigned to **/dev/null**:

```
nobody:  /dev/null
```

¦ Copyright IBM Corp. 1985, 1991

6.6.5.3.4 Special Alias for List Owners

When creating a distribution list alias, you should also assign one ID as the **owner** of that list. If **sendmail** has a problem sending mail using a certain list, it sends an error message to the owner of that list. The format for assigning an owner to a list is:

```
owner-listname:  owner_addr
```

In this format, **listname** is the alias name of the distribution list and **owner_addr** is the address to which to send error messages for that distribution list. For example, the following set of entries in **aliases** defines a distribution list **editors** and its owner **glenda@hera**:

```
editors:  glenda@hera, davidm@kronos, perryw@athena
owner-editors:  glenda@hera
```

If there is no ID for **perryw** on system **athena**, then **sendmail** sends an error message to the mailbox for **glenda@hera** when someone sends mail to the **editors** distribution list.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.4 Starting the sendmail Daemon

The **sendmail** daemon listens on the SMTP socket for connections (to receive mail from a remote system) and processes the queue periodically to ensure that mail gets delivered when remote systems become active. Using flags to the **sendmail** command, you can activate either or both of these functions when you start the daemon.

If you are running TCF in your cluster, you have the choice of starting the **sendmail** daemon either directly from the **/etc/rc.sendmail** file or indirectly by having it started by the **inetd** daemon. If you start **sendmail** from the **/etc/rc.sendmail** file, the **sendmail** daemon runs on every site listening on the SMTP socket for connections. If you choose to start **sendmail** through the **inetd** daemon, when a call comes in, **inetd** invokes **sendmail** with specific flags that indicate that a communication has been established on a particular socket.

If you choose the first option, start **sendmail** from the **/etc/rc.sendmail** file as described in the following paragraph. If you choose the second option, start **sendmail** by making the appropriate change to the **/etc/inetd.conf** file (see "Starting sendmail Through the inetd Daemon" in topic 6.6.5.4.2).

The file **/etc/rc.sendmail** contains the program statements that start the **sendmail** daemon. Depending upon your configuration, you can run this file in one of the following ways:

Manually from the command line, using the **sh** command:

```
sh /etc/rc.sendmail
```

Automatically from the file **/etc/rc.tcpip** (if TCP/IP is installed and running).

Use the manual method only as part of the first installation procedure. Afterwards, start **sendmail** automatically.

Subtopics
6.6.5.4.1 Starting sendmail with TCP/IP Installed
6.6.5.4.2 Starting sendmail Through the inetd Daemon
6.6.5.4.3 Mail Delivery without TCP/IP Installed
6.6.5.4.4 Starting sendmail for Debugging Purposes

¦ Copyright IBM Corp. 1985, 1991

6.6.5.4.1 Starting sendmail with TCP/IP Installed

If TCP/IP is installed, the system runs the file **/etc/rc.tcpip** when it starts. This file contains a section of shell commands that runs the **sendmail** start-up file (**/etc/rc.sendmail**). Initially, this section is prevented from running because it has comment characters (#) at the beginning of the lines. The section looks like the following:

```
# Start up Sendmail Daemon
# if [ -f /usr/lib/sendmail ]; then
#    sh /etc/rc.sendmail 1>>/dev/null 2>.* &
#    echo " sendmail\c" >/dev/console
# fi
```

Use ASCII characters for the *rc.tcpip* file to ensure that users of different locales can access the files successfully.

To start the daemon when the system starts, edit the file and remove the last four comment characters from this section:

```
# Start up Sendmail Daemon
  if [ -f /usr/lib/sendmail ]; then
     sh /etc/rc.sendmail 1>>/dev/null 2>.* &
     echo " sendmail\c" >/dev/console
  fi
```

Save the file. The next time the system starts, **sendmail** will start as a background process.

Note: If you choose to use the **inetd** daemon to run **sendmail**, leave the comment symbols in place and edit the **/etc/inetd.conf** file instead (see "Starting sendmail Through the inetd Daemon" in topic 6.6.5.4.2).

¦ Copyright IBM Corp. 1985, 1991

6.6.5.4.2 Starting sendmail Through the inetd Daemon

If you choose to have the **inetd** daemon active on each site, the **inetd** daemon listens for incoming calls on the socket specified for SMTP connection. When a call is received, the **inetd** daemon starts **sendmail**. This saves CPU cycle time since the **sendmail** program only runs when needed.

This arrangement is the default value. As delivered to you, the **inetd** daemon is set up so that **sendmail** is one of the programs listed in its configuration file, **inetd.conf**. This line looks like the following:

```
smtp stream tcp nowait root /usr/lib/sendmail sendmail -bn
```

If this line has a comment symbol (#) at the beginning, it is disabled. Removing the comment symbol (#), enables it. The **-bn** flag used in the command sends a signal to the **sendmail** program informing **sendmail** that it was started by the **inetd** daemon.

Note: This line has a reciprocal relationship with a line in the **/etc/rc.sendmail** file that starts **sendmail** as a daemon. One of these lines must be commented out. The **sendmail** program can be started either by **inetd** or by the **rc.sendmail** file but not both. Therefore, if you choose to comment out the line in the **inetd.conf** file that allows **inetd** to run **sendmail**, you must remove the comment symbol from the appropriate line in the **rc.sendmail** file.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.4.3 Mail Delivery without TCP/IP Installed

Mail is delivered automatically to users on the local system without running the **sendmail** daemon. However, since mail can still enter the **sendmail** queue, you should process the queue periodically to ensure that mail is delivered. See "Managing the Log and Mail Queue" in topic 6.6.7.3 for information about using **cron** to force delivery of all mail in the queue. You can also process the mail queue by entering the following command at the # prompt on the site where the mail is spooled:

```
/usr/lib/sendmail -q
```

This command is run only on the site where mail is spooled since TCF allows you to define one site as the ***primary spool site***. Mail that does not get delivered is spooled here and retried at specific intervals. If this primary site is not available, the mail is spooled in the local mail queue on the site from which it was sent.

The advantage of defining a primary spool site is that only the primary spool site needs to keep a daemon constantly running to process the mail queue. The following command:

```
/usr/lib/sendmail -q30m.
```

processes the mail queue every 30 minutes.

All other sites will have mail spooled only infrequently--in the case of a failed delivery attempt. These sites can handle their infrequently used spools by occasionally running the cleanup program **smdemon.cleanup** from **cron**.

If your system is connected to others over **uucp** links (see Chapter 8, "Managing the Basic Networking Utilities" in topic 6.8), the mail system uses **uux -r** to place the mail for those systems in the **uucp** spooling directory. After that, **uucp** controls delivery of the mail to the remote systems.

¦ Copyright IBM Corp. 1985, 1991

6.6.5.4.4 Starting sendmail for Debugging Purposes

For special cases, such as testing and debugging a new installation, you may want to start **sendmail** directly from the command line using the command line options described in *AIX Operating System Commands Reference*.

The **sendmail** program includes debug flags. Each flag has a number and a level. Higher levels print out more information. The only debug flag that is useful without **sendmail** source code is flag 21 (for debugging rewriting rules). You must be a member of system group or know the **sendmail** debug password (set with the **-bw** flag to **sendmail** - see *AIX Operating System Commands Reference*). Set flags with the **-d** flag to **sendmail**. The syntax is:

```
/usr/lib/sendmail -d debug-flag[.level]
```

or:

```
/usr/lib/sendmail -d passwd -d debug-flag[.level]
```

In this format **debug-flag** is an integer that specifies the built-in debug flag to be set, and **level** is an optional integer that specifies the debug level defined for that flag. Imbedded blanks are not allowed. If you do not specify **level**, **sendmail** uses level 1. You can specify more than one **-d** flag. If the debug password is required, provide it as the argument to the first **-d** flag. You can specify more than one **debug-flag.level** pair by separating them with a comma:

```
debug-flag.level,debug-flag.level ...
```

where spaces are for reading ease only. You can also specify a range of flags to set using a dash (-) to separate the beginning and the ending flag numbers for the range. For example:

```
-d12                Set flag 12 to level 1
-d12.3              Set flag 12 to level 3
-d3-17              Set flags 3 through 17 to level 1
-d3-17.4            Set flags 3 through 17 to level 4
```

¦ Copyright IBM Corp. 1985, 1991

6.6.6 Defining the Characteristics of the Mail Program

You can define the characteristics of the **mail** program to provide a basic behavior of the program from which the rest of the system users can build. The **mail** program reads the file, **/usr/lib/Mail.rc** each time someone uses the program. This file can contain any of the mail commands for defining:

Aliase

The prompts that the program provide

Normal disposition of mail file

Other operational characteristics of the **mail** program.

These commands are the same as the commands that each user can put in **.mailrc** as described in the chapter of *Using the AIX Operating System* that discusses the mail system. Refer also to *AIX Operating System Commands Reference* for other commands that can be used in this file.

Remember that commands in **Mail.rc** override the default characteristics of the **mail** program for all users on the system. Users can create a **.mailrc** file in their home directories to contain commands to customize the **mail** program for their own use. These commands override similar commands in **Mail.rc**. In addition, users can enter the commands while using the **mail** program to override similar commands in both **.mailrc** and **Mail.rc**.

¦ Copyright IBM Corp. 1985, 1991

6.6.7 Logging Mail System Activities

The **sendmail** program automatically uses the **syslog** daemon to write messages into a log file when activities occur that affect the mail system. The log file is named **syslog** in the mail queue directory (**/usr/spool/mqueue** unless you change it. You must be a member of system group to access this file.

Subtopics
6.6.7.1 Understanding the Log Format
6.6.7.2 Choosing a Log Level
6.6.7.3 Managing the Log and Mail Queue

¦ Copyright IBM Corp. 1985, 1991

6.6.7.1 Understanding the Log Format

Messages in the log file appear in the following format:

<n> **Mmm dd hh:mm:ss** [**pid**] (**uname**) **msg_text**

The symbols in this format have the following meanings:

Table 6-3. Mail Log Fields

Symbol	Meaning
n	This field contains an integer between 0 and 7 that specifies the severity or importance of the message. The most important is 0; the least important is 7. Valid numbers are: **0** The system cannot be used. **1** Take action quickly to correct the problem. **2** The message represents a critical condition. **3** The message represents an error condition in the mail system. **4** The message represents a warning of a condition that could cause a problem. **5** The message informs you of a normal, but significant, condition. **6** The message represents a normal activity. **7** The message contains information for debugging the **sendmail** program.
Mmm dd hh:mm:ss	These fields specify the time that the message was entered in the log file: *Mmm* A three-letter abbreviation of the month *dd* The date of the month (01-31) *hh* The hour of the day (00-23) *mm* The minute of the hour (00-59) *ss* The second of the minute (00-59)
pid	This field contains the process ID of the process that generated the message.
uname	This field contains the user name associated with the message and must be in ASCII characters.
msg_text	This field contains a brief explanation of the condition that caused the message to be written to the log file.

For example, the following entry occurred in a log file when the system was started up, indicating that the **sendmail** program was started:

<6> Jan 7 07:43:49 [116] (root) Daemon/queue proc started, pid 116

For more information about the **syslog** file, see *AIX Operating System Technical Reference*.

¦ Copyright IBM Corp. 1985, 1991

6.6.7.2 Choosing a Log Level

You can choose the types of activities that **sendmail** puts into the log file by changing the **L** option line in **sendmail.cf** (see "Changing the Operational Logging Level" in topic 6.6.10.1.5 for information about changing this option). If you change the configuration file, you must process it with the **sendmail -bz** command before **sendmail** recognizes the change. In the standard configuration file the level is set with the following statement:

```
OL9
```

This statement turns on the logging function to level 9. Valid levels and the activities that they represent are (each number includes the activities of all numbers of lesser value and adds the activity that it represents):

Level	Activity Logged
0	Major activities only (building a configuration file or creating an alias data base).
1	Major problems only.
2	Message collections and failed deliveries.
3	Successful deliveries.
4	Messages being deferred (due to a host being down, etc.).
5	Placing messages in the queue (normal event).
6	Unusual but benign incidents (trying to process a locked file, etc.).
9	Log internal queue id to external message id mappings. This can be useful for tracing a message as it travels between several hosts.
12	Several messages that are of interest when debugging.
16	Verbose information regarding the queue.

If you change this file, you must process it with the **sendmail -bz** command before **sendmail** recognizes the change.

¦ Copyright IBM Corp. 1985, 1991

6.6.7.3 Managing the Log and Mail Queue

Because information is continually appended to the end of the log file, it can grow to become very large. Also, error conditions can cause unexpected entries to the mail queue. To help prevent problems associated with the mail queue and log, use the program **/usr/adm/sendmail/smdemon.cleanu**. This program forces **sendmail** to process the queue, and also maintains four progressively older copies of log files, named **syslog.0**, **syslog.1**, **syslog.2**, and **syslog.3**. Each time the program runs it moves:

1. **syslog.2** to **syslog.3**

2. **syslog.1** to **syslog.2**

3. **syslog.0** to **syslog.1**

4. **syslog** to **syslog.0**

This action removes the **syslog** file, allowing logging to start over with a new file. You can change this program to create more or fewer aging copies of the log file. Use the following procedure to have **cron** start this program at a specified time:

1. Log in as root, and change to the **crontabs** directory:

   ```
   cd /usr/spool/cron/crontabs
   ```

2. Edit the file **root** with your favorite editor.

3. The file contains a line similar to the following line:

   ```
   #45 23 * * * ulimit 5000; /usr/adm/sendmail/smdemon.cleanu > /dev/null
   ```

   Remove the # at the beginning of that line. This entry runs the program every day at 11:45 PM. If your system is not operating at that time, change the second number (23 representing the 23rd hour, or 11:00 PM) to a time when your system is on. For example, to run the cleanup program at 4:45 PM:

   ```
   45 16 * * * ulimit 5000; /usr/adm/sendmail/smdemon.cleanu > /dev/null
   ```

4. Save the file and exit from your editor.

5. Notify **cron** of the change to the file by reloading the file while still in the **crontabs** directory:

   ```
   crontab root
   ```

¦ Copyright IBM Corp. 1985, 1991

6.6.8 Logging Mailer Statistics

The **sendmail** program can keep track of the volume of mail being handled by each of the mailer programs that interface with it (those mailers defined in **sendmail.cf** - see "Defining a Mailer" in topic 6.6.10.7). To start the accumulation of mailer statistics, create the file **/usr/adm/sendmail/sendmail.st** (you must be a member of system group to create this file):

```
touch /usr/adm/sendmail/sendmail.st
chmod 660 /usr/adm/sendmail/sendmail.st
```

When this file exists and is either empty or contains a correctly formatted data structure, **sendmail** keeps track of mail volume handled by each mailer in a data base in that file. If the file is not in the correct format, **sendmail** does not change the file and does not log mailer statistics. This may happen if a statistics file created by an older version of **sendmail** exists on the system. In that case, save the older file for processing by the appropriate older version of **mailstats**. Then you can either remove the present file and create a new statistics file as shown above, or clear the statistics file using **mailstats -z** to begin recording statistics.

If **sendmail** encounters errors when it tries to record statistics information, it writes a message on the system console and in the **sendmail** log file. These errors do not affect other operations of **sendmail**.

The **sendmail** program updates the information in the file each time that it processes mail. The size of the file does not grow, but the numbers in the data base do. They represent the mail volume since the time you created or reset **sendmail.st**. To reset the numbers and start counting over again, use the **-z** flag to the **mailstats** command:

```
/usr/lib/mailstats -z
```

Subtopics
6.6.8.1 Displaying the Mailer Information
6.6.8.2 Statistics Messages

¦ Copyright IBM Corp. 1985, 1991

6.6.8.1 Displaying the Mailer Information

The information kept in **sendmail.st** is in a data base format that cannot be read as a simple text file. To display the mailer statistics, use the **mailstats** command:

```
/usr/lib/mailstats
```

This command reads the information in **sendmail.st**, formats it, and writes it to standard output. The format of the information is shown in the following example:

```
Sendmail statistics from file "/usr/adm/sendmail/sendmail.st"
          Collection started at Thu Feb 18 17:40:41 1988

Mailer         msgs_from    bytes_from    msgs_to      bytes_to
------------   ---------    ----------    ---------    ----------
local              1            2           1           201
prog               0            0           0             0
uucp               0            0           0             0
tcp                0            0           0             0
```

The fields in the report have the following meanings:

Mailer This field contains the name of the mailer program that handled the mail.

msgs_from This field (**messages from**) contains the number of messages that originated with the indicated mailer.

bytes_from This field contains the number of bytes of information in the messages received from the indicated mailer.

msgs_to This field (**messages to**) contains the number of messages that were sent out from **sendmail** using the indicated mailer.

bytes_to This field contains the number of bytes of information in the messages sent to the indicated mailer.

The collection start time indicated on the second line of the report is the time at which the first update to the empty file was performed.

If **sendmail** transmits mail directly to a file, such as **dead.letter** or an alias target, the message and byte counts are credited to the **prog** mailer in addition to the normal statistics for use of the **prog** mailer.

¦ Copyright IBM Corp. 1985, 1991

6.6.8.2 Statistics Messages

When **mailstats** is called with no program flags, it can generate the following messages:

No statistics data in file "/usr/adm/sendmail/sendmail.st"
The **sendmail** program has not written any data into the statistics file.

mailstats: file size change; use previous mailstats version
The statistics file format is not the format expected by **mailstats**. Try using a previous version of **mailstats** to read it.

¦ Copyright IBM Corp. 1985, 1991

6.6.9 Managing the Mail Queue

The **sendmail** program processes the mail queue automatically at intervals determined by a value that you provide to the **sendmail** daemon when it starts (see "Determining the Queue Processing Interval" in topic 6.6.9.1). You can also process the mail queue by starting **sendmail** directly (or from **cron**) with the **-q** flag. When **sendmail** processes the queue, it:

1. Reads and sorts the queue.

2. Processes the jobs in order on the queue:

    a. Checks to see if the job is locked.
    b. If it is not locked, **sendmail** processes the job.

Locking the jobs allows more than one queue processor to be active at the same time. You do not need to wait for one queue processor to finish before starting another.

In most cases, **sendmail** manages the mail queue without need for you to get involved. However, for some unusual circumstances you may need to process the mail queue manually. For example, if a major system on your network is not operating for a long time, the queue at your system may become loaded with messages routed through that system. With a huge backlog of messages to deliver, performance on your system may go down. When the remote system recovers, **sendmail** delivers all the messages to that system. However, you may want to intervene to improve system performance until the remote system recovers (see "Moving the Queue" in topic 6.6.9.4).

Subtopics
6.6.9.1 Determining the Queue Processing Interval
6.6.9.2 Displaying the Mail Queue
6.6.9.3 Examining the Message Queue Files
6.6.9.4 Moving the Queue
6.6.9.5 Flushing the Mail Queue

¦ Copyright IBM Corp. 1985, 1991

6.6.9.1 Determining the Queue Processing Interval

The interval at which the **sendmail** daemon processes the mail queue is determined by the value of the **-q** flag when the program starts. You start the daemon by running the shell program **/etc/rc.sendmail** which contains a default value for the queue processing interval (typically about 30 minutes). If this value does not suit your system, you can change it (if you are a member of system group):

1. Edit **/etc/rc.sendmail**.

2. Near the beginning of the file, find a line that assigns a value to the variable **qpi**, such as:

   ```
   qpi=30m
   ```

   This value indicates a queue processing interval of 30 minutes.

3. Change the value assigned to **qpi** to a number of minutes to suit your needs. You can also change the letter **m** to change the unit of time (minutes in this case) to hours or another unit. See "Specifying Time Values to Sendmail" in topic 6.6.9.1.1 for the format of values for this and other time parameters.

4. Save the file and exit the editor.

For the changes to have an effect, you must kill the current **sendmail** daemon and start a new one:

1. Use the **ps** command to determine the process ID of the current **sendmail** daemon.

2. Use the **kill** command to kill the current **sendmail** daemon.

3. Start a new **sendmail** daemon:

   ```
   sh /etc/rc.sendmail
   ```

Subtopics
6.6.9.1.1 Specifying Time Values to Sendmail

¦ Copyright IBM Corp. 1985, 1991

6.6.9.1.1 Specifying Time Values to Sendmail

All time intervals are set using a number followed by a letter to designate the time unit being used. For example, **10m** represents ten minutes, where **2h30m** represents two hours and 30 minutes. If you don't provide a letter to specify the units, **sendmail** uses **days**. For example, **10** represents ten days. The full set of scales is:

s seconds

m minutes

h hours

d days

w weeks

¦ Copyright IBM Corp. 1985, 1991

6.6.9.2 Displaying the Mail Queue

Use the **mailq** command to print a listing of the current contents of the mail queue. The format of this command is:

```
/usr/lib/mailq
```

This command produces a listing of the content of the mail queue as shown in the following example listing:

```
                Mail Queue (1 request)
--QID-- -Size- -----Q-Time----- ------------Sender/Recipient------------
AA00269      6 Thu Dec 17 10:01 geo
               (Deferred:  Connection timed out during user open with zeus)
                                  amy@zeus
```

Table 6-4 shows the meanings of the fields in this listing.

Table 6-4. Mail Queue Fields

Field	Meaning
QID	This column contains the message queue ID of the message.
Size	This column contains the number of bytes in the text portion of the message (heading information not included).
Q-Time	This column contains the date that the message entered the queue.
Sender/Recipient	This column contains the user ID of the sender (first) and then the user ID of the receiver of the message. In this example, a message also appears to indicate the status of the message.

¦ Copyright IBM Corp. 1985, 1991

6.6.9.3 Examining the Message Queue Files

The **sendmail** program normally keeps its mail queue files in the directory **/usr/spool/mqueue** on each site. If you are running TCF, you can elect to have one primary spool site keep the master mail queue for the entire cluster. (See "Setting Configuration Options" in topic 6.6.10.1.5 for information on how to change the configuration options used by **sendmail**.) You must be a member of the system group to work with the **/usr/spool/mqueue** directory. Each message in the message queue has a number of files in this directory. The files are named with a prefix letter, the letter **f**, and the message queue ID of the message in the following format:

typefID

In this format **ID** is the message queue ID, **f** is a fixed letter, and **type** can be one of the following letters that indicate the type of file it is:

d This is the data file containing the message body without the heading information.

l This is a lock file. This file may not be present. If it is present, the job is currently being processed and a queue run will not process the file. For that reason, an extraneous **l** file can cause a job to apparently disappear without timing out.

n This file is created when an id is being created. Because it is present for only a short period of time, you may not see an **n** file unless an error occurred. It is a separate file to ensure that no mail can ever be destroyed due to a race condition.

q This is the queue control file. This file contains the information necessary to process the job (see "Examining the q File" in topic 6.6.9.3.1).

t This is a temporary file. This file is an image of the **q** file when it is being rebuilt. It is renamed to a **q** file very quickly.

x This is a transcript file that exists during the life of a session showing everything that happens during that session.

As an example, if a message has a queue ID of **AA00269**, then the following files are created and deleted in the mail queue directory while **sendmail** tries to deliver the message:

File	Function
dfAA00269	Data file
lfAA00269	Lock file
nfAA00269	Back up file
qfAA00269	Control file
tfAA00269	Temporary file
xfAA00269	Transcript file

Subtopics
6.6.9.3.1 Examining the q File

¦ Copyright IBM Corp. 1985, 1991

6.6.9.3.1 Examining the q File

The **q** file contains a series of lines each beginning with a code letter. The code letters are as follows:

D This line contains the name of the data file. There can be only one of these lines.

H This line contains a heading definition. There may be any number of these lines. The order in which the **H** lines appear determines their order in the final message. These lines use the same syntax as heading definitions in the configuration file (see "Changing the Sendmail Configuration File" in topic 6.6.10).

M This line contains a message printed by the **mailq** command. It is mainly used to store status information.

P This line contains the priority of the current message. The priority is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.

R This line contains a receiver address. There is one line for each receiver.

S This line contains the sender address. There can only be one of these lines.

T This line contains the job creation time that is used to compute when to time out the job.

As an example, the following is a **q** file sent to **amy@zeus**:

```
P217031
T566755281
DdfAA00269
MDeferred:  Connection timed out during user open with zeus
Sgeo
Ramy@zeus
H?P?return-path:  <geo>
Hreceived:  by george (0.13 (NL support)/0.01)
        id AA00269; Thu, 17 Dec 87 10:01:21 CST
H?D?date:  Thu, 17 Dec 87 10:01:21 CST
H?F?From:  geo
H?M?message-id:  <8712171601.AA00269@george>
HTo:  amy@zeus
Hsubject:  test
```

This example shows the

1. The priority of the message (**P217031**)

2. The submission time in seconds (**T566755281**)

3. The name of the data file (**DdfAA00269**)

4. A status message:

   MDeferred: Connection timed out during user open with zeus

¦ Copyright IBM Corp. 1985, 1991

5. The ID of the sender (**Sgeo**)

6. The ID of the receiver (**Ramy@zeus**)

7. Heading information for the message (lines beginning with **H**).

¦ Copyright IBM Corp. 1985, 1991

6.6.9.4 Moving the Queue

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in **sendmail** spending a large amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the host returns to service. Use the following procedure to recover from this situation:

1. Use the **ps -e** command to determine the process ID of the **sendmail** daemon, and then use the **kill** command to kill the existing daemon.

2. Move the entire queue directory using the following commands:

   ```
   cd /usr/spool
   mv mqueue omqueue
   ```

3. Start a new daemon:

   ```
   sh /etc/rc.sendmail
   ```

4. When the other system returns to operation, use the following command to run the old mail queue:

   ```
   /usr/lib/sendmail -oQ/usr/spool/omqueue -q
   ```

   The **-oQ** flag specifies an alternate queue directory and the **-q** flag specifies to run every job in the queue. To get a report about progress of the operation use the **-v** flag. This operation can take a long time.

When the queue is empty and you have processed the old log files according to your local procedures (saving them to another directory or to diskette), remove the log files and the temporary directory by entering the following commands:

```
rm /usr/spool/omqueue/*
rmdir /usr/spool/omqueue
```

¦ Copyright IBM Corp. 1985, 1991

6.6.9.5 Flushing the Mail Queue

The mail queue contains a time value that represents the time of submission, rather than the amount of time left until timeout. As a result, you can flush messages that have been in the queue for a short period by running the queue with a short message time-out. For example, the following command runs the queue, flushes anything that is at least one day old (**-oT1d**), and tries to send all messages in the queue that have not exceeded the time-out value:

```
/usr/lib/sendmail -oT1d -q
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10 Changing the Sendmail Configuration File

This section describes the configuration file **sendmail.cf**. Examples that appear in this section are taken from the default configuration file, **/usr/adm/sendmail/sendmail.cf**, that is installed with the **sendmail** program.

The configuration file consists of a series of control lines, each of which begins with a single character that defines how the rest of the line is used. Lines beginning with a space or a tab are continuation lines. Blank lines and lines beginning with a # are comments. The control line can be used for the following functions:

Defining macros and classes for use within the configuration fil

Setting options used by **sendmail**

Defining mail delivery informatio

Defining how **sendmail** interprets address information.

The following sections explain how each of these functions are implemented in the configuration file.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.6.10.1 Editing sendmail.cf with the edconfig Command

The **/usr/lib/edconfig** command provides a menu interface to defining some of the parameters in the **sendmail** configuration file as listed in Table 6-5. In addition, the program provides descriptive text to help you enter the correct information. Use this program to edit the local configuration file or to set up a configuration file (using any file name) to be used on another system. The configuration file must be in the format of the configuration file provided with the **sendmail** program. You must be a member of system group to edit **/usr/adm/sendmail/sendmail.cf** with this or any other program. Use the following procedure to edit the configuration file with this program.

Using edconfig

1. Change to the **/usr/adm/sendmail** directory.

   ```
   cd /usr/adm/sendmail
   ```

2. Start **edconfig** with the configuration file:

   ```
   /usr/lib/edconfig sendmail.cf
   ```

   The **edconfig** program starts and displays a menu.

3. Follow the instructions on the screen to find and change the desired configuration parameters.

4. When you have finished, exit the program by entering the number for writing the configuration file and exiting. If you do not want to keep the changes that you made, enter the number for exiting without writing the configuration file.

Table 6-5. Configuration Parameters That You Can Change with edconfig

Symbol	Meaning	More Information
Dw	Host name macro	"Changing the Host Name Macro" in topic 6.6.10.1.1
Cw	Host name class	"Changing the Host Name Class" in topic 6.6.10.1.2
DD	The domain name macro	"Changing the Domain Name Macro" in topic 6.6.10.1.3
DE	A macro that defines the first part of the domain name	"Changing the Domain Name Part Macros" in topic 6.6.10.1.4

Copyright IBM Corp. 1985, 1991

DF	A macro that defines the second part of the domain name	"Changing the Domain Name Part Macros" in topic 6.6.10.1.4
DG	A macro that defines the third part of the domain name	"Changing the Domain Name Part Macros" in topic 6.6.10.1.4
DH	A macro that defines the fourth part of the domain name	"Changing the Domain Name Part Macros" in topic 6.6.10.1.4
OL	Operational logging level	"Changing the Operational Logging Level" in topic 6.6.10.1.5
Od	Default delivery mode	"Setting Delivery Mode" in topic 6.6.10.1.5
OA	Alias file path	"Setting Configuration Options" in topic 6.6.10.1.5
OS	Statistics file path	"Setting Configuration Options" in topic 6.6.10.1.5
OQ	Queue file path	"Setting Configuration Options" in topic 6.6.10.1.5
OT	Maximum mail retention time in queue (queue time-out)	"Setting Configuration Options" in topic 6.6.10.1.5
Oc	Queue use of expensive mailers	"Setting Configuration Options" in topic 6.6.10.1.5
DZ	Configuration file revision level	"Changing the Configuration File Revision Level" in topic 6.6.10.1.6

Subtopics
6.6.10.1.1 Changing the Host Name Macro
6.6.10.1.2 Changing the Host Name Class
6.6.10.1.3 Changing the Domain Name Macro
6.6.10.1.4 Changing the Domain Name Part Macros
6.6.10.1.5 Setting Configuration Options
6.6.10.1.6 Changing the Configuration File Revision Level

¦ Copyright IBM Corp. 1985, 1991

6.6.10.1.1 Changing the Host Name Macro

Note: As a general guideline to avoid confusion in names for your system, the following names should all be the same:

nodename Built into the kernel. Changed with **chparm nodename=node** and displayed with **uname -n**. Must be ASCII characters only.

hostname The name that the Interface Program for use with TCP/IP uses to refer to your system, defined and displayed by the **hostname** command (usually run from **/etc/rc.tcpip**). Must be ASCII characters only.

If you have TCF installed, this name is the name of your cluster. It is not the name of any one site.

The host name macro, **w**, specifies the name of your host system that is used in the return address of all messages that you generate. Define this macro to be the name of your system returned by the **uname -n** command if TCF is not installed or a single global name applied to the whole cluster (not just the name of a particular site within the cluster) if TCF is installed. Use the following procedure to define that macro.

1. If TCF is not installed, determine your host name with the **uname -n** command, for example:

```
uname -n
george
_
```

or:

1. If TCF is installed, use the global name for your TCF cluster.

2. Start the **edconfig** command (see "Editing sendmail.cf with the edconfig Command" in topic 6.6.10.1) and edit the host name macro to match the name you obtained in Step 1. Use the following format:

```
Dw hostname
```

   In this format, **hostname** is the name returned from the **uname** command. For example, the following entry defines the host name for system **george**.

```
Dwgeorge
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.1.2 Changing the Host Name Class

The host name class, **Cw**, specifies the name and all aliases for your host system. If your system uses different names for two different network connections, enter both names as part of the host name class. If you do not define both names, mail sent to the undefined name is returned to the sender. Define this class to be the name of your system returned by the **uname -n** command and a list of other aliases that your system uses. The name of your system refers to your individual machine unless you have TCF installed. If you have TCF installed, you should define the host name as the name by which your cluster is known by other systems and a list of other aliases that your system uses. Use ASCII characters for host names. Use the following procedure to define that macro.

1. If you are not running TCF, determine your host name with the **uname -n** command, for example:

```
uname -n
george
```

or:

1. If you are running TCF, decide upon a suitable global name for the cluster.

2. Determine any aliases that your system uses.

3. Start the **edconfig** command ("Editing sendmail.cf with the edconfig Command" in topic 6.6.10.1) and edit the host name class to match the name determined by Step 1. Use the following format:

```
Cw hostname alias1 ... aliasn
```

   In this format, **hostname** is the name returned from the **uname** command and **alias1** through **aliasn** are alternate names by which your system is known. For example, the following entry defines the host name class for system **george**.

```
Cwgeorge local
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.1.3 Changing the Domain Name Macro

The domain name macro, **DD**, specifies the full domain name of your local domain. Use ASCII characters for domain names. The domain name is a series of names separated by periods (.) in the following format:

name1.name2.name3.name4

You do not need to have all of the parts in your domain name, but you must specify them if they exist. For example, the following are all properly formatted domain name macro definitions:

DDpub.aus.ibm.com

or:

DDrch.ibm.com

or:

DDacct.xyz

Refer to *Using the AIX Operating System* for a discussion of domain addressing, and to *AIX TCP/IP User's Guide* for information to set up the domain name server for your network.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.1.4 Changing the Domain Name Part Macros

The domain name part macros specify the individual parts of the full domain name as defined in "Changing the Domain Name Macro" in topic 6.6.10.1.3. You must define domain name part macros if they exist in the full domain name macro. The values that you enter for each of the domain name part macros must be the same as the corresponding parts that you specify for the full domain name. If you use less than four parts for your full domain name, always begin with the **DE** macro and define as many other macros in order (**DF**, **DG**) as needed. Define the domain name part macros with entries in the following form:

```
DE name1
DF name2
DG name3
DH name4
```

For example, for a domain name of **pub.aus.ibm.com** enter the following macro definitions:

```
DEpub
DFaus
DGibm
DHcom
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.1.5 Setting Configuration Options

You can set configuration options for use by **sendmail** with a control line in the configuration file. The options that can be set are the same as those options that you can specify with the **-o** flag to the **sendmail** command. An option is named with a single character. The format of the set option control line is:

O**oVAL**

In this format, **o** is a single character name for the option being set and **VAL** is either a string, an integer, a time interval or a Boolean option. Legal values for a Boolean option are **t** or **T** for a true value and **f** or **F** for a false value. If you do not specify a value for a Boolean variable, it becomes a true value.

For example, the following entries from the default **sendmail.cf** file show the format of the set option control line:

OL9 Sets the log level option variable **L** to a value of 9. This entry occurs early in **sendmail.cf** to ensure that the log is maintained during the reading of the file. See "Logging Mail System Activities" in topic 6.6.7 for more information about the **sendmail** logging facility.

OQ/usr/spool/mqueue
Sets the local mail queue directory options variable **Q** to a string (**/usr/spool/mqueue**) that defines where the mail log is to be kept.

OA/usr/adm/sendmail/aliases
Sets the option variable **A** to the full path name of the aliases file (**/usr/adm/sendmail/aliases**).

Oh/<specific local directory>
Sets the master mail queue directory options variable **h** to a string (**/<specific local directory>/spool/mqueue**) that defines where the master mail log is to be kept.
<specific local directory> is the name of the <LOCAL> for that machine where the master mail log should be stored. (Conventionally, this local directory is the same as the sitename.)

Setting Delivery Mode: The **sendmail** program can operate in several delivery modes. The default configuration file sets delivery mode to **b**. However, you can change the delivery mode with the **d** option in the configuration file. These modes specify how quickly mail will be delivered. Legal modes are:

i Deliver interactively (wait until the mail is delivered)

b Deliver in background (run delivery in the background)

q Queue only (let the queue processor deliver the mail).

Mode **i** passes the maximum amount of information to the sender but is hardly ever necessary. Mode **q** puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval (see "Determining the Queue Processing Interval" in topic 6.6.9.1). Mode **b** is probably a good compromise. However, this mode can cause large numbers of

¦ Copyright IBM Corp. 1985, 1991

processes if you have a mailer that takes a long time to deliver a message.

Setting Time-outs: The **sendmail** program can time out when reading standard input or when reading from a remote SMTP server. The default configuration file sets this value to 5 minutes. This value should be correct for most situations. However, if you need to change the time-out value, change the **r** option in the configuration file.

After sitting in the queue for a few days, a message times out. The **sendmail** program notifies the sender of the message that it could not be sent. The time-out is typically set to three days. Set this time-out with the **T** option in the configuration file.

Changing the Operational Logging Level: The **OL** option specifies the log level to be used when **sendmail** is running. Valid levels and the activities that they represent are (each number includes the activities of all numbers of lesser value and adds the activity that it represents):

Level	Activity Logged
0	No logging.
1	Major problems only.
2	Message collections and failed deliveries.
3	Successful deliveries.
4	Messages being deferred (due to a host being down, etc.).
5	Placing messages in the queue (normal event).
6	Unusual but benign incidents (trying to process a locked file, etc.).
9	Log internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts.
12	Several messages that are of interest when debugging.
16	Verbose information regarding the queue.

The default level is **OL9**.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.1.6 Changing the Configuration File Revision Level

The configuration file revision level macro, **Z**, helps you track changes that you make to the configuration file. Each time that you make a change to the configuration file, you should also change the value of this macro. You can choose any format for the number that you define. For example, if the configuration file is at level **3.1**, the following entry should be in the configuration file:

```
DZ3.1
```

You can also use a text string for this macro:

```
DZversion_one
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.2 Editing sendmail.cf with a Text Editor

To make changes to **sendmail.cf** other than those that are permitted by the **edconfig** command, you must use a text editor. Most of these parameters should not be changed casually, since they affect how **sendmail** interfaces with the system and networks. Read the following sections carefully to determine whether you need to change any of these parameters. In most cases, you do not need to change them to get your mail system working.

When using an editor on a configuration file that did not come with the **sendmail** installation, be aware that the tab character will be defined as the field separator character in rulesets when **sendmail** reads the file. Some editors (including the **INed** editor, which is also available with the AIX Operating System) store tabs as the number of spaces they represent instead of the tab character.

Do not use such editors to edit the **sendmail.cf**, or when you try to build the configuration file using **/usr/lib/sendmail -bz**. You will get a large stream of error messages, and the data base version of the configuration file will not be built.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.3 Defining Macros and Classes

A ***macro*** is a symbol that represents a value or string. A macro is defined by a **D** control line. A ***class*** is a symbol that represents a set of one or more words. Classes are used in pattern matching when the **sendmail** program is parsing addresses. Macros are not expanded until **sendmail** loads the rulesets when it starts up. The following letters introduce configuration file control lines that define macros and classes to set up the **sendmail** program:

DxVAL Defines symbol **x** to be a macro and assigns a value of **VAL** to it. If a second **Dx** defines the same symbol, the second definition replaces the first definition.

Cc string Defines symbol **c** to be a class and assigns a word or group of words (**string**) to it. If a second **Cc** defines the same symbol, the **string** from the second definition is added to the **string** from the first definition. No words are deleted from the class definition.

Fc file Defines symbol **c** to be a class and assigns a word or group of words listed in a separate file to the symbol.

The symbols must be a single character selected from the ASCII set. Use uppercase letters for macros and classes that you define. Lowercase letters and special symbols are macros and classes defined by the program (see "Understanding System-Defined Macros" in topic 6.6.10.3.4).

To use a macro or class in a control line, put a dollar sign (**$**) before its name. For example, if the name of the macro is **x**, use **$x** when using that macro in a control line. Without the preceding **$**, the program interprets **x** as only the letter x. To specify conditional expressions, use the format:

$?**x text1** $| **text2** $.

In this format, the symbols have the following meaning:

$? If

x The macro being tested (for example, if **x** is defined)

text1 The pattern to be used if **$x** is defined

$| Else

text2 The pattern to be used if **$x** is not defined

$. Specifies the end of the conditional expression.

The else clause ($| **text2**) is not required.

Note: Do not use any of the characters defined as tokens (by the required macro **o**; see "Defining Required Macros" in topic 6.6.10.3.5) when defining a word in a class. The **sendmail** program may not be able to read the definition correctly.

Subtopics
6.6.10.3.1 Creating a Macro
6.6.10.3.2 Creating a Class Using a List

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

6.6.10.3.1 Creating a Macro

Use a control line that begins with the letter **D** to define a macro. The syntax for the **D** macro definition is:

D**xVAL**

In this format, **x** is the name of the macro, and **VAL** is the value to assigned to it.

For example, the following control line in the configuration file defines a macro **D**, representing the name of the local domain, to be the name assigned to that domain. In this example, the local domain is assigned the name **802**, but the name could be any alphanumeric string that you or your network administration chooses.

```
DD802
```

The following example shows the use of a previously defined macro to define a new macro. It defines a required macro **j**, representing the official name for the local site, to be the value returned by the **sendmail** macro **$w** (the host name of the local site).

```
Dj$w
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.3.2 Creating a Class Using a List

Use a control line that begins with the letter **C** to define a class with a specified list of members. The syntax for the **C** macro definition is:

```
Cc word1 word2 ...
```

In this format, **c** is the name of the class that matches any of the named words **word1**, **word2**, and so forth. Any word cannot include characters defined as delimiters, such as period (.). You can spread the assignment of words to the class across many lines. For example, the control line:

```
CDLOCAL 802
```

defines the class **D** to represent the class of names **LOCAL** and **802**. You can accomplish the same function with two separate control lines:

```
CDLOCAL
CD802
```

Use ASCII characters for file names to ensure that users of different locales can access these files successfully.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.3.3 Creating a Class Using a File

Use a control line that begins with the letter **F** to define a class whose members are listed in an external file. The syntax for the **F** class definition is:

```
Fc file
```

In this format, **c** is the name of the class that matches any of the words listed in the file specified by **file**.

Use ASCII characters for file names to ensure that users of different locales can access these files successfully.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.3.4 Understanding System-Defined Macros

The **sendmail** program defines some macros internally for use in writing stanzas in the configuration file. You do not need to define these macros in the configuration file, but you can use the symbols when writing other macros and parsing rules in the configuration file. The following paragraphs describe these macros.

Date Macros: The **sendmail** program defines three macros that represent dates. These macros are:

a The origination date in Arpanet format:

Fri, 9 Oct 87 09:25:47 CDT

This date is the time extracted from the **Date:** line of the message (if there is one). If no **Date:** line is found in the incoming message, **$a** is set to the current time.

b The current date in Arpanet format:

Fri, 9 Oct 87 09:25:47 CDT

This macro is used for postmarks.

d The current date in **ctime** format:

Fri Oct 9 09:25 CDT 1987

Macros Relating to the Sender: The **sendmail** program defines the following macros that identify the sender of the message:

f This macro contains the sender's ID.

g This macro contains the sender's address relative to the receiver.

s This macro contains the name of the sender's host system.

x This macro contains full name of the sender. The **sendmail** program determines this name in one of the following ways:

It can be passed as a flag to **sendmail**.
It can be the value of the **Full-name:** line in the message heading.
It can be the comment field of a **From:** line.
If the message originated locally, the full name can be looked up in **/etc/passwd**.

Macros Relating to the Receiver: The **sendmail** program defines the following macros that identify the receiver of the message:

h This macro contains the name of the receiving host as set by the **$@** part of the mailer resolution line in ruleset 0.

u This macro contains the ID of receiving user as set by the **$:** part of the mailer resolution line in ruleset 0.

z If the message is a local message, this macro contains the home directory of the receiving user.

Use ASCII characters for these file names.

¦ Copyright IBM Corp. 1985, 1991

Macros Relating to Message Routing: The **sendmail** program defines the following macros to help it track and route messages through the network:

c This macro contains the hop count. The ***hop count*** is the number of times that this message has been processed and usually corresponds to the number of separate timestamps in the message (some mail handler programs allow you to change the hop count).

i This macro contains the mail queue ID of the message on this host. This macro is useful for tracking messages when put into the timestamp line of the message.

p This macro contains the process ID of the **sendmail** program. The **sendmail** program uses this macro, along with the **$t** macro to create unique strings for the **Message-ID:** field in the message.

r This macro contains the protocol used to communicate with **sendmail**. This macro is not supported.

t This macro contains a numeric representation of the current time. The **sendmail** program uses this macro, along with the **$p** macro to create unique strings for the **Message-ID:** field in the message.

v This macro contains the version name and number of the AIX Operating System running on the system.

w This macro contains the name of the local host.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.3.5 Defining Required Macros

The configuration file that comes with **sendmail** defines the following macros to transmit information to **sendmail**. If you create a new configuration file, be sure to include definitions for these macros:

e The SMTP entry message

j The official domain name (in ASCII) for this site (in ASCII)

l The format of the **From** line (not the **From:** line)

n The name of the daemon (for error messages)

o The set of operators in addresses

q The default format of sender address (in ASCII)

The following paragraphs describe the use and definition of these macros.

Defining the e Macro: The **e** macro defines the message that the **sendmail** SMTP daemon prints out when another system connects to it. The first word must be the **$j** macro. The definition for this macro in the default configuration file is:

```
De$j Sendmail $v/$Z ready at $b
```

This control line defines the required macro **e** to be the following string:

The official domain name for the site (defined by the **$j** macro),
The word **Sendmail**,
The version number of AIX Operating System (defined by the **$v** macro),
A separating slash **/**) character,
The version number of **sendmail.cf** (defined by the **$Z** macro),
The words **ready at**,
The current date (defined by the **$b** macro).

Defining the j Macro: The required macro **j** defines the official name for the local system. It should be in the same format as all system addresses:

host.domain

In this format, **host** is any name of the local system, and **domain** is the hierarchical domain address. Use ASCII characters for the host name and the domain name. Since this information is already defined in the system, the default configuration file defines the **j** macro to be the value returned by the **sendmail** macro **$w** (the host name of the local site):

```
Dj$w
```

Defining the l Macro: The required macro **l** defines the format for the **From** line in the message heading. In the default configuration file, this definition appears as follows:

```
DlFrom $g  $d
```

This example uses two **sendmail** macros: **$g** is the address of the sender relative to the receiver; **$d** is the date in **ctime** format. This entry results in a line like the following line appearing at the beginning of

¦ Copyright IBM Corp. 1985, 1991

the message:

```
From root Tue Nov  3 14:24:05 CST 1987
```

Notes:

1. Some mailers include a **From** line that they generate, ignoring the **sendmail From** line. In that case, changing the **l** macro has no effect on the message heading format.

2. Many mail handler programs use the **From** line as a message delimiter when scanning mailboxes. Changing the format of this line may cause problems for these programs.

3. The **sendmail From** line is used when **sendmail** writes mail directly to a file, such as **dead.letter** or a file defined by an alias.

Defining the n Macro: The required macro **n** defines the name of the user that will receive error messages pertaining to mail system errors. In the default configuration file, this definition appears as follows:

```
DnMAILER-DAEMON
```

Defining the o Macro: The required macro **o** defines a list of characters that are parsed as separate tokens when reading rules in the configuration file. For example, if **r** were in the **$o** macro, then the input, **address**, would be scanned as three tokens: **add**, **r** and **ess**. In the default configuration file, the definition of this macro appears as follows:

```
Do.:%@!^=/][
```

In addition, the following characters are **always** recognized by **sendmail** as tokens:

```
Ctrl-




, (comma
; (semicolon
\ (backslash
" (quote
Carriage retur
Newlin
```

Network organizations, such as ARPA, have standards concerning tokens. Be sure to check with your network organization before changing this macro.

Defining the q Macro: The required macro **q** defines the format of an address in a message if no other format is specified. In the default configuration file, this macro definition appears as follows:

```
Dq$g$?x ($x)$.
```

This entry defines the address to be the address of the sender relative to the receiver (**$g**), and if the **sendmail** variable containing the full name of the sender is defined (**$?x**) that variable is printed within parentheses following the sender address (($x)). The symbol **$.** closes the

¦ Copyright IBM Corp. 1985, 1991

conditional expression introduced by the symbol **$?**. This format could result in the following addresses:

```
amy@athena.802
amy@athena.802 (Amy Smith)
```

In the second example, the full name of Amy Smith is in **/etc/passwd** on system **athena**. In the first example, the full name field for user ID **amy** in **/etc/passwd** is empty.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.4 Defining Message Precedence

The configuration file contains lines to define mail queue precedence for messages that contain a **Precedence:** field. Normally, you do not need to change the values in the default configuration file. The name defined and the numerical value assigned are based on the needs of the network. Higher numbers have higher priority; numbers less than zero indicate that error messages will not be returned to the sender of these messages. Precedence value is zero for any precedence name not defined in this file. For example, the configuration file may contain the following entries:

```
Pfirst-class=0
Pspecial-delivery=100
Pbulk=-60
Pjunk=-100
```

These entries set **special-delivery** as the highest priority message and **junk** as the lowest priority.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.5 Defining Administrative IDs

Administrative IDs are those IDs that can override the sender address using the **-f** flag to the **sendmail** command. The configuration file defines IDs written in ASCII with the **T** control line. For example, the configuration file may contain the following entries:

```
Troot
Tdaemon
Tuucp
```

These entries define IDs **root**, **daemon**, and **uucp** as administrative IDs for the **sendmail** command.

These IDs could have been defined using only one **T** control line:

```
Troot daemon uucp network
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.6 Defining Message Headings

The **sendmail** program expects mail to have the following parts in order:

1. An operating system **From** line (defined by the five characters: F, r, o, m and space)

2. Mail header lines that begin with a keyword followed by a colon, such as **From:** or **To:**

3. An empty line

4. The body of the message.

The **sendmail** program detects the operating system **From** line by checking the first five characters of the first line. After that, header lines are processed. When it detects a line that does not begin with a keyword followed by a colon, it ends header line processing. If an empty line occurs at that point, it is ignored.

Mailer flags or the mailer determine if an operating system **From** line is generated. Other header lines are present (or not) depending upon those defined in the **sendmail** configuration file, those specified by mailer flags, and those present in incoming mail.

Lines in the configuration file that begin with a capital letter **H** define the format of the headers used in messages. The format of the **H** line is:

```
H[?mflags?]fieldname: content
```

In this format, the variable parameters have the following meaning:

mflags This is an optional field. If you supply it, surround it with question marks (?). This field contains mailer flags (see "Specifying Mailer Flags" in topic 6.6.10.7.3) that determine whether this H line is used. If the mailer being used requires the information specified by the mailer flag, then this H line is included when formatting the heading. Otherwise, this H line is ignored.

fieldname This field contains the text that is displayed as the name of the field in the heading information. The actual text used is a matter of choice. Some typical field names include: From, To, and Rcvd From.

content This field defines the information that is displayed following the field name. It usually uses a **sendmail** macro to specify the information.

Use ASCII characters for IDs, site names, host names, and domains to ensure successful communication for users of different locales.

The following example lines are from a typical **sendmail.cf** file:

```
H?P?Return-Path: <$g>
```

This line defines a field called **Return-Path** that displays the content of the **$g** macro (sender address relative to the receiver). The **?P?** portion indicates that this line is only used if the mailer use the **P** flag (the mailer requires a Return-Path line).

¦ Copyright IBM Corp. 1985, 1991

```
  HReceived:  $?sfrom $s $.by $j ($v/$Z)
           id $i; $b
```

This line defines a field called **Received**. This field displays the following information:

$?sfrom $s $. If an **s** macro is defined (sender's host name), display the text **from** followed by the content of **$s**.

by $j Display the text **by** followed by the content of **$j** (official name for this site).

($v/$Z) Display the version of the **sendmail** program (**$v**) and the version of **sendmail.cf** (**$Z**), set off by parentheses and separated by a slash.

id $i; Display the text **id** followed by the content of **$i** (mail queue ID of the message) and a semicolon (**;**).

$b Display the current date.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7 Defining a Mailer

Note: Defining a mailer entry in the configuration does not ensure that it will be used. You must also define parsing rules (see "Changing the Format of Addresses" in topic 6.6.10.8) that resolve to that mailer.

A ***mailer*** is a program that delivers mail either locally or over some type of network to another system. Use control lines that begin with the letter **M** to define the characteristics of a mailer program that interfaces with **sendmail**. The format of a mailer definition control line (written in ASCII) is:

M**mname**, P=**path**, F=**flags**, S=**xx**, R=**yy**,[E=**eol**,] [A=**string**,] [M=**limit**]

The following paragraphs describe the parameters for the mailer definition. Some examples follow those descriptions.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.1 Specifying a Mailer Name

Each mailer must have an internal name. The name can be any string that you choose, except that the names **local** and **prog** are reserved for the mailers for local delivery and delivery to programs. You must provide definitions for these two mailers in **sendmail.cf** if they are not already there (the default configuration file contains these definitions). To define the mailer name, put the name immediately after the **M** in the mailer definition control line:

```
Mmname
```

You must enter this information in ASCII characters.

For example, the following segment introduces the definition line for a mailer called **lan**:

```
Mlan
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.2 Defining the Path to the Mailer Program

Specify the location of the mailer program with the **P** field in the mailer definition. This field has the format:

P=**path**

The **path** defines the full path name of the mailer program on the local system. If the mailer program is the **sendmail** version of SMTP, use the string **[IPC]** as the path. For example, the following two mailer definition fragments define a local mailer at **/bin/bellmail** and another mailer that is the **sendmail** implementation of SMTP.

```
Mlocal, P=/bin/bellmail,
Mlan, P=[IPC],
```

You must enter this information in ASCII characters.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.3 Specifying Mailer Flags

Mailer flags provide further information to **sendmail** about the mailer program being described. Specify mailer flags with the **F** field in the mailer definition. This field has the format:

F=**flags**

Table 6-6 defines the meaning for the flags that **sendmail** recognizes. For example, the following mailer definition fragment uses the flags **rlsm** to indicate that the mailer requires a **-r** flag, does local delivery, needs quotes stripped from addresses, and can deliver to more than one user at a time:

```
Mlocal, P=/usr/bin/mail, F=rlsm,
```

You must enter this information in ASCII characters.

Table 6-6. Mailer Flags

Flag	Meaning
C	If this flag is set, this mailer inspects the address of any incoming mail that it processes for the presence of an @ sign. If it finds an @, it saves the @ and the remainder of the address to be used when rewriting addresses in header lines in the message (when mail is forwarded to ANY mailer). The receiving mailer adds the saved portion of the address to any address that does not contain an @ after the address has been processed by ruleset 3 (this processing does not depend upon a mailer flag; it always occurs). Do not use this flag for general operation, since it does not interpret complex, route-based addresses properly.
D	This mailer needs a **Date:** or **Resent-Date:** header line.
e	This mailer is expensive to connect to. If the **c** configuration option is set, mail for this mailer is always placed in the queue.
E	Escape lines beginning with the exact five characters F, r, o, m, space in the body of the message with a **>** (greater than symbol). This flag allows operating system **From** lines (or any other text lines beginning with those five characters) to appear in the body of the message without being interpreted as the start of a new message.
f	The mailer needs a **-f** flag. The flag is inserted into the call for the mailer followed by the expansion of the **$g** macro (sender's address relative to the receiver).
F	This mailer needs a **From:** or **Resent-From:** header line.

Copyright IBM Corp. 1985, 1991

h	Preserve uppercase letters in host names for this mailer.
I	This mailer uses SMTP to communicate with another SMTP server that is part of the **sendmail** program (not to an SMTP server that is not a part of **sendmail**). When communicating with another **sendmail** program, the mailer can use features that are not part of the standard SMTP protocol. This option is not required, but if this option is omitted the transmission will still operate successfully but not as efficiently as possible.
l	This mailer is local; final delivery will be performed.
L	For SMTP, restricts data transmission and receiving to 7-bit data unless the **N** flag is set. If the **N** flag is set, full 8-bit data transmission is used to support the national language character set. The **L** flag also enforces SMTP line lengths. Lines that are too long are broken up. If the **L** flag is not set, full 8-bit data transmission is used to and from mailer programs.
m	This mailer can be sent to multiple users on the same host in one transaction. When a **$u** macro occurs in the **argv** part of the mailer definition, that field will be repeated as necessary for all qualifying users.
M	This mailer needs a **Message-ID** or **Resent-Message-ID:** header line.
n	Do not insert an operating system **From** line on the front of the message.
N	This mailer supports the national language character set. This flag modifies the **L** flag to allow 8-bit data transmission through SMTP.
p	Use the return-path in the SMTP **MAIL FROM:** command rather than just the return address; although this type of return address is required for the transfer protocol, many hosts do not process return paths properly.
P	This mailer needs a **Return-Path:** header.
r	Same as **f**, but sends a **-r** flag. Specify this flag to pass the name of the sender as a **-r** flag under certain circumstances.
s	Strip quote characters off of the address before calling the mailer.

Copyright IBM Corp. 1985, 1991

S	Don't reset the user ID before calling the mailer. Use this flag in a secure environment where **sendmail** runs as root to help avoid forged addresses. If the mailer must be called as **root**, use the **S** flag.
u	Preserve uppercase letters in user names for this mailer.
U	This mailer needs AIX-style **From** lines with the UUCP-style **remote from <host>** on the end.
x	This mailer needs a **Full-Name:** header line.
X	This mailer uses a hidden dot algorithm that adds an extra dot to the beginning of any line that begins with a dot. The leading dot must then be removed at the other end of the transmission. This method ensures that lines in the message that contain a dot do not end the message prematurely.

You can also define mailer flags to match flags that you define in special header definitions in your **sendmail** configuration file.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.4 Specifying the Rewrite Ruleset for the Mailer

The **sendmail** program uses sets of rewrite rules to change the format of incoming addresses to a style that the receiving mailer program can understand. These rewrite rules are described in "Changing the Format of Addresses" in topic 6.6.10.8. Specify the rewrite ruleset to use on sender addresses for this mailer with the **S** field; specify the rewrite ruleset to use on receiver addresses for this mailer with the **R** field. These fields have the following format:

S=**xx**, R=**yy**,

In this format, **xx** and **yy** are integers that specify a particular ruleset number for processing the addresses for this mailer. These rules can perform operations such as appending the current domain to addresses that do not already have a domain. For example, if the following header were being processed by the local **sendmail** program, it would require further processing before being sent to another location:

From: geo

If it were sent on a domain address network, it could be changed to:

From: geo@zeus.aus

If it were sent on a **uucp** network, it could be changed to:

From: zeus!geo

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.5 Defining a Different End of Line Character

The normal indication of the end of line is a string that contains only the newline character. To change this character, use the optional **E** field. The format of this field is:

E=**eol**,

Must be in ASCII.

In this format, **eol** is the character string that specifies the end of the line. You can use normal \ escape characters to specify the end of line character (\r, \n, \f, \b).

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.6 Passing Information to the Mailer

Specify information to be passed to the mailer with the **A** field. This field has the following format:

A=**string**

In this format, **string** can be any string of words with imbedded spaces allowed. Any or all of the words can be symbols, such as **$u** (receiving user name) defined in **sendmail.cf**. If you do not include this field, or the field does not contain the **$u** symbol, **sendmail** uses the SMTP protocol to send messages to the mailer. If the **P** field for this mailer (access path name) is [IPC], indicating a mailer accessed with interprocess communications, use the following information in this field:

A=IPC $h [**port**]

In this notation **port** is the optional port number to which to connect.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.7 Limiting Message Size

Use the **M** mailer flag to specify the maximum size in bytes of messages that the mailer program handles. For example, the following field specifies a maximum limit of 10,000 bytes:

```
M=10000
```

Note: The number given is the maximum number of bytes of the message, which may not be the same as the maximum number of characters.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.7.8 Example Mailer Specifications

The following mailer specification specifies a local delivery mailer:

```
Mlocal, P=/bin/bellmail, F=lsDFM, S=10, R=20, A=mail $u
```

It is called **local** and is located in the file **/bin/bellmail**. The mailer takes the following flags:

l	Local delivery
s	Strips quote characters from addresses
DFM	Needs **Date:**, **From:**, and **Message ID** fields.

Ruleset 10 should be applied to sender addresses in the message and ruleset 20 should be applied to receiver addresses. Additional information sent to the mailer in the **A** field is the word **mail** and words containing the name of the receiving user. If a **-r** flag is inserted it will be between the words **mail** and **$u**.

The following mailer specification specifies a mailer for local area network delivery:

```
Mlan, P=[IPC], F=meC, S=11, R=21, A=IPC $h, M=100000
```

It is called **lan** and is connected to via the **sendmail** internal SMTP mailer. It can handle multiple users at once (**m**), the connection is defined as expensive (**e**), and any domain from the sender address on incoming mail is saved for use by an outgoing mailer (**C**). Sender addresses should be processed by ruleset 11 and receiver addresses by ruleset 21. There is a 100,000 byte limit on messages passed through this mailer (**M**).

¦ Copyright IBM Corp. 1985, 1991

6.6.10.8 Changing the Format of Addresses

The **sendmail** program can receive addresses in a number of different formats and output them in a format needed for the network protocol (uucp or tcp/ip) or mailer program being used to route the message. To perform this translation, **sendmail** uses a set of rewrite rules that are defined in **sendmail.cf**. The **sendmail.cf** that is installed with the **sendmail** program contains enough rules to perform the translation for uucp and tcp/ip networks using a domain address structure. You should not have to change these rules unless you are connecting to a system that uses a different addressing scheme. This section describes the operation of the address rewrite rules so that you can change them for special cases.

Mail address processing consists of the following steps:

1. Receive the mail through the incoming mailer. Save headers and text.

2. Define the sender.

3. Define all recipients.

4. Transmit the mail to the outgoing mailer, changing headers of messages as they are transmitted.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.6.10.8.1 Define the Sender

The **sendmail** program determines the sender of a message from one of the following sources:

sendmail user ID (mail originator)

sendmail From flag argument (from administrative user)

sendmail SMTP interface for incoming mail

From type header lines inside the mail when **sendmail** is processing in ARPA-FTP mode (**sendmail -ba**).

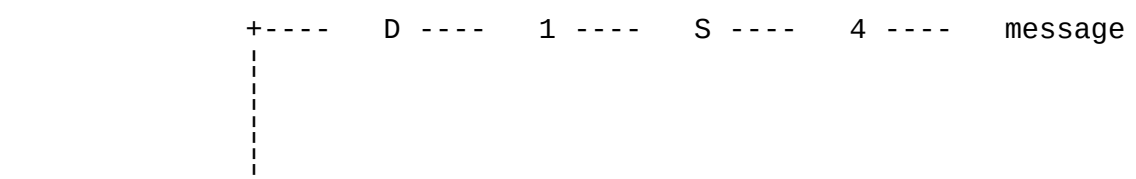
Figure 6-3 shows a flowchart for processing the address of the sender. The following steps occur during processing:

Input	**Apply to Input**	**Resulting Output**
From address	Ruleset 3, ruleset D, ruleset 1, ruleset S, then ruleset 4	Input defined as the **$f** macro

If the mailer has **C** flag, define **from domain** for later usage.

Value of **$f** macro	Search for @ token.	If @ found, save **from domain** as the token string from the first @ to the end of the token string, else, save NULL string.

The **$f** macro represents the sender name as seen from the local host.



D = Rules to add the sender domain
S = Rules (for a specific mailer) for processing sender headings

Figure 6-3. Sender Address Ruleset Processing Flowchart

¦ Copyright IBM Corp. 1985, 1991

6.6.10.8.2 Define the Receiver

The **sendmail** program determines the receiver of a message from one of the following sources:

sendmail mail address parameters.

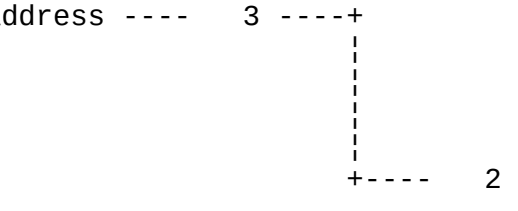
SMTP interface for incoming mail

If the **-t** flag is used, **To** lines inside mail add to the receiver list. All the following processing takes place for each receiver. Each receiver may have a different mailer associated with it.

Figure 6-4 shows a flowchart for processing the address of the receiver. The following steps occur during processing:

Input	**Apply to Input**	**Resulting Output**
To address	Ruleset 3, then ruleset 0	mailer/host/user (defined by the mailer selection rule)

The host part ($@) is placed into the $h macro, and the user part ($:) is placed into the $u macro.



R = Rules (for a specific mailer) for processing receiver headings

Figure 6-4. Receiver Address Ruleset Processing Flowchart

¦ Copyright IBM Corp. 1985, 1991

6.6.10.8.3 Delivering Mail

The deliver function takes place after the sender and all receivers have been registered. Mail is delivered in separate stages for each separate mailer mentioned in the receiver list. For each mailer, the **$g** macro is defined from the **$f** macro, according to the flowchart for a sender type header described in "Processing Headers" in topic 6.6.10.9. Then all mail for that mailer is sent. As the mail is sent, header lines are processed.

If mail is sent through **sendmail**'s SMTP interface, then the SMTP **MAIL FROM** command uses the expanded **$g** macro for the address to be presented to the other SMTP program. The SMTP **RCPT TO** command uses the header line processing sequence for **To** type headers to prepare the address for the other daemon.

¦ Copyright IBM Corp. 1985, 1991

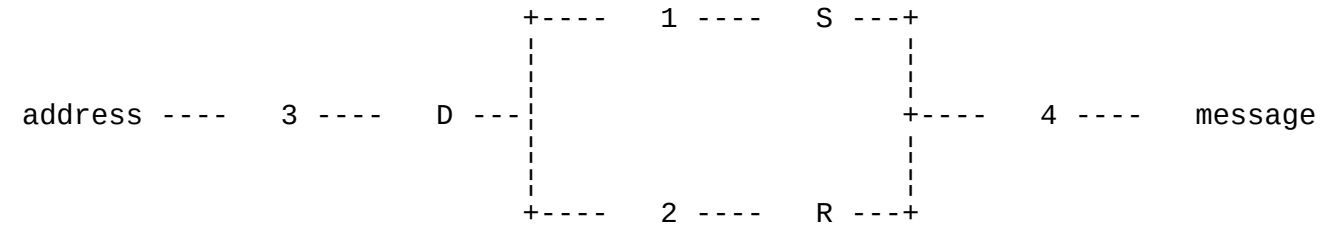
6.6.10.9 Processing Headers

As mail is delivered to an outgoing mailer, headers collected from the received mail are sent to the outgoing mailer. (This process can generate new headers, also.) As the headers go out, they are changed into a form that is suitable for the next host. Also, after each header is changed, any resulting macro references in the final result are expanded.

Headers may include reference to macros (such as **$g**) that get created using rewrite rules. As the header is processed, segments of the header string may get processed twice by the same rulesets.

This process is repeated for each separate mailer. Figure 6-5 shows the flowchart for processing headers. The following steps occur during processing:

Input	**Apply to Input**	**Resulting Output**
To header address	Ruleset 3, D processing, ruleset 2, mailer ruleset R, then ruleset 4	Proper header format for outgoing mail
From header address	Ruleset 3, D processing, ruleset 1, mailer ruleset S, then ruleset 4	Proper header format for outgoing mail



D = Rules to add the sender domain
S = Rules (for a specific mailer) for processing sender headings
R = Rules (for a specific mailer) for processing receiver headings

Figure 6-5. Header Ruleset Processing Flowchart

For automatic appending of sender domain, all the following must be true:

- The receiving mailer at this host must have been configured with the **C** mailer flag.

¦ Copyright IBM Corp. 1985, 1991

There must have been a domain @ token) present in the sender address of a message received through that mailer.

A receiver for that message must not have any domain at all (no @ token). This takes place on a per receiver basis, since any message may have more than one receiver.

The test for the presence of a domain in the receiver address is to check for an @ token. The header processing is the only processing where automatic appending of sender domain can occur.

Subtopics
6.6.10.9.1 Defining Rewrite Rules
6.6.10.9.2 Defining an Input Pattern
6.6.10.9.3 Defining an Output Pattern
6.6.10.9.4 Naming a Set of Rewrite Rules
6.6.10.9.5 Testing the Rewrite Rules

¦ Copyright IBM Corp. 1985, 1991

6.6.10.9.1 Defining Rewrite Rules

Use control lines that begin with the letter **R** to define rewrite rules in the configuration file. The format of a rewrite rule control line is:

R left_side <tab> right_side <tab> comments

The **left_side** contains a set of characters that define an incoming address format. The **right_side** contains a set of characters that define the format to which the incoming address is to be changed. The **sendmail** program ignores the comment field. The following example is a rule from the **sendmail.cf** file:

```
R$+!$+          $:$2<@$1.UUCP>          resolve uucp names
```

The meanings of the symbols are explained in "Defining an Input Pattern" in topic 6.6.10.9.2 and "Defining an Output Pattern" in topic 6.6.10.9.3. The parts of this line are:

R The letter R designates that this is a rule line.

$+!$+ This set of characters is the left side of the rule. It defines the pattern to match against an address.

$:$2<@$1.UUCP>
This set of characters is the right side of the rule. It defines the pattern to replace the address matched by the left side.

resolve uucp names
This phrase is the comment of the rule. It is ignored by **sendmail** and may explain the purpose of the rule.

When **sendmail** processes an address it tries to match the pattern of that address format to a pattern specified by the left side of one of the rewrite rules in **sendmail.cf**. When it finds a rule that matches, it changes the format of the incoming address to the format specified by the right side of that rewrite rule.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.9.2 Defining an Input Pattern

The left side of a rewrite rule defines an pattern to match against the address that **sendmail** is currently examining. The pattern uses a combination of literal characters, words and special symbols. All special symbols are two characters that begin with a dollar sign ($). These special symbols are:

$* Match zero or more tokens
$+ Match one or more tokens
$- Match exactly one token
$=x Match any token in class **x**
$~x Match any token not in class **x**

A ***token*** is a set of one or more characters that **sendmail** treats as a single unit when rewriting the address. For example, if the character @ (at sign) has been defined as the field delimiter in an address, then in the address, **amy@zeus**, **sendmail** recognizes three tokens: **amy**, @, and **zeus**.

If the input address matches the left side of a rule, the token sequence that matches each special symbol is saved in a numbered special symbol **$n**. In this format, **n** is an integer that indicates the position of the special symbol in the left side (counting from left to right). Each of the numbered special symbols is passed to the right side of the rule for reformatting. For example, the left side of a rule may be the following expression:

```
$-@$+
```

This rule can be interpreted as:

1. Match any one token and assign it to the symbol **$1**
2. Look for a literal @ as the next token.
3. Match one or more tokens following the @ and assign them to the symbol **$2**.

Apply this rule to the following address:

```
amy@zeus.aus.IBM.COM
```

This address contains six tokens: amy, @, zeus, aus, IBM and COM. The periods delimit the tokens in the domain name. The rule matches the input exactly. The values passed to the right side of the rule are:

$1 amy
$2 zeus.aus.IBM.COM

This rule receives a domain-format address and separates the user ID from the domain address.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.9.3 Defining an Output Pattern

When the left side of the rule matches the input, the input that matched special characters is saved in the special symbols of the form **$n**, the rest of the input is deleted, and the address is built again using the instructions in the right side.

The right side of a rewrite rule defines a pattern to create a new address from the saved portions of the input address. The pattern uses a combination of literal characters, words, and special symbols. All special symbols begin with a dollar sign ($). These special symbols are:

$n Substitute saved token **n** from left side.

$[name$] Get an address in standard format for **name**. The **sendmail** program uses the **gethostbyname** subroutine to resolve a possible alias to the official name of the host. If the address is an internet protocol address, it uses the **gethostbyaddr** subroutine to try to convert this address to a domain name, if possible.

$>n Call ruleset **n**. This symbol substitutes the remainder of the address as indicated by the remaining symbols and then passes it as the address to be examined by ruleset **n**. The final value of ruleset **n** then becomes the output address for this rule.

$#mailer$@host$:user
Use this complex symbol only in ruleset 0 (see "Changing the Format of Addresses" in topic 6.6.10.8). It stops evaluation of the ruleset and indicates to **sendmail** that the address is resolved to a mailer. This special symbol specifies the mailer, host, and user necessary to direct the mailer. If the mailer is local, the host may be omitted. The **mailer** and **host** must each be a single word, but **user** may contain more than one word.

$@ This symbol causes the ruleset to return with a value that is the remainder of the right side of this rule.

$: This symbol terminates the rule, but allows the ruleset to continue. Use this symbol to avoid continued application of a rule. For example, the following rule matches anything, passes that input to ruleset 7 and continues:

```
R$+             $:$>7$1
```

The **$:** prevents the rule from being evaluated again to avoid an infinite loop.

When **sendmail** evaluates the right side of a rule it uses the following order:

1. Substitutes parameters from the left side
2. Resolves host names
3. Calls subroutines
4. Processes the symbols **$#**, **$@** and **$:**.

¦ Copyright IBM Corp. 1985, 1991

6.6.10.9.4 Naming a Set of Rewrite Rules

The configuration file contains several sets of rewrite rules. The **sendmail** program uses some of the sets internally; the format of these sets cannot change. Other sets whose format can be changed can be referenced by mailer definitions and other sets of rewrite rules. Each set of rules is delimited by a control line that contains the letter **S** followed by a number:

```
Sn
```

This control line means that all rewrite rules that follow this control line, until another **S** control line occurs, are part of rule set number **n**. The variable **n** can be any integer. For example, the following statement marks the beginning of ruleset 10:

```
S10
```

If you start the same ruleset more than once, only the rules contained in the last ruleset are kept. In the following generalized example only rules x, y, and z are part of the final ruleset 11:

```
S11
  rule a
  rule b
  rule c
S11
  rule x
  rule y
  rule z
```

¦ Copyright IBM Corp. 1985, 1991

6.6.10.9.5 Testing the Rewrite Rules

When you build a configuration file, you can test the rewrite rules using the test mode of **sendmail**. To use the test mode on a new configuration file, first build the data base version of the configuration file and then use the test mode of the **sendmail** command:

```
/usr/lib/sendmail -bz -Cnewfile
/usr/lib/sendmail -bt -Cnewfile
```

In this format, **newfile** is the name of the new configuration file to test. For example, to test a new configuration file named **test.cf**, use the commands:

```
/usr/lib/sendmail -bz -Ctest.cf
/usr/lib/sendmail -bt -Ctest.cf
```

This command reads the configuration file **test.cf** and enters test mode. In test mode, you enter lines of the form:

```
ruleset address
```

where **ruleset** is the number of the rewrite ruleset to use and **address** is an input address. Test mode shows you the steps it takes and the final address. To test more than one ruleset, use a list of ruleset numbers with commas between the ruleset numbers. The rules will be applied to the input address in the order that they appear in the list.

For example, the following entry first applies ruleset 3 to the input **zeus:geo**. Ruleset 1 is then applied to the output of ruleset 3, followed similarly by rulesets 21 and 4.

```
3,1,21,4 zeus:geo
```

Using the standard configuration file **/usr/adm/sendmail/sendmail.cf**, the following sequence occurs:

```
#/usr/lib/sendmail -bt -C/usr/adm/sendmail/sendmail.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
> 3,1,21,4 zeus:geo
rewrite:  ruleset  3   input:  "zeus" ":" "geo"
rewrite:  ruleset  6   input:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset  6 returns:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset  3 returns:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset  1   input:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset  1 returns:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset 21   input:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset 21 returns:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset  4   input:  "geo" "<" "@" "zeus" ">"
rewrite:  ruleset  4 returns:  "geo" "@" "zeus"
>
```

Use the End of File character (**Ctrl-D**) to exit the program.

For more detail, use the **d21** flag to turn on more debugging (you may find level **d21.12** useful). For example, the following command turns on a large amount of information; a single word address is probably going to print out several pages of information.

¦ Copyright IBM Corp. 1985, 1991

```
sendmail -bt -d21.99
```

For example, when using the previous address as an example, a portion of the ruleset 4 information that pertains to the rule:

```
R$*<$+>$*               $1$2$3
```

appears as follows:

```
    .
    .
    .
rewrite:  ruleset  4   input:  "geo" "<" "@" "zeus" ">"
    .
    .
    .
-----trying rule:  "^P" "<" "^Q" ">" "^P"
ap="geo", rp="^P"
ap="geo", rp="<"
ap="<", rp="<"
ap="@", rp="^Q"
ap="zeus", rp=">"
ap=">", rp=">"
ap=<null>, rp="^P"
-----rule matches:  "^U1" "^U2" "^U3"
$1:  3fffe05b="geo"
$2:  20011397="@" 3fffe054="zeus"
$3:
rewritten as:  "geo" "@" "zeus"
    .
    .
    .
```

¦ Copyright IBM Corp. 1985, 1991

6.7 Chapter 7. Managing Asynchronous Terminal Emulation (ATE)

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.7.1 Contents

¦ Copyright IBM Corp. 1985, 1991

6.7.2 About This Chapter

This section describes how to customize and configure the Asynchronous Terminal Emulation (ATE) program to fit your particular needs. Topics covered include modifying local settings, altering connection settings, remapping control keys, and changing the default file. This chapter also discusses two file transfer protocols: xmodem and pacing.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System*, and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

6.7.3 Before You Begin

Unless a connection has been made to a remote terminal, you must star the ATE program.

The commands in this chapter can be given from either the Unconnecte or the Connected Main Menu.

Only ASCII characters can be used to communicate with the ATE progra (options, etc.), but any characters can be used while in the emulation mode.

¦ Copyright IBM Corp. 1985, 1991

6.7.4 Modifying Local Settings (modify)

The **modify** command lets you change several features of your local system. You can:

Change the name of the capture file that receives incoming data

Switch or toggle the following features **on** or **off**.

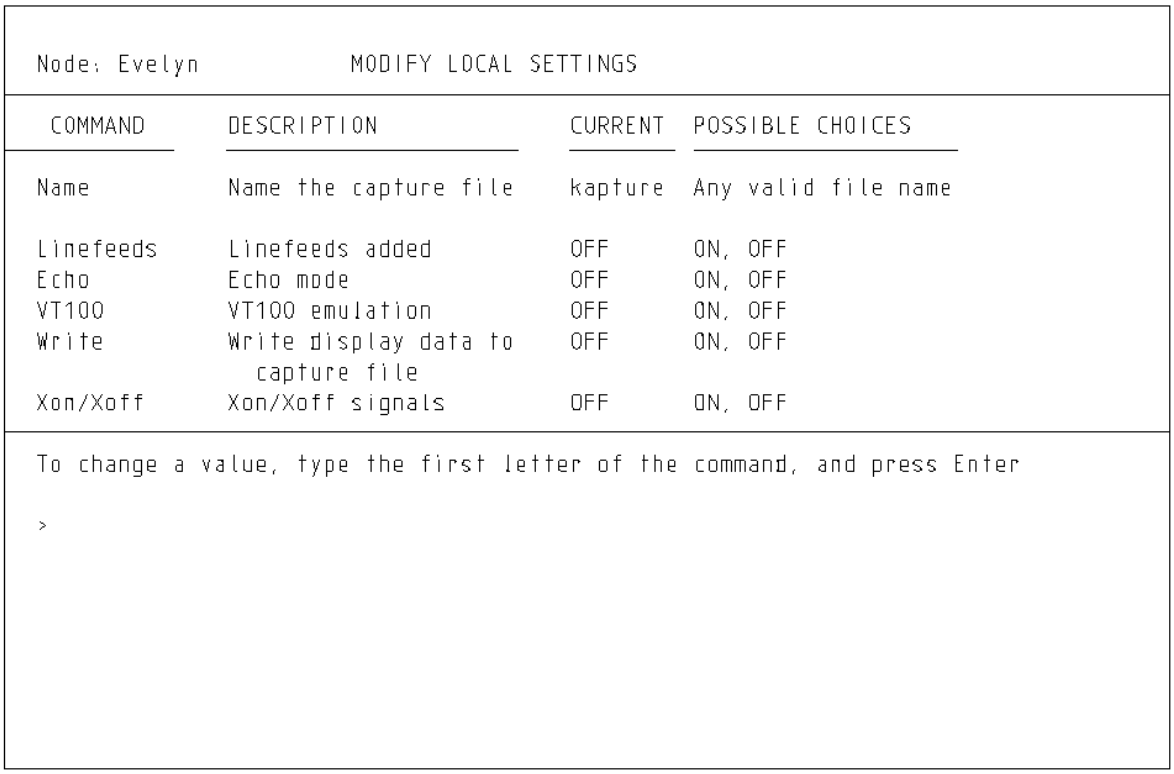
- Add a linefeed character at the end of each line of incoming data.
- Set echo mode.
- Emulate a DEC VT100 terminal at the console.
- Write incoming data to a capture file as well as to the display.
- Use an Xon/Xoff (transmitter on/off) signal.

Note: The locale of the remote environment should be the same as the locale environment. If the locales do not match, unexpected results may occur.

If you want to see the Modify Menu to view the features that can be changed and see the current default settings, type the letter **m** after the command prompt on either main menu.

The Modify Menu with the default values that are in effect each time you start ATE looks like this:

¦ Copyright IBM Corp. 1985, 1991



```
Node: Evelyn          MODIFY LOCAL SETTINGS

 COMMAND      DESCRIPTION              CURRENT  POSSIBLE CHOICES

Name          Name the capture file    kapture  Any valid file name

Linefeeds     Linefeeds added          OFF      ON, OFF
Echo          Echo mode                OFF      ON, OFF
VT100         VT100 emulation          OFF      ON, OFF
Write         Write display data to    OFF      ON, OFF
                capture file
Xon/Xoff      Xon/Xoff signals         OFF      ON, OFF

To change a value, type the first letter of the command, and press Enter

>
```

The features that can be changed are in the first or Command column. For a description of each feature, see "Description of Features" in topic 6.7.4.2. Remember that:

You can change as many features as you want at the same time

If you change the **name** variable, you also must supply a value, the new name.

All other variables are switches that you can turn **on** or **off** by typing the command.

When you type the command, you switch or toggle the value.

You can invoke the **modify** command from either the Unconnected Main Menu or the Connected Main Menu. If you follow the steps in the following instruction box, you can bypass the Modify Menu.

Modifying Local Settings from a Main Menu

1. On the Main Menu, type **m commandinitial** [**value**] (**commandinitial**) after the command prompt.

   For example, to toggle the settings of both the **linefeed** and **echo** features, type:

   ```
   >m l e
   ```

   To change the **name** variable to **schedule** in addition to toggling the **linefeed** and **echo** settings, type:

¦ Copyright IBM Corp. 1985, 1991

```
>m n schedule l e
```

2. Press the **Enter** key.

   Any data you save now is put into the **schedule** file, and the **linefeed** and **echo** commands are switched to the alternate setting.

Subtopics
6.7.4.1 Additional Information
6.7.4.2 Description of Features

¦ Copyright IBM Corp. 1985, 1991

6.7.4.1 Additional Information

The **( )** (parentheses) indicate that you may repeat the contents as many times as you want.

The [ ] (brackets) indicate that a value is optional. A value applie only to the **name** variable, since all the other variables are **on/off** switches.

If you give the **modify** command without typing a variable initial, the Modify Menu displays. You then must type the initial of each variable you want to change after the command prompt.

If you use a value with any feature other than **name**, an error message appears:

```
062-003  The 'command-name' command is not valid.
         Enter the first letter of a command
         from the list on the menu.
```

If you see this message, either you have typed an incorrect letter or included an invalid value.

¦ Copyright IBM Corp. 1985, 1991

6.7.4.2 Description of Features

name File for incoming data when the **write** command is **on** or when the **Ctrl-B** control key is used during a connection.

Options: Any valid file name that is 40 characters or less. The first 18 characters are displayed in the Modify Menu.

Default: **kapture**.

linefeeds Adds a linefeed character after every carriage return in the incoming data stream.

Options: On, Off.

Default: Off.

echo Displays your typed input.

With a remote computer that supports echoing, each character you send returns and appears on your display. When **echo** is **on**, each character displays twice, as you type it and when it returns. When **echo** is **off**, each character displays only once, when returned from the remote computer.

Options: On, Off.

Default: Off.

VT100 The local console emulates a DEC VT100 terminal so you can use DEC VT100 codes with the remote system. With **VT100 off**, the local console functions like a PS/2 or PS/55.

Options: On, Off.

Default: Off.

Note: Use this command only if you have the keyboard that came with your system, since some keys are remapped. Any other keyboard gives you unpredictable results. Some DEC VT100 codes, like 132 columns, double-height and double-width lines, and origin mode are not supported.

write Routes incoming data to the file specified in the **name** command, as well as to the display. This functions like the **Ctrl-B** control key during a connection. Carriage return/linefeed combinations are converted to linefeeds before being written to the capture file. In an existing file, data is appended to the bottom.

Options: On, Off.

Default: Off

Xon/Xoff Controls data transmission at a port as follows:

When an Xoff is received, transmission stops.

When an Xon is received, transmission resumes.

¦ Copyright IBM Corp. 1985, 1991

An Xoff is sent when the receive buffer is nearly full.

An Xon is sent when the buffer is no longer full.

Options: On, Off.

Default: Off.

¦ Copyright IBM Corp. 1985, 1991

6.7.5 Altering Connection Settings - The alter Command

The **alter** command lets you change several data transmission characteristics, including:

Bit length and rat

Stop and parity bit

Port nam

Modem dialing prefixes and suffixe

Waiting time and retry limit

File transfer metho

Pacing character or delay time

If you want to see the Alter Menu to view the characteristics that can be changed and the current default settings, type the letter **a** after the command prompt on either main menu.

The Alter Menu with the original default values in effect each time you start ATE looks like this:

```
Node: Evelyn            ALTER CONNECTION SETTINGS

 COMMAND      DESCRIPTION              CURRENT  POSSIBLE CHOICES
 Length       Bits per character       8        7,8
 Stop         Number of stop bits      1        1,2
 Parity       Parity setting           0        0=none, 1=odd, 2=even
 Rate         Number of bits/second    1200     50,75,110,134,150,300,600
                                                 1200,1800,2400,4800,9600,19200
 Device       /dev name of port        tty0     tty0-tty16
 Initial      Modem dialing prefix     ATDT     ATDT, ATDP, etc.
 Final        Modem dialing suffix              0 for none, valid modem suffix
 Wait         Wait between redialing   0        seconds between tries
 Attempts     Maximum redial tries     0        0 for none, a positive integer
 Transfer     File transfer method     p        p=pacing, x=xmodem
 Character    Pacing char or number    0        0 for none, a single char/integer

 To change a current choice, type the first letter of the command
 followed by your new choice (example: r 300) and press Enter.
 >
```

The characteristics that can be changed are in the first or Command column. For a description of each characteristic, see "Description of Characteristics" in topic 6.7.5.2 Remember that:

You can change as many features as you want at a time

¦ Copyright IBM Corp. 1985, 1991

You must type a new value for each characteristic you change. Fo example, if you want to change the number of bits per second (the rate) to 9600, type:

```
>r 9600
```

You can invoke the **alter** command from either the Unconnected Main Menu or the Connected Main Menu. If you follow the steps in the instruction box below, you can bypass the Alter Menu.

Altering Connection Settings from a Main Menu

1. On the Main Menu, type **a commandinitial value** (**commandinitial value**) after the command prompt. For example, to change the **rate**, **wait**, and **attempt** values, type:

   ```
   >a r 9600 w 5 a 1
   ```

2. Press the **Enter** key.

Subtopics
6.7.5.1 Additional Information
6.7.5.2 Description of Characteristics

¦ Copyright IBM Corp. 1985, 1991

6.7.5.1 Additional Information

The **( )** (parentheses) indicate that you may add as many variables as you want.

Type the first letter of the variable that you want to change followed by the value you select.

Press **Enter** again to leave the Alter Menu.

¦ Copyright IBM Corp. 1985, 1991

6.7.5.2 Description of Characteristics

length Number of bits in a data character. The length must match the length expected by the remote system.

Options: 7 or 8.

Default: 8.

stop Number of stop bits appended to a character to signal the end of that character during data transmission. This number must match the number of stop bits used by the remote system.

Options: 1 or 2.

Default: 1.

parity Checks whether a character was successfully transmitted to or from a remote system. Must match the parity of the remote system.

For example, if you select **even** parity, when the number of 1 bits in the character is odd, the parity bit is turned on to make an even number of 1 bits.

Options: 0=none, 1=odd, 2=even.

Default: 0.

rate Bit rate (bits per second). Determines the number of bits transmitted per second. The speed must match the speed of your modem and that of the remote system.

Options: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19200.

Default: 1200.

device Name of the asynchronous port used to make a connection to a remote system.

Options: tty0 - tty16 or locally created port names. Only the first 8 characters appear in the Alter Menu.

Default: tty0.

initial Dial command that must precede the telephone number when you autodial with a modem. Consult your modem user's guide for the proper dial commands.

Options: ATDT, ATDP, etc. Only the first 8 characters appear in the Alter Menu.

Default: ATDT.

final Dial command that must follow the telephone number when you autodial with a modem. Consult your modem user's guide for the proper dial command.

Options: 0=none, valid modem suffix. Only the first 8

¦ Copyright IBM Corp. 1985, 1991

characters appear in the Alter Menu.

Default: None.

wait Tells the **redial** command in the ATE program to wait **n** seconds between redialing attempts. The wait period does not begin until the connection attempt times out or until you interrupt it. If **attempts** is set to 0, no redial attempt occurs.

Options: 0=none, a positive integer.

Default: 0.

attempts Maximum number of times ATE tries to redial to make a connection. If **attempts** is set to 0, no redial attempt occurs.

Options: 0=none, a positive integer.

Default: 0.

transfer Type of asynchronous protocol that transfers files during a connection.

Pacing File transfer protocol that controls the data transmission rate by waiting for a specified character or a certain number of seconds between line transmissions. This helps prevent loss of data when the transmission blocks are too large or are sent too quickly for the system to process.

For more information, see "Using Pacing Protocol for File Transfer" in topic 6.7.9.

Xmodem An 8-bit file transfer protocol that detects data transmission errors and retransmits the data. For more information, see "Using Xmodem Protocol for File Transfer (xmodem)" in topic 6.7.8.

Options: p=pacing, x=xmodem.

Default: p.

character Specifies the type of pacing to be used.

Character Signal to transmit a line. (Select one character).

Integer Number of seconds the system waits between each line it transmits. (Select one integer.) The default value is 0, indicating a pacing delay of 0 seconds.

Default: 0.

¦ Copyright IBM Corp. 1985, 1991

6.7.6 Changing (Remapping) the Control Keys

The mapping for these key combinations, set in the default file **ate.def**, follows:

capture key Starts or stops capturing (saving) the data displayed on your screen during an active connection. This key has a switching or toggle effect.

Options: Any ASCII control character.

Default: ASCII octal 002 (STX). This is the **Ctrl-B** key combination on the keyboard.

main-menu key Returns the Connected Main Menu so you can issue a command during an active connection. This control key functions only from the connected state.

Options: Any ASCII control character.

Default: ASCII octal 026 (SYN). This is the **Ctrl-V** key combination on the keyboard.

previous key Displays the previous screen any time during the program. The screen that displays varies, depending on the screen in use when you press the previous-screen key.

Options: Any ASCII control character.

Default: ASCII octal 022 (DC2). The ASCII control character is mapped to the AIX interrupt signal. This is the **Ctrl-R** key combination on the keyboard.

Changing or remapping the control keys may be necessary if you have two applications in which control keys conflict. For example, if the control keys mapped for the ATE program conflict with those in your text editor, you may want to remap the control keys for ATE.

A system user with some programming experience can remap the control keys by editing the default file **ate.def**. See "Changing Your Default File" in topic 6.7.7.

Subtopics
6.7.6.1 ASCII Control Characters

¦ Copyright IBM Corp. 1985, 1991

6.7.6.1 ASCII Control Characters

The ASCII control character you select may be in octal, decimal, or hexadecimal format.

octal 000 through 037

The leading zero is required.

decimal 0 through 31

hexadecimal 0x00 through 0x1F

The leading 0x is required. The x may be uppercase or lowercase.

¦ Copyright IBM Corp. 1985, 1991

6.7.7 Changing Your Default File

The first time you run the ATE program, it creates a file named **ate.def** in the current directory. This file contains default values for:

Data transmission characteristics

Local system features

Dialing directory file

Control keys

The default file that is created the first time ATE runs looks like this:

```
LENGTH          8
STOP            1
PARITY          0
RATE            1200
DEVICE          tty0
INITIAL         ATDT
FINAL
WAIT            0
ATTEMPTS        0
TRANSFER        p
CHARACTER       0
NAME            kapture
LINEFEEDS       0
ECHO            0
VT100           0
WRITE           0
XON/XOFF        0
DIRECTORY       /usr/lib/dir
CAPTURE_KEY     002
MAINMENU_KEY    026
PREVIOUS_KEY    022
```

A system user with some data processing experience can edit the **ate.def** file with a text editor to change the values of these characteristics.

How to Edit the Default File

1. Leave ATE.

2. Enter the text editor and display the **ate.def** file.

3. Delete all variables which you **do not** want to change.

   This simplifies reading the file, since the system ignores variables that remain the same as the system's defaults.

4. Change the values for those variables you want to change.

¦ Copyright IBM Corp. 1985, 1991

5. Save the file.

We now give an example. Assume that you want to change:

RATE to **300** bps.
DEVICE to **tty3**
DIRECTORY to **my.dir**
TRANSFER to **x** (xmodem)

The **ate.def** file that you create looks like the following:

```
RATE            300
DEVICE          tty3
TRANSFER        x
DIRECTORY       my.dir
```

From now on when you start using ATE, the program looks for a file in the current directory named **ate.def**. Then it reads the values in the file and changes the system's defaults to those you set.

Subtopics
6.7.7.1 Additional Information

Copyright IBM Corp. 1985, 1991

6.7.7.1 Additional Information

Variable names must be in uppercase and spelled exactly as they appea in the original default file.

Type only one variable per line

If you enter an incorrect value, you receive a system message but th program continues, using the default value.

¦ Copyright IBM Corp. 1985, 1991

6.7.8 Using Xmodem Protocol for File Transfer (xmodem)

Xmodem protocol is an 8-bit transfer protocol that can detect data transmission errors and then retransmit the data. The workstation that sends data must wait until the remote system sends a signal that it is ready to receive data.

When the receiver gets data, it returns an acknowledgement to the sender. The ATE receiver times out if it does not receive data 90 seconds after file transfer is initiated.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.7.8.1 Xmodem Shell Command

A shell command **xmodem**, shipped with ATE, starts an xmodem file transfer protocol on a remote computer. This command is used in combination with the **send** or **receive** command on the Connected Main Menu.

Sending and receiving with **xmodem** are complementary operations. One system must be set to send while the other is set to receive.

You can interrupt the **xmodem** shell command by pressing the **Ctrl-X** key combination.

Subtopics
6.7.8.1.1 Format

¦ Copyright IBM Corp. 1985, 1991

6.7.8.1.1 Format

xmodem -opt filename

xmodem Transfers files between computer systems that have ATE installed.

-opt The option you use with the **xmodem** command:

-s Send data to the local computer.

-r Receive data from the local computer.

filename The name of the file you send or receive.

¦ Copyright IBM Corp. 1985, 1991

6.7.8.2 Sending a File

After a connection to a remote computer is established, you can send a file from the local system to the remote computer with the following procedure. In this procedure, you first request the remote system to receive your file. Then, you execute the sending of your file by selecting the send option from the menu.

Sending a File with Xmodem Protocol

1. Type **xmodem -r filename**.

   For example, to send **mayfile** to a remote computer, type:

   ```
   xmodem -r mayfile
   ```

2. Press **Ctrl-V**.

   The Connected Main Menu displays.

3. Select the **send** command.

   ```
   >s mayfile
   ```

After the file is transferred, your terminal displays the connection screen.

¦ Copyright IBM Corp. 1985, 1991

6.7.8.3 Receiving a File

After a connection to a remote computer is established, you can receive a file from the remote computer with the following procedure. In this procedure, you first request the remote system to send the file to your location. Then, you execute the receiving of your file by selecting the receive option from the menu.

Receiving a File with Xmodem Protocol

1. Type **xmodem -s filename**.

   For example, to receive the **infile** from the remote computer, type:

   ```
   xmodem -s infile
   ```

2. Press **Ctrl-V**.

   The Connected Main Menu displays.

3. Select the **receive** command.

   ```
   >r infile
   ```

After the file is transferred, your terminal displays the connection screen.

¦ Copyright IBM Corp. 1985, 1991

6.7.9 Using Pacing Protocol for File Transfer

Pacing is a file transfer protocol required by some systems. It controls data transmission by waiting for a specified character or waiting a specified number of seconds between lines. This protocol prevents the loss of data when the block size is too large or data is sent too quickly for the system to process.

Use this protocol to send text files only. Files containing embedded control characters cannot be transferred.

Asynchronous Terminal Emulation supports two types of pacing control protocols: character pacing and interval pacing.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.7.9.1 Character Pacing

The **send** protocol transmits data from the file until it finds a linefeed. It then sends out a carriage return and waits to receive the pacing character before sending more data. It times out if it does not receive the correct pacing character in 30 seconds.

The **receive** routine sends a pacing character as soon as it is started. It times out if it does not receive data within 30 seconds.

When data transmission starts, the **receive** routine sends a pacing character to the sender whenever it finds a carriage return in the data. The program ends when it receives nothing for 30 seconds.

¦ Copyright IBM Corp. 1985, 1991

6.7.9.2 Interval Pacing

The **send** routine begins to transfer data as soon as it is started. When it finds a linefeed, it converts it to a carriage return and waits a specified time before it sends the next line.

The **receive** routine looks for data as soon as it is started and times out if it does not receive data within 30 seconds.

¦ Copyright IBM Corp. 1985, 1991

6.7.9.3 Additional Information

Unix files traditionally contain only linefeeds (called newlines) to delimit the end of a line. DOS or CP/M files usually use only carriage returns. For this reason, the pacing **send** protocol converts linefeeds to carriage returns. The pacing **receive** protocol converts any linefeed/carriage return combinations or single carriage returns to linefeeds before saving the file.

¦ Copyright IBM Corp. 1985, 1991

6.8 Chapter 8. Managing the Basic Networking Utilities

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.8.1 Contents

¦ Copyright IBM Corp. 1985, 1991

6.8.2 About This Chapter

The Basic Networking Utilities (BNU) program is part of the Extended Services facility of the AIX operating system. Composed of directories, files, user commands, and administrative programs and daemons, BNU enables users to communicate with computers other than their local computers. This chapter provides the information you need to manage the BNU software.

The chapter begins with a brief overview of the hardware required to use the BNU facility and then continues with a more extensive description of the BNU software. It lists and briefly describes the directories, data base files, administrative files, user commands, administrator commands, and administrative daemons that make up the networking utilities.

The system administrator responsible for the BNU facility must perform the following initial administrative tasks:

Installing the softwar

Setting up the remote communications facilities

Note: If your site uses the Transmission Control Protocol/Internet Protocol (TCP/IP), you will need to perform some additional steps in order for BNU and TCP/IP to communicate.

This chapter also explains the routine tasks involved in maintaining the software. These include the following:

Monitoring file transfer

Cleaning up the spooling directorie

Working with log files

The BNU software includes four daemons that handle remote communications activities such as the following:

Transporting copy request

Scheduling work in the spooling directorie

Executing remote command

Communicating with TCP/IP

This chapter also includes information about running automatic maintenance routines.

The final sections of this chapter describe how to handle some common problems encountered when using the BNU software.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System,* and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

6.8.3 Overview of the BNU Hardware

Before a local computer can communicate with a remote computer, a two-way communication link must be established between the systems. There are three ways to set up such a link:

Using a hardwired line with a device such as tt

Using a telephone line with a mode

Using a TCP/IP connection over a Token Ring or Ethernet network

The hardwired connection links a port on the local computer to a port on the remote computer. A hardwired line is advantageous when users on local systems communicate frequently with remote systems; the link is always available and access time is short. However, a port used for a hardwired communications link is not available for any other purpose.

A hardwired connection is made over an RS-232C serial port at transmission rates of up to 19,200 bits per second. The recommended length of such direct links is 50 feet or less because noise becomes a problem with greater distances. It is possible to obtain longer lengths by using a lower transmission rate and/or limited distance modems (short-haul modems) at both ends of the link.

The second type of connection uses a telephone line and a modem. In this case, the user on the local computer establishes the connection to a remote computer through an Automatic Calling Unit (ACU), also referred to as an autodialer or a modem. The modem attached to the remote system answers the telephone, and the communications software then completes the connection.

The advantage of a modem connection using a phone line is that the local and remote ports are not dedicated to a single computer. The disadvantage is that the port of the remote system may be busy. A dial-up link also requires additional software and hardware, such as the ACU, that is not necessary with a hardwired connection.

The third type of connection uses TCP/IP. This connection requires an optional adapter for the computer. Depending on the type of network used at your site, either a Token Ring adapter or a Baseband (Ethernet) adapter is required. "Setting Up BNU for Use with TCP/IP" in topic 6.8.5.4 contains information about using TCP/IP with the BNU facility.

For additional information about the hardware used in BNU communication, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

¦ Copyright IBM Corp. 1985, 1991

6.8.4 Overview of the BNU Software

The Basic Networking Utilities software is composed of the following items:

Directories in which the various files and programs are store

Files in the supporting data base containing information needed t establish remote connections and determine access permissions

Administrative files used primarily in transferring data betwee computers

User commands that perform the basic BNU function

Administrator commands comprised of programs that enable you t configure and maintain BNU

Daemon programs that handle file transfers, communication with TCP/IP scheduling work, and remote command executions.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.8.4.1 Basic Directories

There are several basic directories that contain the BNU programs and support files. Some of these directories are unique to BNU, while others are common to the AIX operating system. The descriptions that follow are intended only as brief overviews. The remaining sections of this chapter describe the files of these directories in more detail.

The /etc/locks Directory

This directory contains the lock (LCK.*) files for the devices used by BNU in establishing remote connections. Lock files prevent several users from accessing a specified device at the same time. The **/etc/locks** is a symbolic link to **<LOCAL>/locks**.

The /usr/adm/uucp Directory

This directory contains the files comprising the BNU supporting data base, several versions of the command **uutry**, and four scripts for automatic routine maintenance operations.

The files in the supporting data base include the following:

Devices
Dialcodes
Dialers
Maxuuscheds
Maxuuxqts
Permissions
Poll
remote.unknown
Systems
Myname
Spools

The routine maintenance scripts include the following:

uudemon.admin
uudemon.cleanu
uudemon.hour
uudemon.poll

The /usr/bin Directory

This directory includes many of the AIX operating system executable programs. It also contains the BNU executable files for the following commands:

ct
cu
uucp
uulog
uux
uupick
uustat
uuto
uuname

The /usr/lib/uucp Directory

This directory contains two administrator commands:

uucheck
uucleanup

It also includes the following BNU ***daemons***, programs that handle file transfers, communications with TCP/IP, scheduling work requests, and remote command executions:

uucico
/etc/uucpd
uusched
uuxqt

¦ Copyright IBM Corp. 1985, 1991

The /usr/spool/cron/crontabs Directory

This is a standard AIX directory that contains all users' **crontabs** (from **cron** tables) files. The **cron** daemon automatically runs commands at specified dates and times according to the instructions included in these **crontabs** files.

One of the files in this directory is named **uucp**. It contains instructions to run designated BNU commands at specified times in order to perform certain administrative tasks automatically. These tasks include processes such as handling file transfers and cleaning up the files in the spooling directories.

The changes made to **/usr/spool/cron/crontabs/uucp** will not take effect unless the changed file is installed or the system is restarted. Therefore, use the **crontab** command to get a copy of the file, edit the copy, and then use the **crontab** command to install the changed file.

See the *AIX Operating System Commands Reference* for more information about changing the **crontab** file.

The /usr/spool/uucp/system_name Directory

This is a spooling directory on the local system. It contains queued requests issued by local users for file transfers to remote systems and for command executions on remote systems.

The /usr/spool/uucppublic Directory

This is the public directory for the BNU facility, and one of these directories exists on every system connected by the networking utilities. When a user transfers a file to another system, or issues a request to execute a command on another system, the files generated by these BNU commands are stored in the public directory on the designated system. However, the user can specify a destination other than the public directory when issuing the **uucp**, **uuto**, or **uux** commands.

The directory **/usr/spool** is a symbolic link to **<LOCAL>/spool**.

For security reasons, you may prefer to set up restrictions that limit the destination of transferred files to the public directory.

¦ Copyright IBM Corp. 1985, 1991

6.8.4.2 Files in the Supporting Data Base

The files in the supporting data base contain information that BNU requires to perform the following tasks:

Establish connections to remote computer

Determine and modify access permissions

In terms of establishing links to remote computers, a process discussed in "Setting Up Remote Communication" in topic 6.8.5.2, the most important of these files are the **Devices**, **Permissions**, and **Systems** files.

The Devices File

The file **/usr/adm/uucp/Devices** contains information about the devices on the local system that can establish a connection to a remote computer. This file includes information for hardwired, telephone, and TCP/IP communication links.

For detailed information about customizing the **Devices** file for your site, see "Setting Up Devices" in topic 6.8.5.2.1.

The Dialcodes File

The file **/usr/adm/uucp/Dialcodes** contains the initial digits of telephone numbers used to establish communications over a phone line. These numbers, such as the code for an outside line, the area code, and the three-digit exchange, are called ***dialing code abbreviations***. The **Systems** file, discussed below, uses these dialing code abbreviations as part of the telephone entry when attempting to contact the modem on a remote computer.

For information about customizing the **Dialcodes** file for your site, see "Setting Up Dialing Codes" in topic 6.8.5.2.4.

The Dialers File

The file **/usr/adm/uucp/Dialers** includes a line of characters containing information about the way in which every modem connected to the local system acts when contacting a remote system. Each line in this file specifies the "initial handshaking" that determines whether the communications link between the two systems is set up correctly and ready to transfer data.

For information about customizing the **Dialers** file for use at your site, see "Setting Up Dialers" in topic 6.8.5.2.2.

The Maxuuscheds File

The file **/usr/adm/uucp/Maxuuscheds** limits the number of remote systems that the **uucico** program (which handles file transfers) can contact at any one time. This file is used in conjunction with the **uusched** daemon and the lock files in the **/etc/locks** directory to determine the number of systems currently being polled.

You do not need to configure the **Maxuuscheds** file. You simply use it to help manage your system resources and load averages. This file generally requires little or no maintenance unless the system on which it is installed is utilized frequently and heavily by users on remote systems.

¦ Copyright IBM Corp. 1985, 1991

For additional information about the **Maxuuscheds** file, see "Limiting the Number of Scheduled Jobs" in topic 6.8.7.3.2.

The Maxuuxqts File

The file **/usr/adm/uucp/Maxuuxqts** limits the number of **uuxqt** processes running simultaneously on a local system. (The **uuxqt** daemon executes commands on a local system that have been issued by users on a remote system.)

You do not need to configure the **Maxuuxqts** file, which generally requires little or no maintenance unless the system on which it is installed is utilized frequently and heavily by users on remote systems.

For additional information about the **Maxuuxqts** file, see "Limiting the Number of Remote Executions" in topic 6.8.7.2.3.

The Permissions File

The file **/usr/adm/uucp/Permissions** contains information that specifies the remote systems that are allowed to communicate with the local system. This file has options that enable the system manager to specify the remote computers that may log in to the local system, whether these systems may send and receive files, and the commands that users on the remote computer may run on the local system.

For detailed information about setting up the **Permissions** file to meet the special needs of your group of users, see "Customizing the Permissions File" in topic 6.8.5.2.5.

The Poll File

The file **/usr/adm/uucp/Poll** contains information specifying when BNU should poll designated remote systems. This file is used in conjunction with the **/usr/spool/cron/crontabs/uucp** file, the **uudemon.hour** script, and the **uudemon.poll** script. Together, these files are responsible for initiating automatic calls to remote systems to perform certain maintenance tasks.

For information about using the **Poll** file, see "Customizing the Poll File" in topic 6.8.5.2.6 and "Polling Remote Systems (uudemon.poll)" in topic 6.8.8.4.

The remote.unknown File

The file **/usr/adm/uucp/remote.unknown** handles requests for connections from remote computers that are not listed in the **Systems** file.

For information about using the **remote.unknown** file, see "Customizing the remote.unknown File" in topic 6.8.5.2.7.

The Systems File

The file **/usr/adm/uucp/Systems** contains a list of the remote computers with which users on the local system can communicate. Each entry in the file represents a remote system, and users on the local system cannot communicate with a remote system unless that system is listed in the local **Systems** file.

¦ Copyright IBM Corp. 1985, 1991

The names used for remote systems must contain only single-byte ASCII characters.

For information about setting up the **Systems** file for use at your site, see "Customizing the Systems File" in topic 6.8.5.2.3.

The Myname File

The file **/usr/adm/uucp/Myname** contains the global name by which the TCF cluster is known by other remote systems. This name, along with the IP address of the site which runs the cluster's **uucp** daemon, is used to establish TCP/IP connection with remote systems. Edit the **Myname** file to contain your local cluster name.

The Spools File

The file **/usr/adm/uucp/Spools** contains a list of master spool site for remote TCF systems. The format of the file is:

```
remote master local local ...
```

Where remote is the name of the remote uucp node, master is the master spool for that remote, and local is the name(s) of computers on the local TCF cluster that can send files to that remote system.

The system name of the cluster from this file, and the system name of remote systems, must contain only single-byte characters from the set of ASCII characters.

¦ Copyright IBM Corp. 1985, 1991

6.8.4.3 Administrative Files Used to Transport Data

The primary purpose of the Basic Networking Utilities facility is to send and receive data. BNU uses six types of files to accomplish this task:

command (or work) files	lock files
data files	machine log files
execute files	temporary data files

Command/work (C.*) Files

When a user requests a file transfer or a remote command execution, the **uucp** and **uux** commands create **C.*** files in the spooling directory, **/usr/spool/uucp/system_name**, on the local system. (The **system_name** entry in the path name represents the computer to or from which the files are transferred, or the computer where the command is to be executed.)

Command files are often referred to as work files because they contain information necessary to perform the requested work. Essentially, they provide the **uucico** daemon (for **uucp**) and the **uuxqt** daemon (for **uux**) with the data required to transmit the source file or to execute the AIX command on the specified computer.

Command files contain the following types of information:

An S (send) or R (receive) notatio

The full path name of the source fil

The full path name of the destination fil

The sender's login nam

A list of the command option

The name of the associated data fil

The source file's permissions code (mode bits

The name of the recipient on the remote system

For more information about these files, see "Additional Information about Command Files" in topic 6.8.7.1.4.

Data (D.*) Files

When a user issues the **uucp** command with the **-C** flag to send a file to the spooling directory for transfer, **uucp** creates a data file to contain the actual source file. The **uux** command also creates a data file, which either contains the data for a remote command execution, or becomes an execute file on a remote system for a remote command execution.

For more information about these files, see "Additional Information about Data Files" in topic 6.8.7.1.5.

Execute (X.*) Files

Executable files created by **uux** contain the command string that the user wants to run on the remote system. These files contain the following types of information:

¦ Copyright IBM Corp. 1985, 1991

the user line, which includes the user's login and the name of th local system

an error status return lin

the name of the user requesting the command executio

the name(s) of any file(s) required to execute the comman

the name of the system and file designated to receive standard outpu from the command execution

the command string itself

For more information about these files, see "Additional Information about Execute Files" in topic 6.8.7.2.4.

Lock (LCK.*) Files

Every time a user attempts to contact a remote system using a device specified in the **Devices** file, BNU creates a lock file in the directory **/etc/locks**. This temporary file "locks" the device so that another user cannot access it while it is in use.

For more information about these files, see "Additional Information about Lock Files" in topic 6.8.5.2.1.

Machine Log Files

BNU creates a log file on the local system, in the appropriate machine-specific subdirectory under **/usr/spool/uucp/.Log**, for each remote system with which it communicates. At specified intervals, the **uucleanup** command combines these log files and stores them in a directory named **/usr/spool/uucp/.Old**. The combined files remain in this directory for a specified time period, after which they are removed.

For more information about these files, see "Working with Log Files" in topic 6.8.6.3.

Temporary (TM.*) Data Files

BNU transfers a data file from the spooling directory on a local system to the public spooling directory on a remote system, placing the original data file in a temporary data file for further action. If there are no transfer problems, BNU renames the temporary file with the appropriate **D.*** name and sends it on to the specified destination. If the processing terminates abnormally, the temporary file remains in the spooling directory until you remove it either manually, or with an automatic cleanup procedure.

For more information about these files, see "Additional Information about Data Files" in topic 6.8.7.1.5.

¦ Copyright IBM Corp. 1985, 1991

6.8.4.4 User Commands

BNU has a number of commands intended primarily for users, although system administrators will find that several of these commands are useful in managing the networking facility.

Subtopics
6.8.4.4.1 BNU Commands

¦ Copyright IBM Corp. 1985, 1991

6.8.4.4.1 BNU Commands

The Basic Networking Utilities (BNU) commands (**cu** and **uucp**) provide a transparent data path between AIX systems. **cu** provides a data stream for terminal login that passes extended characters accurately, and **uucp** transfers files without distorting extended character contents. (BNU connections to non-AIX systems may not provide transparency, because BNU commands on some systems recognize and transmit only 7-bit ASCII).

The use of host and path names is not fully transparent. The name of the host which is being logged into or to which files are being sent must be in ASCII characters only. Although path names can contain extended characters, they must be used with care. The **uucp** program passes extended-character path names properly. If the receiving system is part of the cluster, it must be running the same operating system and can interpret the path names correctly. However, if the receiver is a remote system that is running some other operating system, it may be unable to interpret path names passed to it.

For detailed information about these commands, which are stored in the **/usr/bin** directory, refer to the BNU chapter in *Using the AIX Operating System* and to the descriptions of the individual commands in *AIX Operating System Commands Reference*.

The ct Command

This program instructs the local computer to use the telephone network to call a modem attached to a remote terminal. When the remote modem answers, the **ct** command issues a login process to the remote terminal, allowing the user on the remote terminal to log in to and perform tasks on the local system.

The cu Command

This program connects a local system to a remote system. The user can then execute commands on either computer without dropping the communications link. If the remote system is also running under a UNIX-based operating system such as AIX, a user can transfer files between the two systems.

The uucp Command

This program enables a user to copy files from one system to another system. The **uucp** command creates command and data files as necessary, and then calls the **uucico** daemon to deliver the files.

For additional information about managing **uucp**, see "Transporting Copy Requests" in topic 6.8.7.1.

The uulog Command

This command displays the contents of a log file of **uucico** or **uuxqt** activities for a specified system. BNU creates individual log files for each remote system with which a local system communicates using the **uucp**, **uuto**, and **uux** programs.

For information about using **uulog**, see "Working with Log Files" in topic 6.8.6.3.

The uuname Command

¦ Copyright IBM Corp. 1985, 1991

This program identifies compatible remote systems with which a user can communicate using BNU. The **uuname** command displays a list of all the computers networked to the local system.

For additional information about using **uuname** in system management, see "Checking Networked Systems (uuname)" in topic 6.8.5.2.11.

The uupick Command

This program retrieves files transferred with the **uuto** command, which places the files in **/usr/spool/uucppublic**, the public directory on the local system. The user then issues **uupick** with the appropriate option to receive and handle the transferred files.

The uustat Command

This program displays information about the status of transfers requested by the **uucp** and **uuto** commands, and about the status of remote command executions requested by **uux**. The **uustat** command also gives the user limited control over BNU jobs queued to run on remote systems. Using **uustat** with the appropriate flags, a user can perform the following actions:

Check the general status of connections to other system

Cancel BNU job requests

For additional information about using **uustat** in system management, see "Performing Routine Maintenance Tasks" in topic 6.8.6.

The uuto Command

This program uses the **uucp** command to transfer files to a specified user on another system. The **uuto** command places the copied file in the public directory (**/usr/spool/uucppublic**) on the recipient's system and notifies the recipient that the file has arrived. The recipient then uses the **uupick** command to receive and handle the file.

The uux Command

This program runs a specified AIX command on a specified AIX system while enabling the user to continue working on the local system. The **uux** command creates command, data, and execute files in the spooling directory on the local system, and then calls the **uucico** daemon to contact the designated system and deliver the files. The **uuxqt** daemon then executes the requested command on the designated system.

For additional information about managing **uux**, see "Executing Remote Commands" in topic 6.8.7.2.

¦ Copyright IBM Corp. 1985, 1991

6.8.4.5 Administrator Commands

BNU has a number of commands intended specifically for system managers. These are actually programs that help the system manager perform both initial administrative tasks such as installing the network facility, and routine administrative tasks such as cleaning up outdated files. You must be logged in under the **bnuadm** ID or have superuser privileges to use the administrator commands (see "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3 for information about setting up the **bnuadm** login ID).

The uucheck Program

This program, stored in **/usr/lib/uucp**, checks for the presence of directories, programs, and files required to operate BNU. The **uucheck** command also checks for certain errors in the **Permissions** file. The system manager should issue **uucheck** after installing BNU, and again after setting up the customized access permissions.

For information about using **uucheck**, see "Checking for Required Directories and Files (uucheck)" in topic 6.8.5.1.2 and "Checking for Correct Permissions" in topic 6.8.5.2.10.

The uucleanup Program

This program, stored in **/usr/lib/uucp**, has several functions, all associated with removing outdated files from the **/usr/spool/uucp** directory. It is normally executed automatically at specified times by the shell **uudemon.cleanup**, which is started periodically by instructions in the file **/usr/spool/cron/crontabs/uucp**. You can also issue **uucleanup** manually if you have superuser privileges.

For information about using **uucleanup**, see "Cleaning Up the Spooling Directories" in topic 6.8.6.2.

The Uutry Program

This program, stored in **/usr/adm/uucp**, contacts a specified system with debugging turned on. Users as well as system managers can issue this command, which provides a method of checking call processing capabilities with debugging output. (This output is both displayed on the screen of the local system and written to a specified file.) The **Uutry** command (note the uppercase U) invokes the **uucico** daemon, which in turn establishes the actual connection to the remote system and transfers the file you are sending.

For information about using **Uutry**, see "The Uutry Command" in topic 6.8.6.1.2.

¦ Copyright IBM Corp. 1985, 1991

6.8.4.6 Administrative Daemons

The BNU software includes a number of daemons, programs that the networking facility executes to handle file transfers, communication with TCP/IP, scheduling work requests, and remote command executions. These daemons are generally started automatically, either by a command or by a shell script. However, the system manager (or someone with superuser privileges) can start them manually, if necessary. The four BNU daemons are stored in **/usr/lib/uucp**.

The uucico Daemon

This program, which is the primary BNU daemon, transports the files required to send data from one AIX system to another AIX system. Both the **uucp** and **uux** commands start **uucico** to transfer command, data, and execute files to the designated system. The **uucico** daemon is also started periodically by the BNU scheduler, **uusched**, which handles transferring files queued in the local spooling directory.

For detailed information about using **uucico**, see "Transporting Copy Requests" in topic 6.8.7.1 and "Executing Remote Commands" in topic 6.8.7.2.

The uucpd Daemon

This program enables the system manager to establish communication with a remote computer by means of the Transmission Control Protocol/Internet Protocol (TCP/IP). With BNU, users can perform tasks such as transferring files between two systems or executing AIX commands on remote systems linked over either a hardwired line or a telephone line attached to a modem. In the same way, they can transfer files and execute remote commands on systems linked via TCP/IP.

> For information about using BNU in conjunction with TCP/IP, se "Setting Up BNU for Use with TCP/IP" in topic 6.8.5.4.
>
> For information about using the Interface Program for use with TCP/IP refer to the TCP/IP chapter in *Using the AIX Operating System*.
>
> For detailed information about TCP/IP, see *AIX TCP/IP User's Guide*.

The uusched Daemon

This program schedules the transfer of files that are queued in the spooling directory, **/usr/spool/uucp**, on the local system. The **uusched** daemon randomizes jobs in the queue and then starts the **uucico** daemon, which actually transfers the files.

For detailed information about using **uusched**, see "Scheduling Work in the Spooling Directory" in topic 6.8.7.3.

The uuxqt Daemon

This program executes a command on a designated system. A user issues the **uux** command to run a specified AIX command on a designated system. After creating the necessary files, **uux** starts the **uucico** daemon, which transfers those files to the public spooling directory on the specified system. The **uuxqt** daemon on every networked system periodically searches the spool directory for command-execution requests. When it locates such a request, **uuxqt** checks for necessary files and permissions and then

¦ Copyright IBM Corp. 1985, 1991

executes the specified command.

For detailed information about using **uuxqt**, see "Executing Remote Commands" in topic 6.8.7.2.

¦ Copyright IBM Corp. 1985, 1991

6.8.5 Performing Initial Administrative Tasks

Before users can take advantage of the Basic Networking Utilities, the system manager must perform certain initial tasks:

Install the BNU softwar

Customize the files used in remote communication

If your site uses TCP/IP to handle remote communication, you must also configure BNU so that the two programs can work together.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.8.5.1 Installing BNU

Installing the networking utilities consists of the following steps:

1. Copying BNU from diskette onto the fixed disk (this occurs automatically when you select the BNU option during the installation of the Extended Services facility)

2. Checking for required directories and files

3. Setting up the system administrator login and password

4. Moving existing UNIX-to-UNIX Copy Program (UUCP) files into the new BNU directories.

Warning: If your site has been using a version of the UUCP facility, the following installation will copy the BNU files, directories, and programs over your existing UUCP files, directories, and programs. However, the BNU installation will **not** destroy your existing UUCP command and data files.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.8.5.1.1 Copying the Software to Standard Storage

BNU is part of the Extended Services feature of the AIX Operating System. The first step in installing the networking utilities is to copy the BNU programs from the Extended Services distribution medium to your computer's standard storage. In AIX/370, Extended Services is delivered on Tape 1 and must be copied to normal working disk. In AIX for the PS/2, or RT, Extended Services is delivered on the Extended Services diskettes and must be copied to the fixed disk.

1. Log in with the **su** command and enter the superuser password required at your site.

2. To install just the BNU component on your system, select the appropriate option from the install menu.

3. Now follow the instructions that appear on the screen.

For additional information about installing the Extended Services options, see the appropriate AIX Operating System installation manual.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.1.2 Checking for Required Directories and Files (uucheck)

After installing the BNU software, you should issue the **uucheck** command. This command scans the data copied to the computer from Extended Services, verifying that the directories, programs, and support files required to operate the networking facility were copied successfully.

Issue **uucheck** with the **-v** flag, which provides an explanation of the way in which BNU checks the file structure:

```
uucheck -v
```

When the checking process is complete, the shell prompt ($) returns to the screen.

If **uucheck** reports any errors, repeat the installation process. If you continue to have problems, refer to the AIX installation manual for your workstation. If you cannot resolve installation problems after consulting the AIX installation manual for your workstation, confer with your system support team or follow the usual procedures at your site for reporting problems.

Note: In addition to issuing **uucheck** at this point, you should also use it after customizing the **Permissions** file because the command examines that file too for certain errors. For detailed information about verifying the **Permissions** entries, see "Checking for Correct Permissions" in topic 6.8.5.2.10.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.1.3 Setting Up the BNU Manager's Login and Password

Anyone with superuser authority can access and edit the BNU programs and files. If you are managing an AIX installation (including BNU) for a relatively small group of users for whom tight security is not an issue, you probably won't find it necessary to set up a separate BNU administrative login. In such a case, the **su** command and its associated password will generally be sufficient for managing the computers at your site. Just remember to protect the superuser password and change it as often as necessary to maintain the required level of security.

However, the installation you're managing may be large enough to require several system administrators, and you may find that time constraints or security requirements necessitate having different individuals maintain different parts of the installation. In that case, you may prefer to limit the number of superusers who can access and modify **all** elements of the software and to create additional administrative logins with the appropriate user ID (UID) and group ID (GID) for individuals responsible for discrete parts of the code.

Note: Administrative logins for users responsible for maintaining the BNU software should normally have a UID of **5** and also a GID of **5**. See "Types of User Accounts" in topic 1.2.5.2 for information about superuser and user accounts.

To set up a BNU administrator's login and password, follow these steps:

1. Enter the **users** command at the superuser prompt (#).

2. Now enter the **add** subcommand to add the new login ID, which is **bnuadm** in the following example:

   ```
   > a u bnuadm
   ```

3. You need to change the user ID, so answer **n** to the **OK? (y)** prompt.

4. Enter the following in response to the **Field?** prompt:

   ```
   Field? uid
   ```

5. Now enter **5** in response to the prompt for the user ID.

   ```
   uid 5
   ```

6. Enter **group** in response to the next **Field?** prompt:

   ```
   Field? gr
   ```

7. Enter **uucp** in response to the **group** prompt:

   ```
   group uucp
   ```

8. You need to enter a password for the new login ID, so answer **n** to the **OK? (y)** prompt.

9. Enter **pa** in response to the **Field?** prompt:

   ```
   Field? pa
   ```

10. Enter the password for the **bnuadm** login ID in response to the **password**

¦ Copyright IBM Corp. 1985, 1991

prompt.

11. The directory field should contain the following entry:

    /usr/adm/uucp

    If it does not, enter **dir** in response to the **Field?** prompt and then enter the correct name in response to the prompt for a directory.

12. The program field should contain the following entry:

    /bin/sh

    If it does not, enter that program name in response to the prompt.

13. To end the procedure, press **Enter** at the next **Field?** prompt. The **users** command displays the new information (the password is encrypted).

14. If the information is correct, press **Enter** after the **OK? (y)** prompt. Press **Enter** again after the prompt **Standard new user initialization? (y)**.

15. Enter the **q** subcommand to exit from the **users** program.

The system adds the new **bnuadm** login ID to the **uucp** entry in the **/etc/group** file and to the list of users in the **/etc/passwd** file.

For information about the **/etc/group** file, see "The Group File" in topic 1.2.5.4.2.

For information about the **/etc/passwd** file, see "The /etc/passwd File" in topic 1.2.5.4.1.

For additional information about the **/etc/passwd** file and encrypting passwords, see *AIX Operating System Technical Reference*.

Once you have set up the **bnuadm** login ID and its associated password, go on to the next step in installing BNU.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2 Setting Up Remote Communication

In order for BNU to function correctly at your site, you must customize the remote communication facilities by setting up the following:

A list of the devices used to establish a hardwired communication lin or a communication link using a telephone line and a modem

A list of autodialers (modems) used to contact remote systems via th telephone network

A list of the remote systems with which the local system ca communicate

An optional list of alphabetic abbreviations representing the prefixe of telephone numbers used to contact the specified remote systems

The appropriate access permissions specifying the way in which loca and remote systems may communicate

A schedule for monitoring the networked remote system

A set of spooling and identification files for use with TCF

You set up these lists, permissions, schedules, and procedures by modifying the following BNU data base files:

Devices	**Poll**
Dialers	**remote.unknown**
Dialcodes	**Systems**
Permissions	

Note: You will also need to edit the **/usr/spool/cron/crontabs/uucp** file to uncomment the lines that handle the automatic maintenance routines such as cleaning up the spooling directories. See "Running Automatic Maintenance Routines" in topic 6.8.8 for information about these shell scripts. If your site uses TCP/IP, you must also modify the **/etc/rc.tcpip** file. See "Setting Up BNU for Use with TCP/IP" in topic 6.8.5.4 for information about that task.

In general, BNU establishes a ***hardwired connection*** between a local and a remote system by using these files in the following way:

1. A user on a local system issues a BNU command and the name of a remote system.

2. BNU checks the **/usr/adm/uucp/Systems** file to determine whether the remote system is listed as an entry.

3. If the remote system is listed, BNU checks the **/usr/adm/uucp/Devices** file to determine the specific port, transmission rate, and type of connection (direct) to use to establish communication with the designated remote system.

4. BNU connects the two systems and, if necessary for the specified command, checks the **Permissions** file on the remote system to determine valid access permissions.

5. If the remote system allows the access, the calling user can log in,

¦ Copyright IBM Corp. 1985, 1991

transfer files, execute commands, and so on, depending on the BNU command specified in step 1.

BNU generally establishes a ***telephone connection*** (using modems) between two systems by using the data base files in the following way:

1. A user on a local system attached to a modem issues a BNU command with the name of a remote system that is also attached to a modem.

2. BNU checks the **Systems** file for that system.

3. If the remote system is listed and accessible, BNU checks the **Devices** file for specific connection information.

4. From the **Devices** file, BNU gets the type of connection, the port, the transmission rate, the type of modem, and the telephone number.

   Note: BNU gets the phone number associated with the specified system from the **Systems** file. If the system manager has entered a dial-code abbreviation in the **Dialcodes** file, BNU gets the prefix for the number from that listing.

5. All modems listed in the **Devices** file are also listed in a file named **Dialers**. BNU gets the initial handshaking information about the local modem from the **Dialers** file and contacts the modem on the remote computer using the phone number from the **Systems** file.

6. BNU connects the two systems.

7. If the remote system allows the access, the calling user can log in.

For an example of the way in which the files discussed in the following sections work together to establish a connection between two systems, see "Sample Configuration Files" in topic 6.8.5.2.9.

Note: Once you have customized the files discussed in this section, particularly the **Systems** and **Permissions** files, users at your site can send mail messages via the BNU communication facilities. Simply by entering the **mail** command with the appropriate address, a user can send a message to any other user on the local or a remote computer. For additional information about mail, refer to the following:

> For information about the **mail** command, see *AIX Operating System Commands Reference*.
>
> For information about using the mail facility, see *Using the AIX Operating System*.
>
> For information about managing the mail facility, see Chapter 6, "Managing the Electronic Mail System" in topic 6.6.

Subtopics
6.8.5.2.1 Setting Up Devices
6.8.5.2.2 Setting Up Dialers
6.8.5.2.3 Customizing the Systems File
6.8.5.2.4 Setting Up Dialing Codes
6.8.5.2.5 Customizing the Permissions File
6.8.5.2.6 Customizing the Poll File
6.8.5.2.7 Customizing the remote.unknown File

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.1 Setting Up Devices

The first step, and one of the most important, in customizing BNU for use at your site is to set up a list of the devices that the networking facility will use to establish connections to remote systems. This task involves editing the file **/usr/adm/uucp/Devices**, which contains data about hardwired, telephone, and TCP/IP communication links.

Note: For detailed information about the hardware used in BNU connections, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

This section includes the following information:

An explanation of standard entries in a **Devices** file

A discussion of dialer-token pair

Several sample device entries, including connections for bot autodialers and hardwired lines (TCP/IP connections are noted in this section and described in more detail in "Setting Up BNU for Use with TCP/IP" in topic 6.8.5.4.)

Additional information about lock files

You must be logged in as the BNU administrator (**bnuadm** is the example login ID), or as **su**, to edit the **Devices** file, which is owned by the **uucp** program login ID.

For information about setting up a BNU administrator's login, se "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3.

For information about the **uucp** program login ID, see "Setting Up a Systems File" in topic 6.8.5.2.3.

For information about login IDs and their associated passwords, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

Standard Entries in a Devices File: Each entry in a **Devices** file includes the following fields:

Caller, which specifies the type of hardwired or autodialer device

Line, which specifies the device name for the port

Line2 depends on the **Caller** entry (seldom used)

Class, which specifies the transmission speed

Dialer-Token pairs, which specify a particular type of autodialer (modem) and the token (a defined string of characters) that is passed to the dialer.

You must have an entry in each of these fields, which are arranged in the following order:

Caller Line Line2 Class Dialer-Token Pairs

A standard entry for a hardwired connection looks like this:

¦ Copyright IBM Corp. 1985, 1991

```
Direct  tty0   -   1200
zeus    tty0   -   1200
```

A standard entry for a telephone connection using a modem looks like this:

```
ACU     tty2   -   1200   hayes
```

The Caller Field

You must type one of the following keywords in this field:

Keyword	Explanation
Direct	Use this keyword, which must begin with an uppercase D, if your site uses hardwired lines to connect multiple systems.
ACU	Use this keyword, which you must type in uppercase letters, if your site connects multiple systems over the telephone network using Automatic Calling Units (autodialers, or modems).
NETWORK	Type TCP (in uppercase letters), if your site uses TCP/IP. (TCP/IP is the network currently used with BNU.)
system_name	Type the name of a particular remote system hardwired to the local system. The **system_name** is the name assigned to each remote system, such as **hera**, **merlin**, **opus**, **spock**, and so on.

The Line Field

Type the device name for the line, or ***port***, used in the communication link. For example, you can use the appropriate tty device name for a hardwired line, such as **tty1**. For a line connected to an ACU (a modem), again use a device name appropriate to the dialer, such as **tty1** or **tty0**. If the device is on another TCF cluster, the device name must be a full path name, for example, **/machinename/dev/tty00**.

The Line2 Field

If you have typed the ACU keyword in the caller field and the autodialer in the dialer-token pair is a standard 801 dialer, type the device name of the 801 ACU in this field. For example, if the caller is **ACU** and the line is **tty0**, the **Line2** entry might be **tty1**. If the device type is not 801, you must type a hyphen, or dash (-) in this field as a placeholder.

Note: The **Line2** field is used only to support older modems that require 801-type dialers. The modem is plugged into one serial port, and the 801 dialer is plugged into a separate serial port.

The Class Field

For a hardwired line, type the transmission rate of the device connecting the two systems.

For a telephone connection, type the speed at which the ACU transmits data, such as **300** or **1200** baud.

Some devices can be used at any speed, in which case you should type the word **Any** (note the uppercase A). This entry tells BNU to match any speed

¦ Copyright IBM Corp. 1985, 1991

requested in the **Systems** file.

The Dialer-Token Pairs Field

For a hardwired connection, enter the word **direct**.

For a telephone connection, enter the type of dialer and the token that is passed on to that modem. The token is either a telephone number or a predefined string used to reach the dialer.

The Dialer Entry: For a telephone connection, enter one of the following as the dialer:

Entry	Definition
hayes	This is a Hayes dialer.
801	This is a standard 801 autodialer, with a separate 212- or 103-type modem.
OTHER DIALERS	These are other dialers that you can specify by including the relevant information in the **Dialers** file. Examine this **Dialers** file to see other possible dialer types.
NETWORK	This represents a communications network; TCP/IP is the network currently supported by BNU. Type **TCP** here if you have also used **TCP** as the keyword in the caller field.

Each dialer included as part of a dialer-token pair in the **Devices** file is also included as an entry in the **Dialers** file. See "Setting Up Dialers" in topic 6.8.5.2.2 for information about the contents of this file.

The Token Entry: The **Token** following the dialer represents either a telephone number or a predefined string used to reach the dialer. If the token represents a telephone number, you can generally leave this part of the field blank, specifying that BNU should use the telephone number listed in the **Systems** file.

Note: Some sites do not include complete telephone numbers in the **Systems** file. Instead, the entry in that file contains only the last four digits of the number, preceded by a dial-code abbreviation. This abbreviation references the remainder of the phone number (such as a **9** for an outside line, the three-digit exchange number, or an access code), which is actually contained in the **Dialcodes** file. However, it is often more efficient to include the complete telephone number in the **Systems** file.

You can also use the characters \D or \T as tokens. The \D string is the default token in a dialer-token pair.

The \D token specifies that BNU should take the phone number listed in the **Systems** file and pass it to the appropriate "dialer script" (entry) in the **Dialers** file **WITHOUT** including the dial-code abbreviation. Leaving the token field blank is the same as entering \D, so a blank is usually sufficient as a token if you have included complete telephone numbers in the **Systems** file.

However, if you are using dial-code abbreviations specified in the

¦ Copyright IBM Corp. 1985, 1991

Dialcodes file for certain telephone numbers, you **MUST** enter the \T string as the token in those entries in the **Dialers** file. The \T token instructs BNU to process the phone number by including the data specified in the **Dialcodes** file.

Sample Entries in a Devices File: The following examples illustrate entries in a **Devices** file for various types of connections. The number preceding the entry is **not** part of the **Devices** file. It is simply a number used to reference that entry in the explanation following the examples. Remember that the first field represents **Caller**, the second **Line**, the third **Line2**, the fourth **Class** , and the fifth the **Dialer-Token pair**. These examples do not include a token; that part of the entry is blank.

```
1. Direct /site1/dev/tty0    -    1200
2. zeus   /site2/dev/tty0    -    1200
3. Direct /site1/dev/tty1    -    1200
4. hera   /site2/dev/tty1    -    1200
5. ACU    /site1/dev/tty2    -    1200  hayes
6. ACU    /site1/dev/tty3    -    1200  hayes
7. ACU    /site2/dev/tty3    -     300  hayes
8. ACU    /site1/dev/tty4   tty5  1200  801
9. ACU    /site1/dev/tty6   tty7   300  801
```

The first four entries specify information about hardwired (**Direct**) connections. The rest of the entries in this sample **Devices** file specify connections made over a telephone line using a modem.

Explanation of Sample Hardwired Connections: Setting up entries for hardwired connections to remote computers is very much like setting up entries for telephone connections. It differs in only two ways:

- It is easier to set up entries for hardwired lines than for telephon connections because the connection itself is simpler. Hardwired connections, by definition, do not require any additional software or hardware, such as a modem.

- In general, each entry for a hardwired connection consists of tw parts: the first specifies the port (line) that the BNU command uses to connect to the remote system, which is specified in the second part. However, if the two systems use a permanent virtual circuit connection, the entry occupies a single line in the **Devices** file.

In lines 1 and 2, (the first example), the caller field lists **Direct** (for a direct connection) in the first part and **zeus** (the name of the remote system) in the second part. The local system is connected to **zeus** via device **tty0**, which is listed in the line field in both parts of the example:

```
Direct /site/dev/tty0    -    1200
zeus   /site/dev/tty0    -    1200
```

The line2 field contains actual data only when the entry specifies a certain type of telephone connection. A hyphen is used as a placeholder in other types of connections, as is the case in lines 1 and 2. This tty device transmits at **1200** bps (bits per second), which is listed in the class field in both parts of the example. There is no entry in the dialer-token field because a hardwired connection does not require a modem.

¦ Copyright IBM Corp. 1985, 1991

Lines 3 and 4 (the second hardwired example) are almost exactly like the entries in lines 1 and 2:

```
Direct /site/dev/tty1    -    1200
hera   /site/dev/tty1    -    1200
```

Again, the caller field lists **Direct** in the first part of the entry and the name of the remote system, **hera**, in the second part. The device name is **tty1** in both line fields, hyphens function as placeholders in the line2 fields, and the tty devices communicates at **1200** bps, as listed in the class field.

Explanation of Sample Autodialer Connections: Notice that in each of these telephone-connection entries, the caller is specified as an autodialer (an ACU, or Automatic Calling Unit). You should use ACU as the caller in all remote connections established over a phone line.

The entry in line 5 is the first autodialer connection in the sample **Devices** file:

```
ACU    /site/dev/tty2    -    1200  hayes
```

The line field is specified with the device name **tty2**. As was the case with the hardwired entries, a hyphen is used as a placeholder in the line2 field, and the class entry is again a transmission rate of **1200** baud. The dialer part of the dialer-token pair is specified as a **hayes** modem, and the token is left blank.

Note: Remember that a /D string or a blank in the token field tells BNU to use the complete telephone number listed in the **Systems** file. If you are using dial-code abbreviations specified in the **Dialcodes** file, you must enter the /T string as the token. All the autodialer examples in the sample **Devices** file shown above specify complete phone numbers listed in the **Systems** file by leaving the token field blank.

The entries in lines 6 and 7 specify the same modem, a **hayes**, which can be used at either **1200** or **300** baud:

```
ACU    /site/dev/tty3    -    1200  hayes
ACU    /site/dev/tty3    -     300  hayes
```

The ACUs are connected to a device named **tty3** (the line field), and the line2 field contains the hyphen placeholder.

In the entries for lines 8 and 9, the ACUs are connected to devices named **tty4** and **tty6**, specified in the line fields. In both cases, there is an entry in the line2 field because a standard 801 autodialer is specified in the dialer-token pair:

```
ACU    /site/dev/tty4   tty5  1200  801
ACU    /site/dev/tty6   tty7   300  801
```

Because 801 is specified as the dialer in these two examples, the line2 fields must contain the device names of the 801 ACUs. In this case, both are 212 modems, although **tty6** may be a 103 type. The class entry is a transmission rate of **1200** baud for the first example and **300** for the second. The token part of the dialer-token pair is blank.

Setting Up Entries for Hardwired Connections: The following example

¦ Copyright IBM Corp. 1985, 1991

demonstrates how to set up a standard **Devices** entry for a hardwired connection:

```
Direct  /site/dev/tty0     -   1200   direct
hera    /site/dev/tty0     -   1200   direct
```

1. Type the keyword **Direct**, with an uppercase D, in the **Caller** field in the first part of the entry.

2. For the caller in the second part, shown as **hera** in the example, type the name of the remote system to which you want to connect the local computer over the hardwired line.

3. In this example, the **Line** (or port) entry is a variation of the standard tty device, **tty0**. Type the device name appropriate for the hardwired connection used at your site in the line fields in both parts of the entry.

4. You must make an entry in each field, so type a hyphen for a placeholder in the **Line2** fields in both parts of the entry.

5. The **tty0** device uses a **1200** bps transmission rate. Type the transmission rate appropriate for the hardwired connection used at your site in the **Class** fields in both parts of the entry.

6. This is a direct connection, so you enter **direct** (all lowercase) in the **Dialer-Token pairs** fields in both parts of the entry.

7. Using the sample entry as a model, continue adding entries to the **Devices** file until you have listed each hardwired device connecting the local system to a remote system.

The following examples show two additional types of entries for hardwired lines:

```
Direct /site/dev/tty8     -   9600   direct
thor   /site/dev/tty8     -   9600   direct
odin           x25.cb     -   9600   direct
```

The first is a two-part entry for a hardwired connection to system **thor**. The first part specifies a **Direct** connection in the caller field, and the second part specifies the remote computer. The line entry in both parts is **tty8**, which communicates at **9600** bps. The third entry connects system **odin** to the local system using an **X.25.cb** permanent virtual circuit connection.

Setting Up an Entry for an Autodialer Connection: The following example demonstrates how to set up a standard entry in the **Devices** file for a Hayes modem:

```
ACU  /site/dev/tty1   -  1200  hayes
```

1. You are using a Hayes modem, so type **ACU** in the **Caller** field.

2. The **Line** field contains the name of the device that is attached to the modem. In this example, the line (or port) is a variation of the standard tty device, **tty1**. Type the device name appropriate for your site.

3. Type a hyphen as a placeholder in the **Line2** field because the ACU is a

¦ Copyright IBM Corp. 1985, 1991

specific modem rather than a standard 801 dialer.

4. The Hayes modem in this example operates at **1200** baud. In the **Class** field, type the baud rate appropriate for your modem and line (this can be 300, 1200, 2400, or higher, depending on the modem).

5. Enter the name of the modem as the dialer in the **Dialer-Token pair** field. The example entry is **hayes**, for a Hayes modem. If you are planning to include complete phone numbers in the **Systems** file, leave the token field blank, like the example. (A blank instructs BNU to use the default \D token.) If you are planning to use dialing-code abbreviations specified in the **Dialcodes** file, enter the token **\T**.

6. Using the sample entry as a model, continue adding entries to the **Devices** file until you have listed each connection between the local system and a remote system that uses a telephone line and a modem.

   For an example of an entry in a **Devices** file specifying a TCP/IP connection, see "Setting Up BNU for Use with TCP/IP" in topic 6.8.5.4.

   For detailed information about the devices used in BNU communications, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

Additional Information about Lock Files: Every time a user attempts to contact a remote system using a device specified in the **Devices** file, BNU creates a lock file in the directory **/etc/locks**. In a TCF cluster, this directory is a symbolic link to **/<LOCAL>/locks**. The lock file is created in the locks directory on the site where the particular device is located. For example, if you want to contact the remote system **hera** and the device is on site 1 (for example, **/site1/dev/tty2**), the lock file is created in the **/site1/locks** directory. This temporary file "locks" the device so that another user cannot access it while it is already in use. The full path name of a lock file is either

/etc/locks/LCK..device_name

or

/etc/locks/LCK..system_name

where **device_name** is the name of a device such as **tty0**, and **system_name** is the name of a system such as **hera**.

BNU uses the first form of the lock-file name whenever a connection to a remote system, established over the specified device, is actually in use. The second form of the lock-file name is used only by the **uucico** program to prevent more than one **uucico** connection at a time to the same remote system.

Under normal circumstances, the software automatically removes a lock file when the user establishes a connection to a remote system. However, if a process executing on the specified device or system does not complete its run (for example, if the computer crashes), the lock file will remain in the **/etc/locks** directory until either it is removed manually or the system is restarted after a shutdown.

Note: The line **rm -f /etc/locks/*** in the file **/etc/rc** instructs the system to clean out all the files in the **locks** directory every time

¦ Copyright IBM Corp. 1985, 1991

```
the system is restarted.  If you prefer to delete any remaining
lock files manually, simply comment out this line.
```

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.2 Setting Up Dialers

The file **/usr/adm/uucp/Dialers** contains an entry for each autodialer (other than an 801-type dialer or a TCP/IP connection) that is included in the **Devices** file. Every modem is listed on a line by itself, and each line includes a series of expect-send sequences that specify the initial "handshaking" that occurs on the communication link before it is ready to send or receive data. In this way, the local and remote systems confirm that they are compatible and configured to transfer data.

The handshaking data is included in a string that tells the **cu**, **ct**, or **uucico** programs the sequence of characters to use to dial out on a particular type of modem. These characters include entries such as **\d** to specify a delay, **\p** for a pause, **\r** for a carriage return, **\c** for a new line, and so on.

You must be logged in as the BNU administrator (**bnuadm** is the example login ID), or as **su**, to edit the **Dialers** file, which is owned by the **uucp** program login ID.

> For information about setting up an administrator's login, se "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3.
>
> For information about the **uucp** program login ID, see "Setting Up a Systems File" in topic 6.8.5.2.3.
>
> For information about login IDs and their associated passwords, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

Standard Entries in a Dialers File: The following example lists several entries in a typical **Dialers** file:

```
hayes   =,-,  ""  \dAT\r\c OK \pATDT\T\r\c CONNECT
penril  =W-P  ""  \d > s\p9\c )-W\p\r\ds\p9\c-) y/c : \E\TP > 9\c OK
ventel  =&-%  ""  \r\p \r\p-\r\p-$ <K\D%%\r>\c ONLINE!
vadic   =K-K  ""  \005\p *-\005\p-* D\p BER?  \E\D\e \r\c LINE
direct
```

The first field in the **Dialers** file matches the fifth field, the dialer-token pair, in the **Devices** file. This is the type of autodialer (modem) used in the connection. As noted above, every dialer listed in the **Devices** must also be listed in the **Dialers** file, with the exception of 801-type dialers and a TCP/IP connection.

Note: Notice that the last entry in the example above consists only of the word **direct**. This entry indicates that hardwired connections do not require any handshaking or dialer negotiations.

The second field consists of two sets of two characters, for a total of four entries. These characters make up a translation string. In the actual phone number of the remote modem, the first character in each string is mapped to the second character in that set.

This entry generally translates the characters **=** and **-** into whatever the dialer uses for "wait for dial tone" and "pause". For example, in the second line of the sample file, the = translates into **W** in the phone number, and the - translates into **P** on a Penril dialer.

The handshaking, which is usually an expect-send sequence of ASCII strings, is given in the remainder of the line. This string is generally

¦ Copyright IBM Corp. 1985, 1991

used to pass telephone numbers to a modem, or to make a connection to another system on the same data switch as the local system. If the match succeeds, the line in the **Dialers** file is interpreted to perform the dialer negotiations.

If the fifth field in the **Devices** file is not a built-in function, this field serves as an index to the **Dialers** file.

Sample Entry in a Dialers File: The following example interprets the first line in the **Dialers** file shown above. This is a standard entry that you can include in your **Dialers** file, but you may need to modify it for use at your site.

```
hayes   =,-,  ""  \dAT\r\c OK \pATDT\T\r\c CONNECT
```

Following is an explanation of how each entry affects the action of the dialer.

Entry	Action
=,-,	Translate the telephone number. Any **=** represents "wait for dial tone" and any **-** represents "pause". Both are sent to the modem as commas.
""	Wait for nothing; continue with the rest of the string.
\dAT	Delay; then send **AT** (the Hayes Attention prefix).
\r\c	Send a carriage return (**r**) followed by a new line (**c**).
OK	Wait for **OK** from the remote modem, signaling that the first part of the string has executed.
\pATDT	Pause (**p**); then send **ATDT**. **AT** is again the Hayes Attention prefix, **D** represents a dialing signal, and **T** represents a touch-tone dial tone.
\T	Send the telephone number, which is specified in the **Systems** file.
\r\c	Send a carriage return and a new line following the number.
CONNECT	Wait for **CONNECT** from the remote modem, signaling that the modems are connected at the baud rate specified in the **Devices** file.

If you need to modify this example for use at your site and are unsure about the appropriate entries in the handshaking string, refer to the documentation that accompanied the modems that you are including in the **Dialers** file.

For an example of an entry in a **Dialers** file, see "Sample Configuration Files" in topic 6.8.5.2.9.

For information about the hardware used in BNU communication, se "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.3 Customizing the Systems File

Each entry in the **/usr/adm/uucp/Systems** file represents a remote system with which the local system can communicate. BNU cannot establish a communication link between the local computer and a remote computer unless the remote system is listed correctly in this file. You must set up a **Systems** file on every computer at your site that uses the BNU facility. If TCF is installed, you must set up a **Systems** file on every TCF cluster that uses the BNU facility. If TCF is installed, there is only one **Systems** file for the entire cluster.

The entries in the **Systems** file include the name of the remote system, the time during which users can establish a connection between the local and the remote system, whether the connection uses a hardwired, telephone, or TCP/IP communications link, the speed at which the line transmits data, the phone number used with a modem, and information required to log in on the remote system.

The mail facility uses the entries in the **Systems** file to send messages to system users; mail is sent only to users on those computers listed in this file. The mail facility also uses the access permissions established in the **Permissions** file (see "Customizing the Permissions File" in topic 6.8.5.2.5).

Note: If you have installed and configured the **sendmail** program from Extended Services, you can also send mail to other remote systems that are either listed in the **/etc/hosts** file, or for which the Internet address is known, as long as these host computers are running a **sendmail** or SMTP daemon. However, you cannot use an alias name for the remote system, even if that alias is defined for the domain name daemon, when you are using BNU to send mail or remote-command execution requests over a TCP/IP connection.

You must be logged in as the BNU administrator (**bnuadm** is the example login ID), or as **su**, to edit the **Systems** file, which is owned by the **uucp** program login ID. Users cannot access this file unless they know the password for one of the valid logins.

For information about setting up an administrator's login, se "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3.

For information about the **uucp** program login ID, see "Setting Up a Systems File."

For information about login IDs and their associated passwords, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

Standard Entries in a Systems File: Each entry in a **Systems** file has the following form:

System_name Time Caller Class Phone Login

A standard entry for a hardwired connection between a local and a remote system looks like this:

```
hera  Any  hera  1200  -  login:--login: uucp  word:sysuucp
```

A standard entry for a telephone connection using a modem looks like this:

```
merlin  0830-1730  ACU  1200  123-4567  in:--in: uucp1  word: passuucp
```

¦ Copyright IBM Corp. 1985, 1991

The System_name Field

This is the name of the remote computer that was assigned when the system was installed at your site. The assigned name is often an indication of the system's use, such as **cad**, **pubs**, or **dev**, but many system managers prefer to assign names such as **spock**, **hal**, or **mrdata**. In general, system names should be a maximum of seven characters in length. In order to be compatible with some older systems, such names should include only lowercase characters (or digits). The system names must contain only single-byte ASCII characters from the set of ASCII characters.

You can list a specific computer in the **Systems** file more than once. Each additional entry for a specific system represents an alternate communication path that BNU will use in sequential order to try and establish a connection between the local and the remote system.

The Time Field

This is a string that indicates the days of the week and the times of day during which users on the local system can communicate with the specified remote system. For example, the string **MoTuTh0800-1730** indicates that local users can contact the specified remote system on Mondays, Tuesdays, and Thursdays from 8 a.m. until 5:30 p.m.

As indicated in the example above, the day part of the entry can be a list including any day or days represented by **Mo**, **Tu**, **We**, **Th**, **Fr**, **Sa**, or **Su**. You may also enter **Wk**, if users can contact the remote system on any week day, or **Any** if they can use the remote system on any day of the week including Saturday and Sunday.

Enter the time at which users can contact the remote system as a range of times, using the 24-hour clock notation. For example, if users can communicate with the specified remote system only during the morning hours, type a range such as **0800-1200**. If users can contact the remote computer at any time of day or night, simply leave the time range blank.

You can also specify times during which users cannot communicate with the remote system--that is, you can specify a time range that spans 0000. For example, typing **0800-0600** means that users can contact the specified system at any time **except** between 6 a.m and 8 a.m. This is useful if you need a free line at a certain time of day in order to use the remote system for administrative purposes.

You can include multiple time fields by using a comma (,) as a separator. For example, **Wk1800-0600,Sa,Su** means that users can contact the remote system on any week day at any time except between the hours of 6 p.m. and 6 a.m., and at any time on Saturday and Sunday.

You can also include an optional subfield that specifies the minimum time in minutes between an unsuccessful attempt to reach the remote system and the retry time when BNU again attempts to communicate with that system. This subfield is separated from the rest of the string by a semicolon (;). For example, **Wk1800-0600,Sa,Su;2** indicates that if the first attempt to establish communication fails, BNU should continue to attempt to contact the remote system at two-minute intervals.

Note: If you include this subfield, it overrides the default retry time.

The Caller Field

¦ Copyright IBM Corp. 1985, 1991

The available callers are **ACU** for a telephone connection using a modem, **system_name** for a hardwired connection, and **TCP** for a connection using TCP/IP.

If you use **TCP**, there is a subfield associated with the caller that specifies a conversation protocol. The default, which you do not have to type, is **g**. To use a different subfield, enter it with a comma and the letter representing one of the other conversation protocols--either **t** or **e**. These protocols are faster and more efficient than the **g** protocol.

Use either the **t** or **e** protocol to communicate with a site running the AIX version of BNU. Use the **e** protocol for a site running a non-AIX versions of BNU. Use the **t** protocol for sites running the Berkeley version of the UNIX-to-UNIX Copy Program (UUCP). See "Setting Up BNU for Use with TCP/IP" in topic 6.8.5.4 for additional information about these subfields.

The Class Field

This is the speed at which the specified hardwired or telephone line transmits data. It is generally **300**, **1200**, **2400** or higher for a hardwired device, or **300**, **1200**, or **2400** for a telephone connection.

Unless it is necessary to enter a specific transmission rate in this field, you can always use the word **Any**. This instructs BNU to match any speed that is appropriate for the ACU (Automatic Calling Unit) or system connection that you specified in the caller field.

Note: For a telephone connection, the rate you enter in this field should correspond to the rate you entered in the **Class** field in the **Devices** file for this particular ACU. Do not include a transmission rate for a TCP/IP connection. Instead, type a hyphen (-) as a placeholder.

As was the case with the **Devices** file, you must have an entry in every field of a line in the **System** file. If you do not type a transmission rate in this class field, use a hyphen as a placeholder, as indicated above.

The Phone Field

You must make an entry in this field. If you are using a hardwired connection, type a hyphen as a placeholder.

If this entry represents a telephone connection using a modem, you can type the remote modem's phone number, composed of a three-digit alphabetic or numeric exchange prefix and a four-digit number, in one of two ways:

Type the complete phone number of the modem. If the system is in you local dialing area, type the seven-digit phone number.

If the system is not in your local dialing area, you should also include any other numbers required to reach the remote modem. These numbers may include a code for an outside line, any appropriate long-distance access codes, and **1** plus the area code. Then type the seven-digit number of the remote modem.

Note: This type of entry is the most efficient method of including phone numbers if your site uses only a relatively small number of telephone connections. However, if your site includes a

¦ Copyright IBM Corp. 1985, 1991

large number of remote connections established via a phone line and a modem, you may prefer to enter those numbers in the manner described below.

For example, if the modem is located in Los Angeles, the phone field might be:

```
2131234567
```

The most extensive phone field you are likely to encounter would be the sequence required for dialing a modem placed from a phone outside of the U.S.A. For example:

```
9011012131234567
```

where:

9 Accesses an outside line

001 Accesses international dialing

01 is the country code for the U.S.A.

213 is the area code for Los Angeles

123 is the local exchange code

4567 is the local phone number.

Type an optional alphabetic abbreviation that represents the dialin prefix and then type the four-digit phone number. If you choose this method, you must make certain that you have included the dialing prefix in the **Dialcodes** file.

For example, if your site communicates regularly via modems to multiple other systems all located at the same remote site, it is more efficient to use a dial-code abbreviation in the **Systems** file than to type the complete phone number of each remote modem.

Suppose that it is necessary to dial a 9, an access code, an area code, and a phone number in order to reach a remote modem. You could type a prefix representing these numbers, together with the four-digit modem number, for each remote system listed in the **Systems** file.

Then, in the **Dialcodes** file, enter the prefix and the numbers associated with it. Note that you need to enter this prefix in the **Dialers** file **only once** for all the remote modems listed in the **Systems** file. See "Setting Up a Systems File" for an example of this usage and "Setting Up Dialing Codes" in topic 6.8.5.2.4 for information about using dial-code abbreviations.

Note: For callers that are actually switches, the phone field is the token the switch requires to get to the particular computer. The token you enter here is used by the caller functions specified in the **Devices** file.

For example, the following phone field could be used in the **Systems** file:

```
boston4567
```

¦ Copyright IBM Corp. 1985, 1991

You would also have to make an try in the **Dialcodes** file to map the letters in the above example to the numbers they represent. For example:

```
boston  1617123
```

The Login Field

Before allowing the calling local system to establish a connection, the designated remote system must receive certain login information from the local system. You specify this login information in a series of fields and subfields called **expect-send** characters.

Expect-Send Characters in Login Fields:

You enter the required login information in the following form:

[**expect send**] ...

The **expect** field contains characters that the local system expects to receive from the remote system. Once the local system receives those characters, it sends another string of characters that make up the **send** field.

For example, the first **expect** field generally contains the remote system's login prompt, and the first **send** field generally contains the login ID of the remote system. The second **expect** field contains the remote password prompt, and the second **send** field contains the remote system's password.

The **expect** field may include subfields entered in the following form:

expect[--**send**--**expect**] ...

In this case, the first **expect** still represents the string that the local system expects to receive from the remote system. However, if the local system does not receive (or cannot read) that first **expect** string, it sends its own string (the **send** string within the brackets) to the remote system. The local system then expects to receive another **expect** string from the remote system.

For example, the **expect** string may contain the following characters:

```
login:--login:
```

The local system expects to receive the string **login:**. If the remote system sends that string and the local system receives it correctly, BNU goes on to the next field in the expect-send sequence. However, if the local system does not receive **login:**, it sends a null character followed by a new line and then expects to receive a second **login:** string from the remote computer.

Note: If the remote system does not send an **expect** string to the local system, you must type two double quotation marks (**""**), representing a null string, in the first **expect** field. For information about using this form of entry, see "Login Failures" in topic 6.8.9.5. Also, every time the local system sends a field, it automatically transmits a new line following that **send** field. If you do not want to include this automatic new line, make the last two characters in the send string a back slash and the letter c (**\c**).

There are two special strings you can include in the login sequence. The

¦ Copyright IBM Corp. 1985, 1991

string **EOT** sends an **EOT** (End of Transmission) character, and the string **BREAK** attempts to send a **BREAK** character.

You may include the following characters in **expect-send** strings in login fields in the **Systems** file:

String	Explanation
N	Null character.
b	Backspace character.
c	If at the end of a field, suppress the new line that normally follows the characters in a send field. Otherwise, ignore this string.
d	Delay 2 seconds before sending or reading more characters.
p	Pause for approximately 1/4 to 1/2 second.
E	Turn on the echo check (useful in the **Dialers** file).
e	Turn off the echo check (useful in the **Dialers** file).
K	Send a &BREAK. character. This is the same as entering **BREAK**.
n	New-line character.
r	Carriage return.
s	Space character.
t	Tab character.
\\	Backslash character.
EOT	**EOT** character. When you enter this string, the system sends two EOT new-line characters.
BREAK	**BREAK** character.
ddd	Collapse the octal digits (**ddd**) into a single character and send that character.

Sample Entry in a Systems File: Following is a line representing a typical entry in a **Systems** file. Remember that the first field represents the name of the remote system, the second is the time during which users can reach the remote system, the third is the caller (hardwired, telephone, or TCP/IP) used for the connection, the fourth is the transmission rate, the fifth is the phone number of the remote modem, and the sixth is the login sequence.

```
  hera  Any  ACU  1200  ch6412  login:--login: uucp  word: sysuucp
```

Setting Up a Systems File: You can set up the entries in your site's **Systems** file based on the model shown in the above example.

1. Enter the **System_name** of the remote system.

¦ Copyright IBM Corp. 1985, 1991

In the example, the name of the remote system is **hera**.

Type the name of the remote system with which you want the local system to communicate.

2. Enter the **Time** during which the local system can communicate with the remote system.

   The example entry is **Any** (notice the uppercase A). This specifies that users on the local system can contact **hera** on any day of the week at any time. If **Any** is not appropriate for this entry at your site, use **Wk** for any weekday or a specific day or days (**Mo**, **Tu**, **We**, **Th**, **Fr**, **Sa**, or **Su**). If you want to specify a time period during which users can connect to the remote system, enter a time range using the 24-hour clock notation.

   For example, if local users can communicate with the remote system only between the hours of 8 a.m. and 5 p.m. on weekdays, type the following string in the **Time** field:

   ```
   Wk0800-1700
   ```

3. Enter the type of **Caller** used to establish the remote communication.

   The caller field in the example contains the string **ACU**. This indicates that the connection is established over a telephone using the appropriate line listed in the **Devices** file and the appropriate modem listed in the **Dialers** file.

   If you are setting up a hardwired connection to a remote computer, type the name of the remote system in this field. This is usually the same name that you used in the **System_name** field in step 1.

   If you are setting up a telephone connection, type **ACU**.

   If you are setting up a TCP/IP connection, type **TCP**.

4. Enter the **Class** that represents the speed of the transmission.

   In the example, the entry in this field is **1200** (bps). This is the rate at which data is transmitted over the communication link specified in the **Caller** field.

   Type the transmission rate appropriate to your entry in the caller field. Enter a hyphen (-) as a placeholder for a TCP/IP connection.

5. Enter either a hyphen or a telephone number in the **Phone** field.

   The example entry uses **ACU** in the caller field, so the phone field contains the telephone number of the remote modem, **ch6412**. This is actually only part of the phone number; the **ch** prefix instructs BNU to find the remainder of the phone number in the **Dialcodes** file.

   If you are setting up an entry for a hardwired or TCP/IP connection, use a hyphen as a placeholder in the phone field.

   If you are setting up an entry for a telephone connection, either you can type the complete telephone number of the remote modem in the phone field, or you can type a dial-code abbreviation defined

¦ Copyright IBM Corp. 1985, 1991

in your **Dialcodes** file plus the four-digit phone number.

See "The Dialer Entry" in topic 6.8.5.2.1 for additional information about telephone numbers used to establish communication with a remote modem and "Setting Up Dialing Codes" in topic 6.8.5.2.4 for information about dial-code abbreviations.

6. Enter the **expect-send** strings in the **Login** field.

   Note: The sample entry contains four strings in this field. The actual number of strings can vary, depending on the remote system and the complexity of the login sequence. Four to six strings are generally sufficient for BNU communications between two AIX systems, but, again, that number can vary.

   a. Type the first string that the local system should expect to receive from the remote system. This is generally the remote system's login prompt.

      The first expect string in the sample entry includes the characters **login:--login:**. This entry indicates that the local system expects to receive a string containing the word **login:**, which is the remote system's login prompt. If the local system receives that string, BNU goes on to the next field, the first send characters. If the local system does not receive (or correctly read) the first expect string, it sends a carriage return to the remote system, expecting then to receive a second **login:** string from that system.

      You do not have to enter the complete prompt; you can use part of the string. For example, the sample entry could use the string **in:** rather than the complete string **login:**. However, the partial string must end with the last character that the local system expects to receive from the remote system.

      Type the login prompt used by the remote system to which you are setting up a connection in the first expect field.

      Note: It is always a good idea to set up this field with the login prompt included a second time in the **send-expect** subfield, as shown in the example.

   b. Type the first string that the local system will send to the remote system. This is generally the remote system's login ID.

      The sample entry uses the string **uucp**, which represents the **uucp** program login ID. In general, BNU uses a version of this login ID as the first send string in the **Login** field in the **Systems** file. (However, on some remote computers this send string could contain a login ID that is not a version of **uucp**, depending on the administrative customs at the remote site.) See "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8 for more information about login IDs and their associated passwords.

      Note: Users do not need to know about the **uucp** program login ID. When employing the BNU facility to transfer files or execute remote commands, users need only enter the appropriate command with the required system name. The **uucp** program login ID, which appears as an entry only in the **Systems** and **Permissions** files, is used internally by

¦ Copyright IBM Corp. 1985, 1991

BNU for transporting files.

Type the login ID of the remote system, which your local system will transmit in response to the remote system's login prompt.

For detailed information about setting up login IDs and their associated passwords, see "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

c. Type the second string that the local system should expect to receive from the remote system. This is generally the remote system's password prompt.

The example entry uses the string **word:**. The actual remote system prompt used in the example is **password:**, but (as indicated in the discussion about the remote system's login prompt) you don't have to type the complete expect string.

In this case, you could use the complete string **password:**, but you can also use a partial string, such as **ssword:** or even **ord:**, as long as the last character in the string is the last character that the local system expects to receive from the remote system. Notice that in the example, the last character in the string is a colon (:).

Type the complete or partial string that represents the password prompt of the remote system for which you are setting up the **Systems** entry. Be sure to include, as the last part of that string, any character that follows the password prompt, such as :, >, #, and so on.

d. Type the second string that the local system will send to the remote system. This is generally the remote system's password.

The sample entry uses the password **sysuucp**. In this example, that is the password associated with the **uucp** program login ID on the remote system.

Note: The encrypted password associated with the **uucp** program login ID is stored in the **/etc/passwd** file. If security is a consideration at your site, you can change the password for the **uucp** login ID frequently. For information about the **uucp** program login ID, see "Setting Up a Systems File." For additional information about login IDs and passwords, see "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

Enter the remote system's password, which your local system will send in response to the password prompt.

7. Using the sample entry as a model, continue adding entries to the **Systems** file until you have specified every remote system with which you want the local system to communicate.

Note: Remember that you must set up this type of entry in the **Systems** file on every computer at your site that uses the BNU facility.

For additional examples of entries in **Systems** files, see "Sample Configuration Files" in topic 6.8.5.2.9.

¦ Copyright IBM Corp. 1985, 1991

For examples of using the **uucp** program login ID and variations of that login with TCP/IP, see "Setting Up BNU for Use with TCP/IP" in topic 6.8.5.4.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.4 Setting Up Dialing Codes

The **Dialcodes** file contains dial-code abbreviations and partial phone numbers that complete the telephone entries in the **Systems** file. If your site uses a large number of remote connections established over phone lines and modems, you may find it useful to set up such dial-code abbreviations.

You must be logged in as the BNU administrator (**bnuadm** is the example login ID), or as **su**, to edit the **Dialcodes** file, which is owned by the **uucp** program login ID.

> For information about setting up an administrator's login, se "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3.
>
> For information about the **uucp** program ID, see "Setting Up a Systems File" in topic 6.8.5.2.3.
>
> For detailed information about setting up other BNU login IDs an passwords, see "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

Entries in the **Dialcodes** file contain an alphabetic prefix attached to a partial phone number that may include, for example, codes for outside lines, long-distance access codes, area codes, and three-digit exchange numbers. You enter the relevant alphabetic prefix (representing the partial phone number), together with the remaining four digits of that number, in the **Phone** field in the **Systems** file.

If users at your site communicate regularly via telephone lines and modems to multiple systems all located at the same remote site, or to multiple systems located at different remote sites, you may find it more efficient to use dial-code abbreviations in the **Systems** file than to enter the complete phone number of each remote modem in that file.

Note: If your site uses only a relatively small number of telephone connections to remote systems, you will probably find it more efficient to include the complete phone numbers of the remote modems in the **Systems** file than to use dial-code abbreviations.

Suppose that it is necessary to dial a 9, an access code, an area code, and a phone number in order to reach remote modems at a site with which your users communicate on a regular basis. Rather than typing 15 or more digits for each modem at the remote site in the **Systems** file, you can simply enter an alphabetic prefix (set up in the **Dialcodes** file) and the remaining four digits of the phone number for each remote modem.

Following is the format of an entry in a **Dialcodes** file:

abv dialing_sequence

The **abv** part of the entry is an alphabetic prefix, containing up to eight letters, that you establish when you set up the dialing-code listing. The **dialing_sequence** is composed of all the digits in the number that precede the actual four-digit phone number.

Following is an example entry in the **Systems** file that includes a dial-code abbreviation:

```
zork  Any  ACU  1200  boston4567  login:--login: uucp2  word: leather
```

¦ Copyright IBM Corp. 1985, 1991

Following is the relevant entry in the **Dialcodes** file for this dialing prefix:

```
boston 9=1617123
```

Note: You need to enter this prefix **only once** in the **Dialcodes** file. Once you have set up a dial-code abbreviation, you can use that prefix in all relevant entries in the **Systems** file.

When the user wants to communicate with system **zork**, he or she simply enters the appropriate command and the system name, such as **cu zork**. The modem attached to the local system contacts the modem attached to **zork**, using the number **9-1-617-123-4567** (the hyphens are optional).

For additional examples of entries in a **Dialcodes** file, see "Sample Configuration Files" in topic 6.8.5.2.9.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.5 Customizing the Permissions File

Each computer that uses BNU in your installation requires both a **Systems** and a **Permissions** file entry. The **/usr/adm/uucp/Permissions** file contains information about the ways in which the remote computers listed in the **Systems** file are allowed to carry out **uucico** and **uuxqt** transactions with a local system.

The system manager must set up entries in the **Permissions** file that specify a remote system's login ID, whether that remote system is allowed to send files to and receive files from the local system, and which commands the remote system is permitted to execute on the local system.

Note: The ***access permissions*** that you set in a **Permissions** file affect only remote systems; they do **not** pertain to individual users who work on those remote systems. Permissions limiting **uucico** and **uuxqt** activities restrict the access to a local system of all users on a specified remote system, including the manager of that remote system and individuals with superuser privileges (unless you have set up a special login and password for such users). Conversely, less restrictive permissions allow more generous access to all users on the specified remote system.

Warning: Entries in a **Permissions** file do not affect a remote-system user with a valid login on a specified local computer. Such users can connect to, log in on, and use a system regardless of the restrictions you set up in the local **Permissions** file. A user with a valid login is subject only to the permission codes established for that user's user ID (UID) and group ID (GID).

The **Permissions** file contains the following two types of entries:

The login IDs of remote systems that are allowed to communicate wit the local system, and the access permissions for those remote systems (information about sending and receiving files, and executing commands)

The names of those remote systems with which the local system ca communicate, and, again, the permissions that govern those remote systems' access to the local computer. (There must also be an entry for each of these remote computers in the **Systems** file on the local computer.)

The **Permissions** file includes very restrictive access permissions as the default values for sending and receiving files and executing commands. However, the file also provides options that enable you to change these defaults if you want to allow remote systems to have less restricted access to local systems.

Warning: The access permissions you set in this file affect all BNU communications, including those made through the mail facility and through a TCP/IP connection.

You must be logged in as the BNU system administrator, or as **su**, to access and edit the **Permissions** file, which is owned by the **uucp** program login ID.

For information about setting up a BNU system administrator login ID see "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3.

¦ Copyright IBM Corp. 1985, 1991

For information about the **uucp** program login ID, see "Setting Up a Systems File" in topic 6.8.5.2.3.

For more information about login IDs and passwords, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

Standard Entries in the Permissions File: Each entry in a **Permissions** file is a logical line. Each logical line is composed of the basic entry (a login ID or the name of a remote system) plus optional entries separated either by a space or a tab. Both the basic and the optional entries are composed of name/value pairs--that is, the name of the entry or option followed by an equal sign (=) followed by the value of the entry or option, with no spaces allowed within the pair. Comment lines begin with a pound sign (#) and occupy the entire physical line; blank lines are ignored.

The two types of entries in the **Permissions** file are for LOGNAMEs and MACHINEs.

LOGNAME This type of entry contains the login ID of, and access permissions for, a remote system that is allowed to conduct **uucico** and **uuxqt** transactions with a local system.

MACHINE This type of entry contains the names of and access permissions for the remote systems with which the local system is allowed to initiate **uucico** and **uuxqt** transactions.

LOGNAME entries are concerned with operations that occur when a remote system contacts a local system. The calling remote computer must be listed in the **Systems** file on the local computer.

MACHINE entries are concerned with operations that occur when a local system contacts a remote system, although the permissions in this entry still apply to the remote system's access to the calling local system.

A remote system listed in a MACHINE entry uses the login ID specified in a LOGNAME entry to communicate with a local system.

A LOGNAME Entry: A LOGNAME entry specifies one or more login IDs of remote systems that are permitted to log in to the local computer in order to conduct **uucico** and **uuxqt** transactions, and the access permissions for those remote systems. The actual login ID can be any name, although the examples in this chapter use a form of the **uucp** program login ID.

Note: Whatever login ID you choose **must** have both a user ID (UID) and a group ID (GID) of 5.

Following is an example of the simplest and most restrictive type of LOGNAME entry:

```
LOGNAME=uucp
```

In this example, the local system permits access only to a remote system that logs in using the **uucp** program ID. The entry does not contain any optional name/value pairs, which means that the remote system's access to the local system is restricted to the following default permissions:

The remote system may not ask to receive any queued files containin work that users on the local system have requested to be executed on

¦ Copyright IBM Corp. 1985, 1991

the calling remote system.

The local system may not send queued work to the calling remote syste when that system has completed its current operations. Instead, the queued work may be sent only when the local system contacts the remote system.

The remote system may not send files to (write) or transfer files fro (read) any location except the BNU public directory (**/usr/spool/uucppublic/system_name**) on the local system.

Users on the remote system may execute only the default commands o the local system. (The ***default command set*** includes only the **rmail** command, which users implicitly execute by issuing the **mail** command.)

Note: A name may appear in only one LOGNAME entry. If you have one entry for the **uucp** program login ID, that single entry is sufficient for all remote systems using that login ID. (You list these systems in the MACHINE entry.) You may have additional entries using other forms of the **uucp** program login ID such as **uucpa** or **uucp1**, as discussed below, but you may not include another **uucp** entry.

The example entry above uses the **uucp** program login ID, which is generally sufficient for **uucico** and **uuxqt** transactions between local and remote computers at most sites. However, you may include alternate versions of the **uucp** program login ID if certain computers at your site require different types of permissions when accessing the local system. For information about such login IDs, see "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

The following LOGNAME entry includes two login IDs used by remote systems (which are specified in a MACHINE entry) to access the local system **zeus**. Notice that both IDs use a form of the **uucp** program login, that they are separated by a colon, and that there are no spaces in the entry:

```
LOGNAME=uucp:uucp1
```

For additional information about login IDs and their associated passwords, see "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

A MACHINE Entry: The second type of standard entry in a **Permissions** file is the MACHINE entry. This contains the name of the local system, the names of the remote systems with which the local system is allowed to engage in **uucico** and **uuxqt** transactions, and, again, the access permissions for those remote systems. Following is the simplest kind of MACHINE entry:

```
MACHINE=zeus:hera
```

In this example, **zeus**, the local system, is permitted to communicate with **hera**, the remote system. Notice that the two system names are separated by a colon, and that the entry includes no spaces or tab characters. As was the case in the LOGNAME examples, there are no optional name/value pairs in this entry, indicating that the remote system's access to the local system is limited to the following:

The remote system may not ask to receive any local-system files queue to run on the calling remote system.

The local system may not access (read) any files except those in th

¦ Copyright IBM Corp. 1985, 1991

public directory on the local system.

The remote system may send (write) files only to the local publi directory.

The remote system may execute only those command in the defaul command set on the local system.

Like a LOGNAME entry, a MACHINE entry may also include a number of different remote systems:

```
MACHINE=zeus:hera:venus:merlin
```

Remember that you must include the name of the local system in the MACHINE entry. You may find it helpful to list that name first so that you do not forget it. Also, unless you explicitly specify additional options as described in "Options Used in the Permissions File," the default is for the most restrictive permissions in both the LOGNAME and the MACHINE entries.

Options Used in the Permissions File: The **Permissions** file includes a number of options that enable you to control access to local systems by remote systems involved in **uucico** and **uuxqt** transactions. The default permissions are restrictive, but you can change these defaults with one or more of the following options:

REQUEST	**COMMANDS**
SENDFILES	**VALIDATE**
READ, WRITE	**CALLBACK**
NOREAD, NOWRITE	**OTHER**

These options enable you to customize your **Permissions** file in such a way that different remote systems are allowed different types of access to the local system when using the BNU file transport and command execution programs.

The REQUEST Option

This option permits a remote system to ask to receive any queued files containing work that users on the local system have requested to be executed on that remote system. The default is not to allow such requests.

When a remote system contacts the local system in order to transfer files or execute commands, that remote system may also request permission to receive any files queued on the local system for transfer to or execution on that remote system. The following option permits such requests:

```
REQUEST=yes
```

The default, which you do not have to enter, is **REQUEST=no**. This specifies that the remote system cannot ask to receive any work queued for it on the local system. In this case, the local system must contact the remote system before files and execute commands queued on the local system can be transmitted to the remote system.

You can use the **REQUEST=yes** option in both LOGNAME and MACHINE entries in the **Permissions** file. Including this option makes it easy for remote-system users to transfer files to and execute commands on a local system. If security is a consideration at your site, you may prefer to

¦ Copyright IBM Corp. 1985, 1991

restrict this access so that the local system retains controls of file transfers and command executions initiated by remote systems.

Note: Remember, however, that entries in the **Permissions** file do not affect remote-system users with valid logins on a local system.

The SENDFILES Option

This option permits the local system to send queued work to the calling remote system after that remote system has completed its current **uucico** or **uuxqt** operations. The default is not to allow such transfers of queued local work.

When the remote computer has finished transferring files to or executing commands on a local system, that local system may try to send queued work to the calling remote system. This work generally includes files to be transferred to the remote system, or command requests waiting for execution on the remote system. The following option permits this type of operation:

```
SENDFILES=yes
```

This option, which you include only in a LOGNAME entry in a **Permissions** file, allows the transfer of queued work from the local to the remote system once the remote system has completed its operations.

The default value, which you do not have to enter, has the following form:

```
SENDFILES=call
```

This specifies that local files queued to run on the remote system will be sent only when the local system contacts the remote computer. As was the case with the **REQUEST** option, security considerations at your site may require that you limit a remote systems's access to a local system by using the default value for this option. (Remember that you do not have to enter default values.) Remember, too, that entries in the **Permissions** file do not affect remote-system users with valid logins on a local system.

The READ and WRITE Options

These options specify the path names of locations accessible to the **uucico** daemon when transferring files to or from the local system. The default location for both the **READ** and **WRITE** options is the BNU public directory on the local system:

```
READ=/usr/spool/uucppublic WRITE=/usr/spool/uucppublic
```

You can use the slash (/) that represents the **root** directory on the local system as the value part of the name/value pair in a **READ** or **WRITE** option. For example,

```
READ=/ WRITE=/
```

specifies that **uucico** (the daemon that transfers **uucp** and **uux** requests) can read from or write to any file on the local system under the **root** directory that allows access by a user with **other** permissions.

Note: The source or destination file or directory **must** be readable or writable or both to the **other** group. You set these permissions

¦ Copyright IBM Corp. 1985, 1991

with the **chmod** command. A user who is not logged in with **su** can take away permissions granted by the **READ** and **WRITE** options, but that user cannot grant permissions that are denied by these options.

You can specify more than one path for **uucico** activities, as in the following example:

```
WRITE=/usr/spool/uucppublic:/usr/news
```

This entry permits **uucico** to send files to both the BNU public directory and the **/usr/news** directory.

If you do not specify path names in the **READ** and **WRITE** options, BNU permits files to be transferred only to the **/usr/spool/uucppublic** directory. However, if you decide to specify path names in these options, you **must** enter the path name for every source and destination. For example, if you enter

```
WRITE=/usr/news
```

then the **uucico** daemon would be able to transfer files **only** to that directory. If you enter any path name in either option, you must also explicitly specify the public directory if you want **uucico** to be allowed to place files in that location.

You can include **READ** and **WRITE** options in both LOGNAME and MACHINE entries.

The NOREAD and NOWRITE Options

These options specify exceptions to the **READ** and **WRITE** options.

For example, the following entry,

```
READ=/ NOREAD=/etc WRITE=/usr/spool/uucppublic
```

permits the remote system to read any file on the local system **except** those in the **etc** directory and its subdirectories. The **WRITE** option in this example allows the remote system to transfer files only to the BNU public directory on the local system.

The **NOWRITE** option functions in exactly the same way as the **NOREAD** option--it explicitly specifies directories and files on the local system to which the remote system cannot transfer work.

Note: The specifications you enter with the **READ**, **WRITE**, **NOREAD**, and **NOWRITE** options can help determine the security of your local system in terms of **uucico** transactions.

The COMMANDS Option

This option, which you include only in a MACHINE entry, specifies the commands that remote systems listed in that MACHINE entry can execute on the local system.

Warning: The **COMMANDS** option can jeopardize the security of your system. Use it with extreme care.

The default for this option severely limits the commands that remote

¦ Copyright IBM Corp. 1985, 1991

systems can execute on the local system:

```
COMMANDS=rmail
```

This means that remote systems can run only the **rmail** command on the local system. (Remember that users enter the **mail** command, which then calls **rmail**.)

When you enter the **COMMANDS** option in the MACHINE part of an entry in the **Permissions** file, the commands you specify in that option override the default. For example, the entry

```
MACHINE=zeus:hera:venus:merlin COMMANDS=rmail:print
```

specifies that the remote systems **hera**, **venus**, and **merlin** can execute both the **rmail** (**mail**) and the **print** commands on **zeus**, the local system. These commands now make up the ***default command set*** for the remote systems listed in the MACHINE entry.

You can also specify path names to those locations on the local system where commands issued by users on remote systems are stored. For example,

```
COMMANDS=rmail:/bin/print
```

means that in addition to the **rmail** command, remote systems can also execute the **print** command, which is stored in the /**bin** directory. This option is useful when the default path of the **uuxqt** daemon does not include a particular directory where a permitted command resides.

Note: The default path of the **uuxqt** daemon includes only the **/bin** and **/usr/bin** directories.

If you want to allow a certain remote system to execute all the available AIX commands on the local system, enter the **COMMANDS** option with the value **ALL**:

```
COMMANDS=ALL
```

This specifies that the default command set available to the designated remote system (or for a particular login ID used by a remote system) includes all the available AIX commands.

The VALIDATE Option

This option provides a certain amount of security when you find it necessary to include commands in the default command set that could potentially cause damage when executed by a remote system on a local system. Use this option, which you specify only in a LOGNAME entry, in conjunction with a **COMMANDS** option that you specify in a relevant MACHINE entry.

The **VALIDATE** option verifies, to a certain degree, the identity of the calling remote computer. Including this option in a LOGNAME entry means that the calling remote system must have a unique login ID and password for file transfers and command executions.

Note: This option is meaningful only when the login ID and password are protected. Giving a remote system a special login and password that provide unlimited file access and remote command execution ability is equivalent to giving any user on that remote system a

¦ Copyright IBM Corp. 1985, 1991

normal login and password on the local system, unless the special login and password are well protected.

For example, the following entries,

```
LOGNAME=uucp VALIDATE=hera:venus:merlin

MACHINE=zeus:hera:venus:merlin COMMANDS=ALL
```

specify that if one of the remote systems **hera**, **venus**, or **merlin** attempts to log in to the local system, it must use the login ID **uucp** and the password associated with that login. Once the remote system is logged on, users on that remote system can then execute all AIX commands on the local system.

The **VALIDATE** option links a MACHINE entry, which includes a specified **COMMANDS** option, to a LOGNAME entry associated with a privileged login. BNU requires this validating link because the **uuxqt** daemon, which executes commands on the local system that have been requested by users on a remote system, isn't running while the remote system is logged in and therefore doesn't know which remote system sent the execution request.

Each remote system permitted to log in to a local system has its own spooling directory on that local system; only the BNU file transport and command execution programs are allowed to write to these directories. For example, when the **uucico** daemon transfers execution files from the remote system **hera** to the local system **zeus**, it places these files in the **/usr/spool/uucp/hera** directory on **zeus**.

Then, when the **uuxqt** daemon attempts to execute the specified commands, it determines the name of the calling remote system (**hera**) from the path name of the remote-system spooling directory (**/usr/spool/uucp/hera**). The daemon then checks for that name in a MACHINE entry in the **Permissions** file. The daemon also checks for the commands specified in the **COMMANDS** option in a MACHINE entry to determine whether the requested command may be executed on the local system.

The CALLBACK Option

This option, which you include only in LOGNAME entries, specifies that no **uucico** transactions will occur until the local system contacts the remote system that is attempting to establish a connection.

Following is the default **CALLBACK** option:

```
CALLBACK=no
```

This specifies that the remote system may contact the local system and begin transferring files without the local system initiating the **uucico** operations.

The following option,

```
CALLBACK=yes
```

specifies that the local system must contact the remote system before that remote system may transfer any files to the local system. The default value, **CALLBACK=no**, is generally sufficient for most sites.

Warning: If two systems both include the **CALLBACK=yes** option in their

¦ Copyright IBM Corp. 1985, 1991

respective **Permissions** files, they will never be able to communicate with each other.

The OTHER (System) Option

This option, which represents a system name in a MACHINE entry, enables you to set up access permissions for remote systems not explicitly specified in the existing MACHINE entries in a **Permissions** file.

This option is useful in the following circumstances:

When your installation includes a large number of remote systems tha regularly conduct **uucico** and **uuxqt** transactions with a local system

When it is occasionally necessary to change the default command se specified in the **COMMANDS** option in the MACHINE entry.

Rather than creating separate MACHINE entries for each of these numerous remote systems, you can set up one entry, with **OTHER** listed as the MACHINE, that includes the appropriate AIX commands specified in a **COMMANDS** option entry. Then, when it becomes necessary to change the default command set, you change the list of commands in only one entry rather than in numerous entries. You may also want to specify different (generally more restrictive) option values for these remote systems.

Following is an example of this type of entry:

```
LOGNAME=uucp1

MACHINE=OTHER COMMANDS=rmail:/bin/print:/usr/bin/nroff
```

This specifies that all remote systems using the **uucp1** login ID that are not included in existing MACHINE entries can execute the **rmail** (**mail**), **print**, and **nroff** commands on the local system.

Note: Notice that this example has very restricted access permissions. With the exception of the limited command set, both the LOGNAME and MACHINE entries use the default options, which restrict remote systems' **uucico** and **uuxqt** transactions with a local system. It's a good idea to restrict access permissions when using the **OTHER** option, although you can include any of the available MACHINE options.

Relationship between LOGNAME and MACHINE Entries: The following example shows the relationship between the LOGNAME and MACHINE entries in a **Permissions** file:

```
LOGNAME=uucp VALIDATE=hera REQUEST=yes SENDFILE=yes READ=/ WRITE=/

MACHINE=zeus:hera REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
```

The remote computer **hera** may engage in the following **uucico** and **uuxqt** transactions with the local system **zeus**:

The remote systems may request that files be sent from the loca system.

The local system may send files to the remote systems

The remote systems may execute all available AIX commands on the loca

¦ Copyright IBM Corp. 1985, 1991

system.

The remote systems may read from and write to all directories an files under the **root** directory.

Note: In this example entry, files owned by the **uucp** program login ID, such as the **Systems** file, are accessible by editing programs like **ed**. This means that a user on **hera** can both examine and modify the **Systems** file on **zeus** (if the AIX permission codes specify that the file is writable).

The example above obviously allows unrestricted access to the local system by the remote system listed in the MACHINE entry. If security is a concern at your site, you should probably set up this type of unrestricted LOGNAME/MACHINE entry on a local system **only** for a remote computer used by a system administrator or members of the **uucp** group.

Set up another LOGNAME/MACHINE entry in the local **Permissions** file for remote machines used by individuals who do not require unlimited access to that local system. Use another version of the **uucp** login ID in the LOGNAME entry. Then list the remote systems with restricted access in the MACHINE entry, and include only those commands that general users should execute on the local system.

For information about setting up BNU login IDs, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

Combining MACHINE and LOGNAME Entries: You can combine a LOGNAME and a MACHINE entry into one single entry when both parts include the same options.

For example, consider the following entries:

```
LOGNAME=uucp REQUEST=yes SENDFILE=yes READ=/ WRITE=/

MACHINE=zeus:hera REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
```

As you can see, both the LOGNAME and MACHINE entries include the same values for the REQUEST, READ, and WRITE options. You can, therefore, merge the two parts, as shown in the following example:

```
LOGNAME=uucp MACHINE=zeus:hera REQUEST=yes SENDFILE=yes COMMANDS=ALL \
READ=/ WRITE=/
```

Note: If the physical line representing an entry is too long to fit on the screen, make the last character in that physical line a backslash (\), which indicates continuation, and then type the remainder of the entry on the next line.

Sample Entries in a Permissions File: This section contains examples of several types of LOGNAME and MACHINE entries. You can use these entries as models for similar entries in your **Permissions** file.

1. The first LOGNAME entry specifies a restricted login for remote systems engaged in **uucico** and **uuxqt** transactions with a local system. This entry is intended for remote computers used by individuals who do not require complete access to the local system.

2. The second LOGNAME entry specifies a much less restricted remote-system login on a local system. Such an entry is intended for

¦ Copyright IBM Corp. 1985, 1991

a system administrator or members of the **uucp** group.

3. The MACHINE entries specify the remote systems with which a local system may communicate, and the access permissions granted to those remote systems.

   For an explanation of how to set up BNU login IDs and passwords, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

   For examples of complete BNU configuration files, se "Sample Configuration Files" in topic 6.8.5.2.9.

Setting Up a Restricted Login on a Local System: The first LOGNAME example illustrates an entry for a very restricted login on a local system. This entry specifies the most restricted access permissions for remote systems attempting to transfer files to and from, and execute commands on, a local system:

```
LOGNAME=uucp1
```

This entry specifies that the remote systems logging in on the local system must use the **uucp1** login and its associated password.

Note: Remember that you must set up this **uucp1** login and its associated password with the **users** command, as described in "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

This LOGNAME entry includes no options, so BNU uses the restrictive default options that permit the following remote-system access to the local system:

The remote system may not ask to receive any queued files containin work that users on the local system have requested to be executed on the calling remote system.

The local system may not send queued work to the calling remote syste when that system has completed its current operations. Instead, the queued work may be sent only when the local system contacts the remote system.

The remote system may send (write) files to and transfer (read) file from only the BNU public directory (**/usr/spool/uucppublic/system_name**) on the local system.

Users on the remote system may execute only the default commands o the local system. (The default command set includes only the **rmail** command, which users implicitly execute by issuing the **mail** command.)

This entry alone is sufficient to start **uucico** and **uuxqt** transactions between the local system and the remote systems specified in the relevant MACHINE entry. It permits a specified remote system to transfer files to the public directory on the local system.

Note: For information about setting up the MACHINE entry specifying the remote systems that use the **uucp1** login, see "Setting Up a Local System's Access to Remote Systems."

Setting Up an Unrestricted Remote Login on a Local System: The sample login in this section specifies a broad range of access permissions granted to remote systems logging in with the **uucp** program login ID.

¦ Copyright IBM Corp. 1985, 1991

Note: It is *not* a good idea to distribute this type of login and its associated password to the general users at your site. Logins of this kind are usually reserved for a tightly controlled group of computers that use a closely protected **Systems** file.

```
LOGNAME=uucp VALIDATE=hera:venus:merlin REQUEST=yes SENDFILES=yes \
READ=/ WRITE=/
```

This entry provides the following permissions when a remote system logs in using the **uucp** program login ID:

The calling remote system has a valid, unique login and password o the local system (the **VALIDATE** option).

The remote system may request that files be transferred to it from th local system (the **REQUEST** option).

Any requests queued on the local system for work on the remote syste will be sent to that remote system during the current session. These queued items are requests by local users for file transfers to and command executions on the remote system (the **SENDFILES** option).

The remote system may request any files that are readable by a use with **other** permissions, as specified with the AIX command **chmod** (the **READ** option).

The remote system may transfer files to any file or directory that i writable by a user with **other** permissions. This means that the file or directory is writable by a user with neither **owner** nor **group** permissions (the **WRITE** option).

The commands that the remote system can execute on the local system are specified in a relevant MACHINE entry.

Setting Up a Local System's Access to Remote Systems: The LOGNAME examples illustrate entries specifying the ways in which remote systems listed in the MACHINE part of an entry can conduct **uucico** and **uuxqt** transactions with a local system.

Following is an example of a MACHINE entry that specifies the remote systems with which a local system can communicate. This example is related to the **LOGNAME=uucp1** entry set up for computers with restricted access to the local system:

```
MACHINE=zeus:hera:venus:merlin
```

Local system **zeus** can communicate with the remote systems **hera**, **venus**, and **merlin**. Remember that you must include the name of the local system in a MACHINE entry.

Note: Remember too that the access permissions that govern **zeus**'s **uucico** and **uuxqt** transactions with the remote systems listed in this MACHINE entry are included in the LOGNAME/MACHINE entries in the **Permissions** files on **hera**, **venus**, and **merlin**.

Notice that there are no options specified in this entry. BNU therefore uses the restricted list of default options to specify the access permissions granted to the remote systems specified in this list:

¦ Copyright IBM Corp. 1985, 1991

The remote system may not ask to receive any queued files containin work that users on the local system have requested to be executed on the calling remote system.

The remote system may access (read) only those files that are in th public directory (**/usr/spool/uucppublic**) on the local system.

The remote system may send (write) files only to the public director on the local system.

Users on the remote system may execute only the default commands o the local system. The default command set includes only the **rmail** command (the user enters the **mail** command).

The next sample MACHINE entry accompanies the unrestricted remote-system access provided by the **LOGNAME=uucp** program login ID illustrated in "Setting Up an Unrestricted Remote Login on a Local System."

```
MACHINE=zeus:hera:venus:merlin REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
```

The options in this entry provide the following access to the remote systems **hera**, **venus**, and **merlin**:

The calling remote system may request that files be transferred to i from the local system (the **REQUEST** option).

The calling remote system may execute all AIX commands on the loca system (the **COMMANDS** option).

The calling remote system may request any files that are readable by user with **other** permissions, as specified with the AIX command **chmod** (the **READ** option).

The remote system may transfer files to any file or directory that i writable by a user with **other** permissions. This means that the file or directory is writable by a user with neither **owner** nor **group** permissions (the **WRITE** option).

For additional examples of entries in a **Permissions** file, see "Sample Configuration Files" in topic 6.8.5.2.9.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.6 Customizing the Poll File

The **Poll** file (**/usr/adm/uucp/Poll**) contains information specifying when BNU should poll designated remote computers. This file is used in conjunction with the **/usr/spool/cron/crontabs/uucp** file, the **uudemon.hour** script, and the **uudemon.poll** script. Together, these files are responsible for initiating automatic calls to remote systems to perform certain maintenance tasks.

Each entry in the **Poll** file contains the name of the remote computer followed by a tab character and a sequence of times to poll. You must specify times as digits between 0 and 23.

Following is a standard entry in the **Poll** file:

```
  hera <TAB> 0  4  8  12  16  20
```

This entry specifies that the local system will poll the remote system **hera** every four hours. Depending on how the systems at your site are used, you may need to modify the times specified in the **Poll** file.

You must be logged in as the BNU administrator (**bnuadm** is the example login ID), or as **su**, to edit the **Poll** file, which is owned by the **uucp** program login ID.

Note: You may have trouble modifying the **Poll** file with the INed editor, which expands tab characters.

For information about setting up an administrator's login, se "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3.

For information about the **uucp** program login ID, see "Setting Up a Systems File" in topic 6.8.5.2.3.

For more information about login IDs and passwords, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.7 Customizing the remote.unknown File

BNU executes the shell script **/usr/adm/uucp/remote.unknown** when a remote computer that is not listed in the local **Systems** file attempts to communicate with that local system. BNU does not permit the unknown remote system to connect with the local system.

Instead, the **remote.unknown** script appends the following type of entry to the file **/usr/adm/uucp/Admin/Foreign**:

```
FOREIGN=/usr/spool/uucp/.Admin/Foreign
echo "¦date¦:call from the system $1"<<FOREIGN
```

You can modify this file to fit the needs of your site. You must be logged in as the BNU administrator (**bnuadm** is the example login ID), or as **su**, to edit the **remote.unknown** file, which is owned by the **uucp** program login ID.

> For information about setting up an administrator's login, se "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3.
>
> For information about the **uucp** program login ID, see "Setting Up a Systems File" in topic 6.8.5.2.3.
>
> For more information about login IDs and passwords, se "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.8 Setting Up BNU Login IDs and Passwords

As you know, **uucp** is the login ID for the **uucp** file transport programs; the encrypted password for this login is included in the **/etc/passwd** file. All the example entries in "Customizing the Systems File" in topic 6.8.5.2.3 and most of the examples in "Customizing the Permissions File" in topic 6.8.5.2.5 use the **uucp** program login ID, and the **Systems** examples use the sample password **passuucp**.

If your site is relatively small and security is not a major concern, you will probably need only one version of the **uucp** program login ID and its associated password. In that case, go on to "Creating BNU Login IDs and Passwords" and follow the instructions in that section.

However, conditions at your site may require that certain remote computers have different types of access to a specific local system. You can meet this need by creating an alternate version of the **uucp** program login ID with its associated password in the **Systems** file on the local computer.

You then include both the regular **uucp** login ID and the alternate version in LOGNAME entries in the local system's **Permissions** file. These entries give the two remote-system login IDs different access to the local system for **uucico** and **uuxqt** operations.

Note: The practice of creating additional, more restrictive **uucp** login IDs is a good idea if security is a consideration at your site.

You should include one entry in the local **Systems** file intended only for the BNU system administrator (or members of the **uucp** group). You then set up an unrestricted **uucp** LOGNAME entry in the local system's **Permissions** file intended only for that administrator. Together, the entry in the **Systems** file and the entry in the **Permissions** file give the system administrator, who is working on the remote computer specified in the MACHINE entry, unrestricted access to the local system for **uucico** and **uuxqt** operations.

In addition to the unrestricted entry in the local **Systems** file, you can also set up an alternate **uucp** login in that file intended for users of remote systems whose access to the local system should be restricted. You then set up a LOGNAME entry in the local **Permissions** file that limits **uucico** and **uuxqt** activities on the local system. These activities have been requested by users working on the remote system identified with the alternate **uucp** login.

> For examples of such LOGNAME and MACHINE entries, se "Entries in the Systems and Permissions Files on a Local System" and "Entries in the Systems and Permissions Files on the Remote Systems."
>
> For additional examples of LOGNAME and MACHINE entries, se "Sample Configuration Files" in topic 6.8.5.2.9.

Creating BNU Login IDs and Passwords: Following are instructions for creating a **uucp** login ID and its associated password. You issue the AIX command **users** for this task, and then follow the instructions on the screen.

If you are including two versions of the **uucp** program login ID in a **Permissions** file, first issue **users** and set up the unrestricted login and password. Then issue the command again and set up the restricted login and its associated password.

¦ Copyright IBM Corp. 1985, 1991

Note: The instructions below are very similar to those given in "Setting Up the BNU Manager's Login and Password" in topic 6.8.5.1.3 for setting up a BNU system administrator login, with a few exceptions. You must be a member of the **system** group, or be operating with superuser authority, to execute the **users** command.

1. Enter the **users** command at the superuser prompt (#).

2. Enter the **add** subcommand to add the appropriate version of the **uucp** login ID:

   ```
   > a u uucp
   ```

   or

   ```
   > a u uucp1
   ```

3. Answer **n** to the **OK? (y)** prompt and then enter the following in response to the **Field?** prompt:

   ```
   Field? uid
   ```

4. Now enter **5** in response to the prompt for the user ID:

   ```
   uid 5
   ```

5. Enter **group** in response to the next prompt:

   ```
   Field? gr
   ```

6. In response to the **group** prompt, enter the following:

   ```
   group uucp
   ```

7. Answer **n** to the **OK?** prompt and then enter the following to set up a password for the new login ID:

   ```
   Field? pa
   ```

8. Enter the password for the login ID that you specified in step 2 in response to the **password** prompt.

9. The directory field should contain the following entry:

   ```
   /usr/spool/uucppublic
   ```

   If it does not, enter **dir** in response to the **Field?** prompt, and then enter the name of the BNU public directory in response to the prompt for a directory name.

10. The program field should contain the following entry:

    ```
    /usr/lib/uucp/uucico
    ```

    If it doesn't, enter that program name in response to the prompt.

11. To end the procedure, press **Enter** at the next **Field?** prompt. The **users** command displays the information for the new login ID (the password is encrypted).

¦ Copyright IBM Corp. 1985, 1991

12. If the information is correct, press **Enter** at the **OK? (y)** prompt and then press **Enter** again at the prompt **Standard new user initialization? (y)**. If the information is not yet correct, answer **n** to the **OK?** prompt and then re-enter the data.

13. Enter the **q** command to exit from the **users** program.

The system adds the new **uucp** login ID and password to the list of users in the **/etc/passwd** file and the login ID to the **uucp::5:** entry in the **/etc/group** file.

Entries in the Systems and Permissions Files on a Local System: Following are example entries in the **Systems** file on the local system **zeus** for communications with the remote systems **hera**, **venus**, and **merlin**.

System **zeus** is used only by the system manager. No other user at the site has a valid login on this computer. The login ID this system uses to initiate **uucico** and **uuxqt** activities on remote computers is **uucp**, and the associated password is **alluucp**.

System **hera** is used only by members of the **system** group. Regular users cannot log in on this computer. System **hera** also uses the **uucp** login ID to initiate file-transfer activities, but the associated password is **sysuucp**.

Both systems **venus** and **merlin** are available to all users working at the site. The login ID for these systems is **uucp1**, and the associated password is **passuucp**.

1. Following are the relevant entries in the **Systems** file on system **zeus**:

```
hera Any hera 1200 - "" \r\d\r\d\r in:--in: uucp word: sysuucp

venus Any venus 1200 - "" \r\d\r\d\r in:--in: uucp1 word: passuucp

merlin Any merlin 1200 - "" \r\d\r\d\r in:--in: uucp1 word: passuucp
```

Zeus can communicate with the three remote systems at any time. The local system sends the strings **uucp** and **sysuucp** in response to **hera**'s login and password prompts, and the strings **uucp1** and **passuucp** in response to the login and password prompts of **venus** and **merlin**.

2. **Zeus** is now configured to communicate with each of the other systems. However, before users on **hera**, **venus**, and **merlin** can log in to and use **zeus** for **uucico** and **uuxqt** activities, the following entries must exist in the **Permissions** file on **zeus**:

```
LOGNAME=uucp REQUEST=yes SENDFILES=yes VALIDATE=hera READ=/ WRITE=/
MACHINE=zeus:hera REQUEST=yes COMMANDS=ALL READ=/ WRITE=/

LOGNAME=uucp1
MACHINE=zeus:venus:merlin
```

The first LOGNAME/MACHINE entry allows system **hera**, used only by the **system** group, free access to the files on **zeus**. The second LOGNAME/MACHINE entry allows systems **venus** and **merlin** only to send files to **zeus**.

¦ Copyright IBM Corp. 1985, 1991

Entries in the Systems and Permissions Files on the Remote Systems:

1. The **Systems** file on the remote system **hera** contains the following entries for communicating with **zeus**, **venus**, and **merlin**:

```
zeus Any zeus 1200 - "" \r\d\r\d\r in:--in: uucp word: alluucp

venus Any venus 1200 - "" \r\d\r\d\r in:--in: uucp1 word: passuucp

merlin Any merlin 1200 - "" \r\d\r\d\r in:--in: uucp1 word: passuucp
```

   Like **zeus**, system **hera** can communicate with the remote systems at any time. In response to **zeus**'s login and password prompts, **hera** sends the strings **uucp** and **alluucp**, and the strings **uucp1** and **passuucp** in response to the prompts from **venus** and **merlin**.

2. Following are the entries for the three remote systems in the **Systems** file on **venus**:

```
zeus 0800-1700 zeus 1200 - "" \r\d\r\d\r in:--in: uucp word: alluucp

hera 0800-1700 hera 1200 - "" \r\d\r\d\r in:--in: uucp word: sysuucp

merlin Any merlin 1200 - "" \r\d\r\d\r in:--in: uucp1 word: passuucp
```

3. The entries in the **Systems** file on **merlin** are almost exactly like those on **venus**:

```
zeus 0800-1700 zeus 1200 - "" \r\d\r\d\r in:--in: uucp word: alluucp

hera 0800-1700 hera 1200 - "" \r\d\r\d\r in:--in: uucp word: sysuucp

venus Any venus 1200 - "" \r\d\r\d\r in:--in: uucp1 word: passuucp
```

4. Following are the relevant entries in the **Permissions** file on **hera** for communication with the three other systems:

```
LOGNAME=uucp REQUEST=yes SENDFILES=yes VALIDATE=zeus READ=/ WRITE=/
MACHINE=hera:zeus REQUEST=yes COMMANDS=ALL READ=/ WRITE=/

LOGNAME=uucp1
MACHINE=hera:venus:merlin
```

   The first LOGNAME/MACHINE entry allows **zeus** unlimited access to **hera**. The second entry allows **venus** and **merlin** only to send files to **hera**.

5. The entries in the **Permissions** file on **venus** contain the following permissions:

```
LOGNAME=uucp REQUEST=yes SENDFILES=yes VALIDATE=zeus:hera READ=/ WRITE=/
MACHINE=venus:zeus:hera REQUEST=yes COMMANDS=ALL READ=/ WRITE=/

LOGNAME=uucp1 REQUEST=yes SENDFILES=yes VALIDATE=merlin READ=/ WRITE=/
MACHINE=venus:merlin REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
```

6. Again, the entries in the **Permissions** file on **merlin** are almost exactly like those on **venus**:

```
LOGNAME=uucp REQUEST=yes SENDFILES=yes VALIDATE=zeus:hera READ=/ WRITE=/
```

¦ Copyright IBM Corp. 1985, 1991

```
MACHINE=merlin:zeus:hera REQUEST=yes COMMANDS=ALL READ=/ WRITE=/

LOGNAME=uucp1 REQUEST=yes SENDFILES=yes VALIDATE=merlin READ=/ WRITE=/
MACHINE=merlin:venus REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
```

For additional examples of entries in a **Permissions** file, see "Sample Configuration Files" in topic 6.8.5.2.9.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.9 Sample Configuration Files

This section contains two sets of sample configuration files. Each example illustrates the entries in the relevant files that you need to include on both the local and the remote systems in order for the two computers to communicate using BNU.

Configuration Files for a Hardwired Connection: These files are set up for a hardwired connection between systems **zeus** and **hera**, where **zeus** is considered the local system and **hera** is considered the remote system. The hardwired device is **tty0**. The login ID for **zeus** is **uucp**, and the associated password is **alluucp**. The login ID for **hera** is also **uucp**, but the associated password is **sysuucp**.

Entries in the Local System's Files for a Hardwired Connection:

1. The **Devices** file on **zeus** contains the following entry in order to connect to the remote system **hera**:

```
Direct  tty0  -  1200  direct
hera    tty0  -  1200  direct
```

2. The **Systems** file on **zeus** contains the following entry for the remote system **hera**:

```
hera Any hera 1200 - "" \r\d\r\d\r in:--in: uucp word: sysuucp
```

3. The **Permissions** file on the local system **zeus** contains the following entries specifying the ways in which the remote system **hera** can conduct **uucico** and **uuxqt** transactions with **zeus**:

```
LOGNAME=uucp VALIDATE=hera REQUEST=yes SENDFILES=yes READ=/ WRITE=/
MACHINE=zeus:hera REQUEST=yes SENDFILES=yes COMMANDS=ALL READ=/ WRITE=/
```

Entries in the Remote System's Files for a Hardwired Connection:

1. The **Devices** file on **hera** contains the following entry for communication with **zeus**:

```
Direct  tty0  -  1200  direct
zeus    tty0  -  1200  direct
```

2. The **Systems** file on **hera** contains the following entry for **zeus**:

```
zeus Any zeus 1200 - "" \r\d\r\d\r in:--in: uucp word: alluucp
```

3. The **Permissions** file on **hera** contains the following entries specifying the ways in which **zeus** can conduct **uucico** and **uuxqt** transactions with **hera**:

```
LOGNAME=uucp VALIDATE=zeus REQUEST=yes SENDFILES=yes READ=/ WRITE=/
MACHINE=hera:zeus REQUEST=yes SENDFILES=yes COMMANDS=ALL READ=/ WRITE=/
```

Configuration Files for a Telephone Connection: These files are set up to connect systems **venus** and **merlin** over a telephone line using modems. **Venus** is considered the local system, and **merlin** is considered the remote system.

On both systems, the device **tty1** is hooked to a Hayes modem at 1200 baud.

¦ Copyright IBM Corp. 1985, 1991

The login ID for both systems is **uucp1**, and the associated password is **passuucp**. The phone number for the modem attached to **venus** is **9=3251436**; the number of the **merlin** modem is **9=4458784**. Both computers include partial phone numbers in their **Systems** files and dialing codes in their **Dialcodes** files.

Entries in the Local System's Files for a Telephone Connection:

1. The **Devices** file on **venus** contains the following entry for the connection to **merlin**:

   ```
   ACU  tty1  -  1200  hayes
   ```

2. The **Dialers** file on **venus** contains the following entry for its modem:

   ```
   hayes  =,-,  ""  \dAT\r\c OK \pATDT\T\r\c CONNECT
   ```

3. The **Systems** file on **venus** contains the following entry for **merlin**, including a phone number and a dialing prefix:

   ```
   merlin Any ACU 1200 local8784 "" \r\d\r\d\r in:--in: uucp1 word: passuuc
   ```

4. The **Dialcodes** file on **venus** contains the following dialing code prefix for use with the number in the **Systems** file:

   ```
   local 9=445
   ```

5. The **Permissions** file on **venus** contains the following entries specifying the ways in which **merlin** can conduct **uucico** and **uuxqt** transactions with **venus**:

   ```
   LOGNAME=uucp1 VALIDATE=merlin REQUEST=yes SENDFILES=yes READ=/ WRITE=/
   MACHINE=venus:merlin REQUEST=yes SENDFILES=yes COMMANDS=ALL READ=/ WRITE
   ```

Entries in the Remote System's Files for a Telephone Connection:

1. The **Devices** file on **merlin** contains the following entry for the connection to **venus**:

   ```
   ACU  tty1  -  1200  hayes
   ```

2. The **Dialers** file on **merlin** contains the following entry for its modem:

   ```
   hayes  =,-,  ""  \dAT\r\c OK \pATDT\T\r\c CONNECT
   ```

3. The **Systems** file on **merlin** contains the following entry for **venus**, including a phone number and a dialing prefix:

   ```
   venus Any ACU 1200 local436 "" \r\d\r\d\r in:--in: uucp1 word: passuucp
   ```

4. The **Dialcodes** file on **merlin** contains the following dialing code prefix for use with the number in the **Systems** file:

   ```
   local 9=325
   ```

5. The **Permissions** file on **merlin** contains the following entries specifying the ways in which **venus** can conduct **uucico** and **uuxqt** transactions with **merlin**:

   ```
   LOGNAME=uucp1 VALIDATE=venus REQUEST=yes SENDFILES=yes READ=/ WRITE=/
   ```

¦ Copyright IBM Corp. 1985, 1991

```
      MACHINE=merlin:venus REQUEST=yes SENDFILES=yes COMMAND=ALL READ=/ WRITE=
```

For information about the hardware used in BNU communications, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.10 Checking for Correct Permissions

Now that the BNU files are customized for your site, you should issue the **uucheck** command again to check for possible errors in the **Permissions** file. However, **uucheck** does **not** check file/directory modes, nor does it check for duplicate login or machine names.

Issue **uucheck** with the **-v** flag, which provides a detailed explanation of the way in which BNU interprets the **Permissions** file:

```
uucheck -v
```

If **uucheck** displays an error message, use the **pg** command to examine the **Permissions** file and make sure the entries are correct. Then issue **uucheck** again.

When the shell prompt returns to the screen, continue with the next section.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.2.11 Checking Networked Systems (uuname)

Before you continue, it's a good idea to check to see that all the systems included in the **Systems** file on the local system are actually on the BNU network. Use the **uuname** command for this task.

```
uuname
```

If all the systems are networked correctly, each system name will appear on the list displayed on the screen. The systems on this list are also the systems to which users can send mail.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.3 Setting Up BNU for Use with TCF

With the Transparent Computing Facility (TCF), a cluster of machines can be configured to act as a single Unix-to-Unix Copy Program (UUCP) node. Multiple machines within the cluster can provide UUCP (BNU) communication facilities.

In order to enable full TCF function for the Basic Networking Utility (BNU), each site in the cluster must have an accompanying directory named **<LOCAL> /spool/uucp**. For example, if a cluster consisted of the following three sites, **clubs, spaces** and **hearts**, there would be three accompanying directories:

1. **/clubs/spool/uucp**
2. **/spades/spool/uucp**
3. **/hearts/spool/uuc/spades/spool/uucp**

If the site named **spades** is the machine that physically possesses the BNU communication facility (a modem, for example), then for maximum efficiency designate the directory **/spades/spool/uucp** as the master spool directory for this cluster in the file **/usr/adm/uucp/Spools** (see below). The directories **/clubs/spool/uucp** and **/hearts/spool/uucp** are local spool directories, used as backups to the master spool directory in the event that the master spool directory is full or unavailable.

If the site named **spades** goes down, BNU transmission requests originating on the remaining sites in the cluster **hearts and clubs** cause the files to be spooled to the local directories. When spades comes back up, the **uncollect** program must be started to collect all files from the local spool directories and put them in the master spool directory for transmission. The usual way **uucollect** is started is to run it from the **uucp cron** job just before polling remote sites.

To set up BNU to run in a TCF cluster, you must perform the following steps:

1. Select a site within the cluster which will have the modem or other communication facility attached. This will be the master site.

2. Update the UUCP configuration files in /usr/adm/uucp in the same way that you would if the site were not a member of a TCF cluster. These configuration files are common and hence shared by all sites in the cluster.

3. Update the Spools file to identify one site as the master and the other sites as local sites. Note that for each remote system, there can be one or more master sites, each with its list of local sites. The usual usage is to designate a single site the master and all other sites local sites. To specify multiple master sites, use multiple lines in the Spools file with the same remote node name.

4. Create or update the file **/usr/adm/uucp/Myname** with the name of your cluster.

Subtopics
6.8.5.3.1 Updating and Setting the Spools File
6.8.5.3.2 Administering the BNU on a TCF Cluster

¦ Copyright IBM Corp. 1985, 1991

6.8.5.3.1 Updating and Setting the Spools File

To update the **Spools** file, you should use a text editor to edit the file **/usr/adm/uucp/Spools**. After editing the file, make sure this file is stored on all sites in the cluster by issuing the command:

```
chfstore all /usr/adm/uucp/Spools
```

The Spools file should consist of one or more lines, each using one of the following two formats:

```
remote master [local...]
remote master all
```

where the fields have the following meanings:

remote This is the name given to the remote uucp node.

master This is the name of the site within the TCF cluster that you designate as containing the master spool directory for this remote node. It is recommended that this be the name of the site in the cluster which you have designated as the node to contact or be contacted by the remote uucp node.

local... These are optional sites in the cluster which you permit to temporarily spool files queued to be delivered to the remote node. Usually, you will place the word "all" here to indicate all sites in the cluster may serve as local spool sites if the master spool directory is full or unavailable. Use an explicit list of local sites, however, if you designate two master sites for the same remote uucp node and you want to selectively associate certain local sites with specific master sites. If a site in the TCF cluster is not named as a master or local site for a remote uucp node, the files to be delivered to that remote node may go undelivered indefinitely.

Note: If a particular site is not included, either explicitly or implicitly by using **all**, as a local site, then any jobs queued locally on that site will be ignored by **uucollect** and will not be moved to the master spool site for transmission.

Every hour, on each master site the **uucollect** program will be started by the script **uudemon.hour**. (This script is run from the uucp **crontab** file). The **uucollect** command will scan the **Spools** file looking for entries where the master site indication matches the site where uucollect is running. It then searches the **<LOCAL>/spool/uucp** directory on each of the indicated local sites, moving to the master spool directory all files that are to be delivered to the indicated remote node.

Here is the contents of a typical **Spools** file (sample data):

```
ucla-cs       spades all
trwrb  spades clubs
laidbak       spades clubs hearts
```

In this example, **spades** communicates with the remote sites **ucla-cs**, **trwrb** and **laidbak**. If all three of the cluster sites are up and running and the spool directories are not in full or read-only file systems, then each outgoing uucp request from any site in the cluster is spooled in the directory **/spades/spool/uucp**, since, in this example, **spades** is the master

¦ Copyright IBM Corp. 1985, 1991

spool site in all cases.

If however, an outgoing file is initiated from hearts or clubs and the file cannot be spooled in **/spades/spool/uucp** for some reason, the fill will be spooled in **/hearts/spool/uucp** or **/clubs/spool/uucp**, respectively. When the **uucollect** program runs on spades, **/clubs/spool/uucp** is searched for files to be delivered to **ucla-cs**, **trwrb** or **laidbak** and **/hearts/spool/uucp** is searched for files to be delivered to **ucla-cs** or **laidbak**. Files in **/hearts/spool/uucp** which are to be delivered in **trwrb** will not be copied to **spades/spool/uucp** because hearts has been inadvertently not listed as a local site for **trwrb**. The **trwrb** entry, therefore, is an example of an incorrect **Spools** file **entry** that should be corrected to look like the **laidbak** entry below it.

Using a device attached to another TCF cluster site is an easy way to temporarily switch to use a different line, should there be problems with the regular line. Also, this feature can be used to allow an AIX/370 machine with a large spool directory to make use of a modem attached to a serial line of an AIX PS/2 workstation.

The **/usr/adm/uucp/Myname** file should be set up so that it contains, on a single line, the UUCP name for the cluster. This is the name by which remote systems know the cluster. The sites (**spades**, **clubs** and **hearts** in this example) are not used by BNU, nor are they known to remote systems. This file (the **Myname** file) must be created with a text editor when BNU is being configured to run in a TCF environment. Choose a name that represents the cluster as a whole (for example, **cards**).

In a TCF cluster, the file **/usr/adm/uucp/devices** can contain a full pathname in the line and caller fields (fields 2 and 3). A standard **devices** line could look like:

```
ACU  tty  -  2400  hayes  \D
```

where the line field, **tty**, indicates that the line is on **/dev/tty0** on the local site. The line field can also have a full pathname to the site to which the modem is attached, for example:

```
ACU  /spades/dev/tty00  -  2400  hayes  \D
```

¦ Copyright IBM Corp. 1985, 1991

6.8.5.3.2 Administering the BNU on a TCF Cluster

To administer the BNU on a Transparent Computing Facility (TCF) cluster, follow the guidelines below:

Each site in the cluster must have a complete spool structure tha consists of:

<LOCAL>/spool/uucp/.Admin
<LOCAL>/spool/uucp/.Corrupt
<LOCAL>/spool/uucp/.Log
<LOCAL>/spool/uucp/.Log/uucp
<LOCAL>/spool/uucp/.Log/uux
<LOCAL>/spool/uucp/.Log/uucico
<LOCAL>/spool/uucp/.Log/uuxqt
<LOCAL>/spool/uucp/.Old
<LOCAL>/spool/uucp/.Sequence
<LOCAL>/spool/uucp/.Status
<LOCAL>/spool/uucp/.Workspace
<LOCAL>/spool/uucp/.Xqtdir
<LOCAL>/spool/uucppublic

/usr/spool must be a symbolic link to **<LOCAL>/spool**.

/etc/locks must be a symbolic link to **<LOCAL>/locks**, which must exist.

For efficiency, the master spool for a particular remote node shoul be on the site responsible for conversing with that remote node (as defined in **/usr/adm/uucp/Spools**).

The **uucollect** program should be started by **uudemon.hour** before invoking **uusched**. The **uucollect** program only needs to be run on sites with master spool directories.

uudemon.poll and **uudemon.hour** should only be started on sites that have master spools.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.4 Setting Up BNU for Use with TCP/IP

If your site is using Transmission Control Protocol/Internet Protocol (TCP/IP), you will need to perform some additional tasks in order for BNU and TCP/IP to communicate. You must enable **inetd** so it can start the **uucpd** daemon, and you must include the relevant TCP/IP entries in the **Devices**, **Systems**, and **Permissions** files.

The **uucpd** daemon handles communication between BNU and TCP/IP. This daemon is an internal program that enables users on systems linked by BNU to establish a TCP/IP connection to other systems linked over a Token Ring or Ethernet network.

Following are the steps you need to perform to enable BNU and TCP/IP to communicate:

1. Uncomment the following line in **/etc/inetd.conf**:

   ```
   uucp stream tcp nowait root /etc/uucpd uucpd
   ```

2. Check the **/etc/services** file to ensure that it includes the following line:

   ```
   uucp 450/tcp uucpd
   ```

3. Enter:

   ```
   kill -HUP pid
   ```

   where **pid** is the process ID. This step causes **inetd** to reread the configuration file **inetd.conf** by issuing a hangup signal to the **inetd** process.

4. Check to make sure that the **/etc/hosts** file contains an entry for the remote computer in the TCF cluster to which you want to connect.

   Entries in the **/etc/hosts** file look similar to the following:

   ```
   130.200.8.10     regulus       QAnet
   ```

5. Update the Systems, Devices, and Permissions files in the **/usr/adm/uucp** directory to include the relevant TCP/IP entries.

Even though the **uucpd** daemon is running, you must still add the relevant TCP/IP entries to the BNU configuration files. In order to make the appropriate entries in the **Systems** file, you must decide on the appropriate TCP/IP conversation protocol to enter in the **TCP** caller subfield.

There are three kinds of protocols: **g**, **t**, and **e**.

Protocol	**Explanation**
g	This is the default. The **g** protocol is useful for modem connections, but it involves a large overhead in running BNU commands.
t	The **t** protocol presumes an error-free channel, and is essentially the **g** protocol without the checksumming and

¦ Copyright IBM Corp. 1985, 1991

packetizing. You can use the **t** protocol for two types of communication:

To communicate with a site running the AIX version of BNU

To communicate with a site running the Berkeley version of the UNIX-to-UNIX Copy Program (UUCP).

The **t** protocol is not reliable for modem connections.

e You can also use the **e** protocol for two types of communication:

To communicate with a site running the AIX version of BNU

To communicate with a site running a non-AIX version of BNU.

The **e** protocol is not reliable for modem connections.

If you have not already done so, add the appropriate entries to the BNU configuration files.

Enter the following in the **Devices** file:

```
TCP  -  -  -  TCP  \D
```

Specify **TCP** in the caller field. Enter hyphens in the line, line2, and class fields. Enter **TCP** as the dialer and either **\D** or **\T** as the token. Remember that the **\D** string instructs BNU to use the telephone number in the **Systems** file, but not to expand that number by including the dial code abbreviation specified in the **Dialcodes** file. If you want to expand the number with the dialing code abbreviation, use the characters **\T** instead of **\D**.

Enter the following in the **Systems** file to use TCP/IP to connect to system **venus** with the default **g** protocol:

```
venus  Any  TCP  -  -  in:--in: uucp1  word: passuucp
```

Replace the send and expect characters in the sample login field with the login prompt, login, password prompt, and password appropriate to the remote system for which you are establishing a connection.

Enter the following in the **Systems** file to use TCP/IP with the **t** protocol:

```
venus  Any  TCP,t  -  -  in:--in: uucp1  word: passuucp
```

Again, replace the strings in the login field with the characters appropriate for your systems.

Enter the following in the **Systems** file to use TCP/IP with the **e** protocol:

```
venus  Any  TCP,e  -  -  in:--in: uucp1  word: passuucp
```

Again, replace the strings in the login field with the characters appropriate for your systems.

¦ Copyright IBM Corp. 1985, 1991

Enter the following in the **Systems** file to use TCP/IP with a program login ID other than **uucp**:

```
opus  Any  TCP,t  -  -  in:--in: commo  word: bloom
```

Enter the appropriate LOGNAME and MACHINE entries in the **Permissions** file:

```
LOGNAME=uucpo
MACHINE=zeus:venus:opus
```

Depending on the remote systems that are connecting to the local system over the TCP/IP connection, use either the strict default permissions, or enter the options appropriate to your site.

Remember that you must set up an appropriate login ID and password for any remote system that will initiate **uucico** and **uuxqt** activities. For information about this task, see "Setting Up BNU Login IDs and Passwords" in topic 6.8.5.2.8.

¦ Copyright IBM Corp. 1985, 1991

6.8.5.5 Setting up a TCP/IP Connection with a Remote TCF Cluster

If you want to establish a BNU connection with a remote TCF cluster via TCP/IP, the name by which that cluster is addressed must be recorded on your own site. Therefore, the following additional steps are necessary:

1. Contact the system administrator for the remote cluster to obtain the following:

   The name by which that cluster is to be addressed (found in the **/usr/adm/uucp/Myname** file)

   The IP address of the machine that runs the **uucp** daemon, **uucpd** for that cluster.

2. Edit the **/etc/hosts** file, adding the remote cluster's name. Each line has the following general format:

   ```
   IP-address    site-name    alias    alias
   ```

   If the remote cluster's name is **Dragon** and the IP address is **130.200.4.1**, this line would look like:

   ```
   130.200.4.1    Dragon     Thunderlizard
   ```

3. Edit the **/usr/adm/uucp/Systems** file, adding the remote cluster's name. For a remote cluster whose name is **Dragon**, this line would look like:

   ```
   Dragon   AnyTCP,t--in:--in: uucp word: Open-Sesame
   ```

4. Create a machine entry in the **/usr/adm/uucp/Permissions** file.

¦ Copyright IBM Corp. 1985, 1991

6.8.6 Performing Routine Maintenance Tasks

Once BNU is installed and configured to run at your site, you will have to perform some maintenance operations to ensure that the networking utilities run properly. Managing a small installation is relatively easy; managing a large BNU site is a little more difficult for several reasons, including the following:

If there is a problem that requires making a change in one of the BN configuration files, you must check and, if necessary, update all related entries in all relevant files on all the systems connected through BNU.

Users, who are impatient with any system failure, are generally eve less patient when failures occur between two systems that are under local control.

In general, the major problem in a network such as BNU is dealing with the backlog of file-transfer and command-execution requests that, for whatever reason, cannot be transmitted to the specified system. In order to minimize problems of this type, you should perform the tasks described in this section as part of your basic system maintenance procedures.

This section describes the following maintenance tasks:

Monitoring remote connections and file transfer

Cleaning up spooling directorie

Working with log files

You must perform some parts of these tasks manually; BNU performs other parts automatically. The BNU software also contains several daemons and shell scripts that automatically execute certain maintenance operations, as described in "Using the Daemons" in topic 6.8.7 and "Running Automatic Maintenance Routines" in topic 6.8.8.

The **cron** daemon and its supporting command, **crontab**, are very useful tools for managing AIX systems.

The **crontab** file (**/usr/spool/cron/crontabs/uucp**) contains various commands useful for managing BNU. When the local system is run in state 2 (multi-user), **cron** runs the commands in **/usr/spool/cron/crontabs/uucp** that are scheduled to be run at that time. You can instruct **cron** to do this as often as you want.

Initially, the commands in **/usr/spool/cron/crontabs/uucp** are commented out (these lines begin with the comment character (#) and, therefore, are not executed by **cron**). Changes made to this file do not take effect unless the changed file is installed or the system is restarted. Therefore, to uncomment certain lines in this file, use the **crontab** command while logged in as the BNU Administrator (**uucp** is the example login ID) to get a copy of the file, edit the copy, and then use the **crontab** command to install the changed file. For example:

```
crontab -1 > new.crontab
```

gets a copy of the **uucp** user's **crontab** file into **new.crontab**.

```
vi new.crontab
```

¦ Copyright IBM Corp. 1985, 1991

uses **vi** to edit (delete comment characters) from the copy.

```
  crontab new.crontab
```

installs the changed file.

See *AIX Operating System Commands Reference* for more information about **cron** and **crontab**.

Subtopics
6.8.6.1 Monitoring Remote Connections and File Transfers
6.8.6.2 Cleaning Up the Spooling Directories
6.8.6.3 Working with Log Files

¦ Copyright IBM Corp. 1985, 1991

6.8.6.1 Monitoring Remote Connections and File Transfers

BNU has several programs that enable you to monitor the progress of file transfers. These include the command **uustat**, which is employed by both general users and system administrators, and the command **Uutry**, which is used primarily by system administrators (although you do not need superuser privileges to issue **Uutry**).

Subtopics
6.8.6.1.1 The uustat Command
6.8.6.1.2 The Uutry Command
6.8.6.1.3 Monitoring a Remote Connection
6.8.6.1.4 Monitoring a File Transfer

¦ Copyright IBM Corp. 1985, 1991

6.8.6.1.1 The uustat Command

The BNU command **uustat** provides information about the status of several types of networking operations, including jobs that have been queued on a local system for transfer to a remote system. You will probably issue **uustat** frequently to check on the status of such queued jobs. The command is particularly helpful in monitoring requests for file transfers generated with the **uucp** and **uuto** commands, and requests for command executions generated with the **uux** command. (The **uustat** report also includes information about the status of mail activities.)

The **uustat** command also gives users limited control over BNU jobs queued to run on remote systems. By issuing the command with the appropriate flag, you can check the general status of BNU connections to other systems, and cancel transfer requests.

> For detailed information about the **uustat** command and numerous usage examples, see the BNU chapter in *Using the AIX Operating System*.
>
> For additional information about and examples of using **uustat**, see *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.1.2 The Uutry Command

The BNU command **Uutry**, which contacts a remote system with debugging turned on, is also extremely useful in monitoring the progress of a file transfer. In addition, the command enables you to monitor BNU connections to remote computers. **Uutry** (note the uppercase "U") invokes the **uucico** daemon, which is the BNU file-transport program, and then generates information about the way in which **uucico** is working.

You contact a specific remote system with **Uutry**, which produces debugging output that enables you to monitor the progress of **uucico** as it establishes the connection to the remote system, performs the remote login, and transfers a file. In addition to scrolling the debugging output on the screen of your local system, **Uutry** also directs this information to a file named **/tmp/system_name**, where **system_name** is the name of your local system.

The **Uutry** command has the following two flags:

-xdebug_level

-rsystem_name

Debugging Level (**-xdebug_level**)

The **Uutry** command produces a moderate amount of debugging information if you enter it without a flag. The command has a default ***debugging level*** of 5. (The debugging level, a single-digit number between 0 and 9, specifies the amount of debugging information that BNU places in the **tmp** file.) Higher numbers produce more detailed output, so the default produces a moderate amount of debugging information.

If you want more detailed information about the progress of the **uucico** operation, use the **-x** flag to specify a higher debugging level. For example,

```
Uutry -x9
```

instructs BNU to generate as much information as possible about the way in which **uucico** is working.

Override Retry Time (**-rsystem_name**)

When you issue the **uucp**, **uuto**, or **uux** command, that command calls the **uucico** daemon, which contacts the remote system and transfers the file. (The BNU scheduler, **uusched**, also starts **uucico** automatically at specified intervals.) If for some reason **uucico** cannot complete the connection and transfer the file, the daemon waits for a set amount of time and then tries again. The default retry time is five minutes.

Note: The time the remote system was last polled is specified in the file **/usr/spool/uucp/.Status/system_name**. However, the format of this file makes it somewhat difficult to read. To produce a more readable version, issue the **uustat -m** command.

When you invoke **uucico** with the **Uutry** command in order to monitor a remote connection or observe a file-transfer operation, you probably do not want to wait for several minutes between attempts to contact the remote system. In that case, use the **-r** flag to override the default retry time specified in the **.Status** file.

¦ Copyright IBM Corp. 1985, 1991

For information about transporting files, see "Transporting Copy Requests" in topic 6.8.7.1 and the BNU chapter in *Using the AIX Operating System*.

For additional information about **uucico**, see "The uucico Daemon" in topic 6.8.7.1.3 and the description of the command in *AIX Operating System Commands Reference*.

For information about **uusched**, see "The uusched Daemon" in topic 6.8.7.3.1 and the description of the command in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.1.3 Monitoring a Remote Connection

As described above, the **uucico** daemon is generally invoked either by a command such as **uucp** or **uux**, or automatically by the **uusched** daemon. (Of course, you can also issue **uucico** manually.) The **Uutry** command, which is actually a shell script stored in the **/usr/adm/uucp** directory, enables you to invoke **uucico** directly from your local computer with debugging turned on. You can then verify that the specified remote system is up and running, or use the command to monitor the file-transfer process.

For **Uutry** to execute, you either must be in the **/usr/adm/uucp** directory when you issue the command, or you must include the complete path name to the shell script on the **Uutry** command line. If you use **Uutry** frequently, you can put the path name to the command in the PATH entry in your **.profile** file.

Monitoring a Successful Connection: Following is an example of the way in which **Uutry** can help you monitor the **uucico** process if users at your site report file-transfer problems.

1. First, issue **uustat** to determine the status of all the transfer jobs in the current queue:

   ```
   uustat -q
   ```

   The system displays a status report like the following:

   ```
   venus   3C  (2)  05/09-11:02  CAN'T ACCESS DEVICE
   hera    1C       05/09-11:12  SUCCESSFUL
   merlin  2C       05/09-10:54  NO DEVICES AVAILABLE
   ```

   This report indicates that three command (**C**.*) files intended for system **venus** have been in the queue for two days. There could be several reasons for this delay--for example, **venus** could have been shut down for maintenance, the modem could be turned off, and so on.

2. Before you begin more extensive troubleshooting activities, issue **Uutry** to determine whether your local system can contact **venus** at this time:

   ```
   /usr/adm/uucp/Uutry -r venus
   ```

   This command starts **uucico** with a moderate amount of debugging and the instruction to override the default retry time. The **Uutry** command directs the debugging output to a temporary file, **/tmp/venus**, and executes a **tail -f** command.

   Note: If there is a large amount of debugging information, you can use the **INTERRUPT** sequence (usually **Ctrl-C**) to return to the **Uutry** shell while the command continues to place the debugging output in the temporary file. Then, when the operation is finished, issue the **pg** command to examine the **/tmp/venus** file.

3. If your local system succeeds in establishing a connection to **venus**, the debugging output will contain a good deal of information. However, the final line in this script is the most important:

   ```
   Conversation Complete:  Status SUCCEEDED
   ```

4. In the case of a successful connection, assume that the temporary

¦ Copyright IBM Corp. 1985, 1991

file-transfer problems are now resolved and issue **uustat** again to make certain that the files in the spooling directory were transferred successfully to **venus**.

Monitoring an Unsuccessful Connection:

1. Issue **uustat** to determine the status of the jobs intended for the remote system.

2. If the status report indicates a problem transferring a file to a specific remote system, issue the **Uutry** command to determine whether your local system can now contact that remote system.

3. If your local system cannot contact the remote system **venus**, the debugging output generated by **Uutry** will contain the following type of information (the exact form of the output may vary):

```
mchFind called (venus)
conn (venus)
getto ret -1
Call Failed:  CAN'T ACCESS DEVICE
exit code 101
Conversation Complete:  Status FAILED
```

4. First check the physical connections between the local and the remote systems. Make sure that the remote computer is turned on and that all the cables are properly connected, that the ports are enabled or disabled (as appropriate) on both systems, that the modems (if applicable) are working, and so on.

   For detailed information about this hardware, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

5. If the physical connections are correct and secure, then you must check all the relevant configuration files on both the local and the remote systems.

   Make certain that the entries in the **Devices**, **Systems**, and **Permissions** files are correct on both systems.

   In addition, if you are using a modem, make sure that the **Dialers** and **Dialcodes** files contain the proper entries.

   If you are using a TCP/IP connection, make sure the **uucpd** daemon is running, and that the configuration files contain the correct **TCP** entries.

6. Once you have checked the physical connections and configuration files, issue **Uutry** again. If the debugging output still reports that the connection failed, you may need to confer with a member of your systems support team. Be sure to save the debugging output produced by the **Uutry** command; it may prove helpful in diagnosing the problem.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.1.4 Monitoring a File Transfer

In addition to monitoring remote connections, the **Uutry** command also enables you to monitor file transfers. The two processes are very similar.

1. Issue the **uustat** command to check the status of the files in the spooling directory on your local system.

2. Now issue **Uutry** to verify that the local system can contact the remote system.

3. If the debugging output indicates that the connection was not successful, follow the steps described in "Monitoring an Unsuccessful Connection" in topic 6.8.6.1.3. If the connection was successful, go on to step 4.

4. Now prepare a file for transfer using the **uucp** command with the **-r** flag:

   ```
   uucp -r test1 venus!~/test2
   ```

   The **-r** option instructs BNU to place the **test1** file in the queue but **not** to start the **uucico** daemon.

   Note: If you use **csh**, place a backslash (\) in front of the exclamation mark (!), for example:

   ```
   uucp -r test1 venus\!~/test2
   ```

5. Now issue **Uutry** with the **-r** flag to start **uucico** with debugging turned on. The daemon contacts **venus**, logs in, and transfers the file, while **Uutry** produces debugging output that enables you to monitor the **uucico** process.

For additional information about handling file-transport problems, see "Login Failures" in topic 6.8.9.5.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.2 Cleaning Up the Spooling Directories

Each system connected by BNU has several types of spooling directories:

The **/usr/spool/uucp/system_name** directory contains queued requests issued by local users for file transfers to remote systems, and for command executions on remote systems. The queued files remain in this spooling directory until they are transferred to the designated system, removed manually with the **uucleanup** command, or removed automatically by the **uudemon.cleanup** shell script.

The **/usr/spool/uucppublic** directory is the BNU public directory. When a user transfers a file to a remote system or issues a request to execute a command on another system, the files generated by these BNU commands are stored in the public directory on the designated system. (However, the user can specify a destination other than the public directory when issuing the **uucp**, **uuto**, or **uux** commands.) Again, the transferred files remain in this directory until they are removed manually or automatically.

This section includes information about the **uucleanup** command and about cleaning up the files in the spooling and public directories.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.8.6.2.1 The uucleanup Command

The **uucleanup** program scans the **/usr/spool/uucp/system_name** directory for files older than a specified period of time and takes appropriate action to remove such files in a useful way. The command does the following:

Informs the system manager of requests to send files to and receiv files from remote systems that the local system cannot contact.

Warns users about requests that have been waiting in the spoolin directory for a given period of time (the default is one day).

Returns to the original sender mail that cannot be delivered to th requested recipient.

Removes all other files older than a specified number of days from th spooling directory.

Note: Depending on the size of your installation and the available storage space on the local system, you can set the default for any length of time, but you should probably allow files to remain in the spooling directory for at least the default number of days, as described in the command options.

The **uucleanup** program is started by the shell **uudemon.cleanup**, located in **/usr/adm/uucp**. The shell script is, in turn, started periodically by the **cron** daemon, based on instructions in the file **/usr/spool/cron/crontabs/uucp**. In general, **uucleanup** is executed automatically, although you can start it manually if you are logged in as **su**.

Note: Automatic cleanup is not enabled when BNU is installed. As described in "Cleaning Up Files and Directories (uudemon.cleanup)" in topic 6.8.8.2, you must edit the **uucp crontab** file and remove the comment character (#) from the beginning of the **uudemon.cleanup** line to enable the automatic cleanup routine. Remember to use the **crontab** command to install the changed file.

For additional information about automatic cleanup procedures, see the following:

"Cleaning Up Undeliverable Jobs" in topic 6.8.6.2.2

"Cleaning Up the Public Directory" in topic 6.8.6.2.3

"Running Automatic Maintenance Routines" in topic 6.8.8.

Following is the form of the **uucleanup** command:

uucleanup [**options**]

The command includes the following flags:

-Ctime
-Dtime
-Ttime
-Wtime
-Xtime
-mstring
-otime
-ssystem

Remove Command Files (**-Ctime**)

¦ Copyright IBM Corp. 1985, 1991

This flag removes any command (**C.***) files as old as, or older than, the number of days specified in **time**, and sends notice of the removal to the user who requested the file transfer or command execution. Unless you specify otherwise, the default **time** is seven days.

For information about command files, see "Additional Information about Command Files" in topic 6.8.7.1.4.

Remove Data Files (**-Dtime**)

This flag removes any data (**D.***) files as old as, or older than, the number of days specified in **time**. Again, the default is seven days. This option also instructs **uucleanup** to attempt to deliver any mail messages in the spooling directory.

For information about data files, see "Additional Information about Data Files" in topic 6.8.7.1.5.

Warn Requestor about Remaining Command Files (**-Wtime**)

This flag sends a mail message to the user who requested the file transfer or command execution, warning that command files as old as, or older than the number of days specified in **time** are still in the spooling directory. The default is one day. The message includes the job ID and, in the case of mail, the actual mail message. You can use the **-m** option to include a message line telling the user who to contact to check on the transfer problem.

Remove Temporary Files (**-Ttime**)

This flag removes any leftover temporary files (**T.*** files) as old as, or older than, the number of days specified in **time**. The default is two days.

Remove Execute Files (**-Xtime**)

This flag removes any execute (**X.***) files as old as, or older than, the number of days specified in **time**. The default is two days. There are probably no data files related to these execute files. If any related data files do exist, however, the **-Dtime** processing removes them, as described above.

For information about execute files, see "Additional Information about Execute Files" in topic 6.8.7.2.4.

Send Message Line (**-mstring**)

This flag instructs the **uucleanup** command to include a specified line of text in the warning message generated by the **-Wtime** option. The default message line is "See your local administrator to locate the problem".

Remove Other Files (**-otime**)

This flag removes files other than command, data, or execute files. These other files must be as old as, or older than the number of days specified in **time**. The default is two days.

Execute Only on Specified System (**-ssystem**)

¦ Copyright IBM Corp. 1985, 1991

This flag executes the **uucleanup** command only on the spooling directory on the specified system. The default is to clean up all the BNU spooling directories.

Note: Unless you set one of the **time** flags to a specific number of days, **uucleanup** uses the default time values.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.2.2 Cleaning Up Undeliverable Jobs

You should issue the **uustat** command frequently to determine the status of the connections to various remote systems, and to check the number and age of the file-transfer and command execution requests queued in the **/usr/spool/uucp/system_name** directory.

Note: You should also uncomment the **uudemon.admin** shell script line in the uucp crontab file so that you automatically receive status information about the queue at least once a day. For information about enabling this feature, see "Getting Status Information (uudemon.admin)" in topic 6.8.8.1.

The status information you need to check carefully includes the following:

The age in days of the oldest request in each queu

The number of times the local system has tried and failed to reach th specified computer

The reason for the failure to contact the specified system

You can probably delete (either manually or automatically) execute files that have been in the queue for at least two days. The reason they are still queued is that the related data files required to execute the specified command on the designated system were not transferred, for some reason. Since **D.*** files are generally sent at the same time as **X.*** files, the transfer probably failed at the point of destination.

For information about monitoring file-transfer failures, see "Monitoring a File Transfer" in topic 6.8.6.1.4 and "Login Failures" in topic 6.8.9.5.

You should be careful, however, before deleting old command files created to transfer files from the local to a remote system. Before removing these old **C.*** files, you should make every possible effort to establish the connection and transfer the files. Again, see "Monitoring a File Transfer" in topic 6.8.6.1.4 for information about monitoring remote-system connections and file-transfer requests.

As described above, you can remove these files manually, by issuing the **uucleanup** command with the appropriate options, or you can have them removed automatically by the **uudemon.cleanup** shell script, which is described in some detail in "Cleaning Up Files and Directories (uudemon.cleanup)" in topic 6.8.8.2.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.2.3 Cleaning Up the Public Directory

All spooling directories are dynamic, including the public directory. Depending upon the size of your installation and the number of files sent to the local **/usr/spool/uucppublic** directory by users on remote systems, this directory may at times become quite large.

The **uudemon.cleanup** shell script cleans up all the BNU spooling directories, including the public directories, unless you direct it to clean up only the directories on one specific system by issuing the **uucleanup -ssystem_name** command.

In order to keep the local file system from overflowing when users send files to the public directory, the **uudemon.cleanup** script includes a **find** command intended to deal with this problem. The **find** command locates any empty directories and any files older than seven days, and the shell script then deletes these items.

If the local system does not have enough storage space to permit a large **/usr/spool/uucppublic** directory, you can change the seven-day default to a shorter time period.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.3 Working with Log Files

BNU creates individual log files for each remote system with which your local system communicates using the **uucp**, **uuto**, or **uux** commands.

BNU normally places status information about each transaction in the appropriate log file each time you use the networking utilities facility. When more than one BNU process is running, however, the system cannot access the log file, so it places this status information in a file with a **.LOG** prefix that covers just the single transaction.

You can use **uulog** to display a summary of **uucp** and **uux** requests by user or by system. All these transactions are logged in files with a form of the name **/usr/spool/uucp/.LOG//daemon_name/system_name**.

> The **uucp** and **uuto** commands are executed by the **uucico** daemon. The **uucico** activities are logged in **/usr/spool/uucp/.LOG/uucico/system_name**.
>
> The **uux** command is executed by the **uuxqt** daemon. The **uuxqt** activities are logged in **/usr/spool/uucp/.LOG/uuxqt/system_name**.

You can examine these individual log files by issuing **uulog** directly. However, you can also have BNU automatically append these temporary log files to a primary log file. This is called ***compacting*** the log files, and it is handled by the **uudemon.cleanup** shell script, as described in "Compacting Log Files" in topic 6.8.6.3.2.

Subtopics
6.8.6.3.1 The uulog Command
6.8.6.3.2 Compacting Log Files
6.8.6.3.3 Cleaning Up sulog and cron/log

¦ Copyright IBM Corp. 1985, 1991

6.8.6.3.1 The uulog Command

Following is the form of the **uulog** command:

uulog [**options**]

The command includes the following flags:

-fsystem_name **-xsystem_name**
-ssystem_name **-number**

Display Log for Specified System (**-fsystem_name**)

This flag performs a **tail -f** on the file-transfer log for the specified system. In this case, that means BNU displays the end of the log file. Use **INTERRUPT** (usually **Ctrl-C** on the keyboard) to leave the file and return to the prompt.

Display Summary of Transfer Requests (**-ssystem_name**)

This flag displays a summary of information about **uucico** activities for the specified system.

Display uuxqt Log (**-xsystem_name**)

This flag displays the **uuxqt** log file for the specified system.

Execute tail -f for Specified Lines (**-number**)

This flag instructs BNU to execute a **tail -f** command for the specified number of lines.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.3.2 Compacting Log Files

As discussed above, you can use **uulog** to access the individual log files on each computer. For example, system **hera** has a log file for **uucico** requests and a log file for **uuxqt** requests, and you can examine each of these files with **uulog**.

However, you may find it more useful to have BNU automatically append these various log files to one primary log file, rather than examining each file individually. The **uudemon.cleanup** shell script handles this task.

As discussed in "Running Automatic Maintenance Routines" in topic 6.8.8, the instructions governing the execution of the automatic maintenance routines are contained in the **/usr/spool/cron/crontabs/uucp** file. The **cron** daemon automatically runs commands and shell scripts at specified dates and times according to the instructions you include in this **crontabs/uucp** file.

Whenever **cron** executes **uudemon.cleanup**, that script combines the **uucico** and **uuxqt** log files on a system and stores them in a directory named **/usr/spool/uucp/.Old**. The default is for **uudemon.cleanup** to save log files that are two days old.

You can change the default by simply modifying the appropriate line in the shell script. If storage space is a problem on a particular system, consider reducing the number of days that the files are kept in the individual log files.

¦ Copyright IBM Corp. 1985, 1991

6.8.6.3.3 Cleaning Up sulog and cron/log

The **/usr/adm/sulog** and **/usr/lib/cron/log** files are both related indirectly to BNU transactions.

The **sulog** file contains a history of superuser (**su**) command usage. Because the various **uudemon** entries in the **/usr/spool/cron/crontabs/uucp** file each uses the **su** command, the **sulog** file can grow quite large over a period of time. You should purge this file periodically to keep it to a reasonable size.

In a similar way, the **/usr/lib/cron/log** file contains a history of all the processes generated by **/etc/cron**. Over a period of time, this file too will become quite large, and you should purge it periodically to limit its size.

¦ Copyright IBM Corp. 1985, 1991

6.8.7 Using the Daemons

BNU contains two daemons that handle file transfers and remote-command executions. These are the **uucico** daemon, which transports files from one system to another, and the **uuxqt** daemon, which executes specified AIX commands on designated systems.

In addition to these daemons, BNU also includes a daemon that schedules the transfer of files queued in the spooling directory. This is the **uusched** daemon.

This section includes the following types of information:

- How BNU transfers copy requests, using the **uucp** command and the **uucico** daemon
- How BNU executes remote commands, using the **uux** command and the **uuxqt** daemon
- How BNU schedules work in the spooling directory

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.8.7.1 Transporting Copy Requests

BNU is used primarily to perform these kinds of tasks:

Transfer files within and between systems, using the **uucp** and **uuto** commands and the **uucico** daemon.

Request a remote system to run AIX commands for another system, usin the **uux** command and the **uuxqt** daemon. For information about running AIX commands on remote systems, see "Executing Remote Commands" in topic 6.8.7.2.

Following is a brief overview of the file-transfer process:

1. A user issues the BNU command **uucp** (or **uuto**) to send a source file from a local to a remote system. This command carries out the file transfer in two steps: It first creates a command file in the spooling directory on the local computer, and then calls **uucico** to send the source and command files to the remote computer.

2. If the user has invoked **uucp** with the **-C** flag to send the file to the spooling directory for transfer, **uucp** creates not only a command file, but also a data file that contains the actual source file.

3. Once the command file (and data file, if necessary) is created, **uucp** calls **uucico**, which in turn contacts the remote computer and delivers the file.

The rest of this section contains the following information:

An overview of the directories and files used in transferring file and executing command remotely

A brief discussion of the **uucp** command

A discussion of the **uucico** daemon

Examples of command and data files

Subtopics
6.8.7.1.1 Overview of Files and Directories
6.8.7.1.2 The uucp Command
6.8.7.1.3 The uucico Daemon
6.8.7.1.4 Additional Information about Command Files
6.8.7.1.5 Additional Information about Data Files

¦ Copyright IBM Corp. 1985, 1991

6.8.7.1.1 Overview of Files and Directories

BNU uses a number of directories and files when transferring information and running commands remotely. Following are brief descriptions of some of these directories and files.

The /usr/adm/uucp Directory

This directory contains the files in the supporting data base, several versions of the **uutry** command, and four scripts for automatic maintenance routines.

The /usr/lib/uucp Directory

This directory contains the **uucheck** and **uucleanup** commands, and the **uucico**, **uusched**, and **uuxqt** daemons.

The /usr/spool/uucp/.Admin Directory

This directory contains the following files:

audit	**Foreign**
errors	**xferstats**

The most important of these is the **xferstats** file. This file includes information about the status of the transfer request, including such items as the system name and the name of the user requesting the transfer, the date and time of the transfer, the name of the device used in the transfer, the size of the transferred file, and so on.

The /usr/spool/uucp/.Corrupt Directory

This directory contains copies of files that, for some reason, could not be processed. If, for example, a file is not in the correct form for transfer, BNU places a copy of that file in the **.Corrupt** directory for later handling by the system manager. However, this directory is rarely used.

The /usr/spool/uucppublic Directory

This is the public directory for the BNU facility, and one of these directories exists on every system connected by the networking utilities. When a user transfers a file to a remote system or issues a request to execute a command on another system, the files generated by these BNU commands are stored in the public directory on the designated system until the destination directory is ready to receive them. (A user can also specify a destination other than the public directory when issuing the **uucp**, **uuto**, or **uux** commands.)

The /usr/spool/uucp/.Status Directory

This directory contains additional types of information about the file transfer, such as the machine called, the time of the last call in seconds, the status of the last call, the number of retries, the retry time in seconds to the next call, and so on.

The /usr/spool/uucp/system_name Directory

This is the spooling directory on the local system. It contains queued requests issued by local users for file transfers to remote systems and

¦ Copyright IBM Corp. 1985, 1991

for command executions on remote system.

The /usr/spool/uucp/.Xqtdir Directory

This directory contains execute files with lists of commands that remote system may run.

The /usr/spool/uucp/.Workspace Directory

This directory holds temporary files of various kinds that the file transport programs use internally.

BNU also uses three types of administrative files to transfer data between systems:

Command/work files These files contain the directions for **uucico**.

Data files These files contain the data to be sent to remote systems.

Execute files These files contain instructions for running commands that require the resources of a remote system.

For detailed information about command files, se "Additional Information about Command Files" in topic 6.8.7.1.4.

For detailed information about data files, see "Additional Information about Data Files" in topic 6.8.7.1.5.

For detailed information about execute files, se "Additional Information about Execute Files" in topic 6.8.7.2.4.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.1.2 The uucp Command

Users issue the **uucp** command to copy one or more source files from one AIX system to one or more destination files on another AIX system.

The **uucp** command first creates a command file in the spooling directory on the local system. If the user has issued the command with the **-C** flag, **uucp** also creates a data file that contains the actual source file. After creating the command file (and data file, if necessary), **uucp** calls the **uucico** daemon, which handles the actual file transfer.

For detailed information about using the **uucp** command, see the BNU chapter in *Using the AIX Operating System*.

For detailed information about the command itself, see the descriptio of **uucp** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.1.3 The uucico Daemon

The **uucico** daemon selects the device to be used for the communication link, establishes the connection with the specified system, invokes the appropriate login sequence, checks permissions, transfers the command (and data) files, logs the results of the transfer, and notifies a designated user that the transfer is complete. Both the local and the remote systems run **uucico**, and the two daemons talk to each other to perform these tasks.

Specifically, **uucico** assists **uucp** (and **uux**) by performing the following actions:

Scanning the spooling directory on the local system for transfe requests

Placing a call to the specified remote syste

Negotiating a line protocol to be used to connect the local and remot systems

Running all transfer requests from both the local and the remot system

Logging transfer requests and completions

Note: In the case of a **uux** request for the execution of an AIX command on a remote system, **uucico** transfers the files, and then the **uuxqt** daemon executes the command on the remote system. For information about **uuxqt**, see "Executing Remote Commands" in topic 6.8.7.2.

When a user issues **uucp** or **uux**, those commands call **uucico**, which runs as a background process. In addition, **uucico** is also started periodically by the BNU scheduler, **uusched**. The scheduling daemon is, in turn, started periodically by the **cron** daemon, based on instructions in the uucp crontab file. These instructions execute the shell script **/usr/adm/uucp/uudemon.hour**.

For detailed information about **uusched**, see "Scheduling Work in the Spooling Directory" in topic 6.8.7.3.

For detailed information about the **uudemon.hour** shell script, see "Calling File-transport Programs (uudemon.hour)" in topic 6.8.8.3.

You can also start **uucico** manually for debugging purposes. Following is the form of the command:

uucico [**options**]

Note: You must either be in the **/usr/lib/uucp** directory when you issue the **uucico** command, or you must enter the command with the full path name: **/usr/lib/uucp/uucico**.

The **uucico** command uses the following flags:

-rrole_number
-ssystem_name
-xdebug_level

Role Number (**-rrole_number**)

¦ Copyright IBM Corp. 1985, 1991

This flag specifies the server and client relationship. The role numbers are 1 for the server mode and 0 for the client mode. The default is 0 because **uucico** is generally started automatically by a program such as **uucp** or by the **cron** daemon. If you start **uucico** manually, set this flag to 1.

Remote System Name (**-ssystem_name**)

This flag specifies the name of the remote system. Use this flag only when you are starting **uucico** manually. BNU supplies the system name internally when **uucico** is started automatically.

Debug Level (**-xdebug_level**)

This flag produces debugging information about the progress of the **uucico** activity. The valid range for the debugging level is 0 to 9, with a default of 5. Higher numbers produce more detailed debugging information.

Note: As described in "Monitoring a File Transfer" in topic 6.8.6.1.4, the BNU command **Uutry** also starts **uucico** with debugging turned on.

When starting **uucico** manually, use the following form to run the daemon as a background process on the computer specified by **-ssystem_name**:

```
/usr/lib/uucp/uucico -r1 -ssystem_name &
```

For additional information about using the **uucico** daemon for debugging, and for example debugging output, see "Login Failures" in topic 6.8.9.5.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.1.4 Additional Information about Command Files

The full path name of a command/work file is a form of the following:

/usr/spool/uucp/system_name/C.sitename_Nxxxx

where the first system name is that of the local system, **sitename** is the name of the cluster site, the **N** character represents the grade of the work, and the **xxxx** notation is the four-digit hexadecimal transfer-sequence number--for example, **C.merlinC3119**.

Note: The grade of the work specifies when the file is to be transmitted during a particular connection. The grade notation is a single number (0-9) or letter (A-Z, a-z). Lower sequence characters cause the file to be transmitted earlier in the connection than do higher sequence characters. The number 0 is the highest grade, signifying the earliest transmittal; z is the lowest grade, specifying the latest transmittal. The default grade is **N**.

A command file consists of a single line that includes the following kinds of information in the following order:

1. An **S** (send) or **R** (receive) notation.

   Note: A send command file is created by the **uucp** or **uuto** commands; a receive command file is created by the **uux** command.

2. The full path name of the source file being transferred. A receive command file, discussed below, does not include this entry.

3. The full path name of the destination file or a path name preceded by **~user**, where **user** is a login name on the specified system.

4. The sender's login name.

5. A list of the options, if any, included with the **uucp**, **uuto**, or **uux** command.

6. The name of the data file associated with the command file in the spooling directory. This field must contain an entry. If one of the data-transfer commands (such as **uucp** with the default **-c** flag) does not create a data file, BNU instead creates a placeholder with the name **D.0** for send files, or **dummy** for receive files.

7. The source file's permissions code, specified as a three-digit octal number (for example, 777).

8. The login name of the user on the remote system who is to be notified when the transfer is complete.

Examples of Two Send Command Files: The send command file **/usr/spool/uucp/venus/C.heraN1133**, created with the **uucp** command, contains the following fields:

```
S /u/amy/f1 /usr/spool/uucppublic/f2 amy -dC D.herale73655 777 lgh
```

where:

1. **S** denotes that **uucp** is sending the file.

¦ Copyright IBM Corp. 1985, 1991

2. The full path name of the source file is **/u/amy/f1**.

3. The full path name of the destination is **/usr/spool/uucppublic/f2**, where **/usr/spool/uucppublic** is the name of the BNU public spooling directory on the remote computer and **f2** is the new name of the file.

   Note: The destination name may be abbreviated as **~uucp/f2**. The tilde (~) is a shorthand way of designating the public directory.

4. The person sending the file is **amy**.

5. The sender entered the **uucp** command with the **-C** option, specifying that **uucp** should transfer the file to the local spooling directory and create a data file for it. (The **-d** option, which specifies that the command should create any intermediate directories needed to copy the source file to the destination, is a default.)

6. The name of the data file, **D.herale73655**, which **uucp** assigns.

7. The octal permissions code 777.

8. The login name of the user on **hera** who is to be notified of the file's arrival, **lgh**.

Look at another send command file, **/usr/spool/uucp/hera/C.zeusN3130**. In this case, the user issued the **uuto** command to produce the following file:

```
S  /u/amy/out  ~/receive/msg/zeus  amy  -dcn  D.0  777  msg
```

The **S** denotes that the source file **/u/amy/out** was sent to the **receive/msg** subdirectory in the public spooling directory on system **zeus** by user **amy**.

Note: The **uuto** command creates the directory **receive/msg** if it does not already exist.

The **uuto** command used the defaults **-d** (create directories), **-c** (transfer directly, no spooling directory or data file), and **-n** (notify recipient). The **D.0** notation is a placeholder, **777** is the permissions code, and **msg** is the recipient.

Example of a Receive Command File: The format of a receive command file is somewhat different from that of a send command file. The **uux** command creates a receive command file when files required to run a specified AIX command on a remote system are not present on that specified system.

For example, running the following command on the local site **zeus**:

```
uux - "diff /u/amy/out hera!/u/amy/out2 > ~uucp/DF"
END OF FILE
```

produces a receive command file named **/usr/spool/uucp/hera/C.zeusR1e94** on the spooling site for the remote site **hera**. To find out the spooling site, see the **/usr/adm/uucp/Spools** file.

Note: The command in this example invokes **uux** to run a **diff** command on the local system, comparing file **/u/amy/out** with file **/u/amy/out2**, which is stored on remote system **hera**. The output of the comparison is placed in file **DF** in the public directory on the local system.

¦ Copyright IBM Corp. 1985, 1991

The actual receive command file looks like this:

```
  R  /u/amy/out2 D.hera1e954fd  amy  -  dummy  0666  amy
```

The **R** denotes a receive file. The **uucico** program is started by regularly scheduled **cron** programs on the spooling site and gets the file **/u/amy/out2** from **hera** and places it in a data file called **D.spool-site1e954fd** for the transfer. Once the files are transferred, the **uuxqt** program executes the command on the specified system.

User **amy** issued the **uux** command with the - (minus sign) option, which makes the **uux** command expect standard input from the command line until you press **END OF FILE**. No data file was created in the local spooling directory, so BNU uses **dummy** as a placeholder. The permissions code is 666 (BNU prefixes the three-digit octal code with a 0), and user **amy** is to be notified when the command has finished executing.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.1.5 Additional Information about Data Files

The full path name of a data file has the following general form:

/usr/spool/uucp/system_name/D.ssssxxxx###

The system name is that of the local system, and **ssss** is either the name of the remote computer or the **sitename** of the originating site if the remote computer is part of a TCF cluster. For outgoing data files, the **ssss** component represents the **sitename** of the originating site in the TCF cluster. The **xxxx###** represents the hexadecimal sequence number of the command file associated with that data file--for example, **D.venus471afd8**.

After a set period of time (specified in the **uusched** program), the **uucico** daemon transfers the data file to the designated system. It places the original data file in a subdirectory of the BNU spooling directory named **/usr/spool/uucp/system_name**, where **system_name** is the name of the computer that is transmitting the file. BNU creates a temporary data file to hold the original data file.

The full path name of the temporary data file has the following general form:

/usr/spool/uucp/system_name/TM.00PID.000

The **system_name** is the name of the computer that is sending the file, and **TM.xxPID.000** is the name of the file--for example, **TM.00451.000**. (The PID number is the process ID of the job.)

After receiving the entire file, BNU takes one of three actions:

If the file was sent with **uucp** and there were no transfer problems, the program immediately renames the **TM.*** file with the appropriate data file name, such as **D.venus471afd8**, and sends it to the specified destination.

If the file was sent with the **uuto** command, BNU also renames the temporary data file with the appropriate **D.*** name. It then places the data file in the public directory, **/usr/spool/uucppublic**, where the user receives and handles it with one of the **uupick** options.

If there were transfer problems (such as a failed login or a unavailable device), the temporary data file remains in the spooling subdirectory. The **uudemon.cleanup** shell script removes these files automatically at specified intervals, or you can remove them manually.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.2 Executing Remote Commands

Users can run AIX commands on remote systems by invoking the BNU command **uux**, which functions in the following way:

1. The command gathers any necessary files from the designated systems.

2. It then creates a command file, a data file, and an execute file. The command files contain the same type of information as those created by **uucp**, while **uux** data files either contain the data for a remote command execution, or else become execute files on remote systems for remote command execution. Execute files contain the command string to be run on the specified system.

3. After creating the files, **uux** calls the **uucico** daemon, which contacts the designated system and delivers the files.

4. The **uuxqt** daemon executes the command on the specified system, placing any output from the command in a designated file on a specified system.

In executing remote commands, BNU uses the same directories and files described in "Overview of Files and Directories" in topic 6.8.7.1.1. However, in addition to command and data files, **uux** also requires execute (**X.***) files, which are described in detail in "Additional Information about Execute Files" in topic 6.8.7.2.4.

Subtopics
6.8.7.2.1 The uux Command
6.8.7.2.2 The uuxqt Daemon
6.8.7.2.3 Limiting the Number of Remote Executions
6.8.7.2.4 Additional Information about Execute Files

¦ Copyright IBM Corp. 1985, 1991

6.8.7.2.1 The uux Command

As described above, the **uux** command runs a specified AIX command on a specified AIX system.

After creating the required command, data, and execute files, **uux** calls the **uucico** daemon to transfer the files from the spooling directory on the local system to the designated remote system. Once the files are transferred, the **uuxqt** daemon executes the command string contained in the execute file on the remote system.

The command string is made up of one or more arguments that look like an AIX command line, except that the command string may be prefixed by the name of the remote system in the form **system_name**!. Unless the user entering the **uux** command includes the **-n** flag, the command notifies that user if the remote system does not run the command. This response comes by mail from the remote system.

For detailed information about using the **uux** command, refer to the BNU chapter in *Using the AIX Operating System*.

For detailed information about the command itself, see the descriptio of **uux** in *AIX Operating System Commands Reference*.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.2.2 The uuxqt Daemon

After a user enters **uux** to execute an AIX command on a designated system, the BNU command generates command, data, and execute files. The execute file, described in detail in "Additional Information about Execute Files" in topic 6.8.7.2.4, contains the names of the files needed to run the command on the remote computer. The **uux** command then calls the **uucico** daemon.

The **uucico** daemon places the transferred files in the spooling directory on the designated system. The **uuxqt** daemon searches this directory periodically for **X.*** files whose names indicate that they have been sent from another system. When it finds such a file, **uuxqt** takes the following actions:

Reads the file to determine the list of data files required for th command execution.

Checks to see whether the required data files are available an accessible.

Checks the **Permissions** file to verify that the requested command may be executed on that system.

You can start **uuxqt** manually, but the daemon is generally started automatically at specified intervals by the **uudemon.hour** shell script, which is stored in **/usr/adm/uucp**. The shell script, in turn, is started periodically by the **cron** daemon, based on instructions in the file **/usr/spool/cron/crontabs/uucp**.

For information about the **uudemon.hour** shell script, see "Calling File-transport Programs (uudemon.hour)" in topic 6.8.8.3.

You start the **uuxqt** daemon manually by issuing the following command:

uuxqt [**options**]

Note: Either you must be in the **/usr/lib/uucp** directory when you issue the **uuxqt** command, or you must enter the full path name of the command: **/usr/lib/uucp/uuxqt**.

This command uses the following flags:

-ssystem_name
-xdebug_level

Remote System Name (**-ssystem_name**)

This flag specifies the name of the remote system. Use this flag only when you are starting **uuxqt** manually. BNU supplies the system name internally when **uuxqt** is started automatically.

Debug Level (**-xdebug_level**)

This flag produces debugging information about the progress of the **uuxqt** activity. The valid range for the debugging level is 0 to 9, with a default of 5. Higher numbers produce more detailed debugging information.

When starting **uucico** manually, use the following form to run the daemon as a background process on the computer specified by **-ssystem_name**:

¦ Copyright IBM Corp. 1985, 1991

```
/usr/lib/uucp/uuxqt -s system_name &
```

¦ Copyright IBM Corp. 1985, 1991

6.8.7.2.3 Limiting the Number of Remote Executions

The **Maxuuxqts** file, located in the **/usr/adm/uucp** directory, limits the number of **uuxqt** processes running simultaneously on a local system.

This file contains an ASCII number that you can change in order to meet the needs of your installation; the default is 2. In general, the larger the number, the greater the potential load on the local system.

The **Maxuuxqts** file requires no configuration and no maintenance unless the system on which it is installed is utilized frequently and heavily by users on remote systems.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.2.4 Additional Information about Execute Files

The full path name of a **uux** execute file is a form of the following:

/usr/spool/uucp/system_name/X.system_nameNxxxx

where the system name preceding the **X.** notation is the name of the local computer and the name following it is the name of the remote system. The **N** character represents the grade of the work, and the **xxxx** notation is the four-digit hexadecimal transfer-sequence number--for example, **X.zeusN2121**.

Note: Notice that the only difference between the name of a command file and that of an execute file is the **C.** or **X.** following the name of the local system. As with command files, the grade of an execute file specifies when that file is to be transmitted during a particular connection. Lower sequence characters cause the file to be transmitted earlier than do higher sequence characters. The default grade is **N**.

Standard Entries in an Execute File: An execute file consists of several lines, each with an identification character and one or more entries:

User Line U **user_name system_name**

This user line includes the login name of the user issuing the **uux** command, and the system from which the command was issued.

Error Status Line N or Z

The **N** character means that a failure message is **not** sent to the user issuing **uux** if the specified AIX command does not execute successfully on the remote system.

The **Z** character means that a failure message is sent to the user issuing **uux** if the specified AIX command does not execute successfully on the remote system.

Requestor's Name R **user_name**

This is the login ID of the user requesting the remote command execution.

Required File Line F **file_name**

This line contains the names of the files required to execute the specified command on the remote system. The **file_name** can be either the complete path name of the file including the unique transmission name assigned by BNU, or simply the transmission name without any path information (see the examples below). This line can contain zero or more file names. The **uuxqt** daemon checks for the existence of all listed files before running the specified AIX command.

Standard Input Line I **file_name**

¦ Copyright IBM Corp. 1985, 1991

The standard input is either specified by a < (less than) symbol in the command string, or inherited from the standard input of the **uux** command if that command was issued with the hyphen (-) flag.

If standard input is specified, it also appears in an F (Required File) line. If standard input is not specified, BNU uses **/dev/null**.

Standard Output Line O **file_name system_name**

This is the name of the file and system that are to receive standard output from the command execution. Standard output is specified by a > (greater than) symbol within the command string. (You cannot use the >> sequence in **uux** commands.) As was the case with standard input, if standard output is not specified, BNU uses **/dev/null**.

Command Line C **command_string**

This is the command string that the user requests to be run on the specified system. BNU checks the **Permissions** file on the designated computer to see whether the login ID may run the AIX command on that system.

All required files go to the execute directory, usually **/usr/spool/uucp/.Xqtdir**. After execution, the standard output is sent to the requested location.

Sample Execute Files: User **amy** on local system **zeus** issued the following command:

```
uux - "diff /u/amy/out hera!/u/amy/out2 > ~uucp/DF"
```

The command in this example invokes **uux** to run a **diff** command on the local system, comparing file **/u/amy/out** with file **/u/amy/out2**, which is stored on remote system **hera**. The output of the comparison is placed in file **DF** in the public directory on the local system.

That command produced an execute file named **/usr/spool/uucp/hera/X.zeusN212F**, which contains the following information:

```
U amy zeus
# return status on failure
Z
# return address for status or input return
R amy
F /usr/spool/uucp/hera/D.herale954fd out2
O ~uucp/DF zeus
C diff /u/amy/out out2
```

The user line identifies **amy** on **zeus**. The error status line indicates that **amy** will receive a failure status message if the **diff** command fails to execute. The requestor is **amy**, and the file required to execute the command is the following data file:

¦ Copyright IBM Corp. 1985, 1991

```
/usr/spool/uucp/hera/D.herale954fd out2
```

The output of the command is to be written to the public directory on **zeus** with the name **DF**. (Remember that **~uucp** is the shorthand way of specifying the public directory.) The final line is the command string that **amy** entered with the **uux** command.

Following is another example of an execute file:

```
U uucp hera
# don't return status on failure
N
# return address for status or input return
R uucp
F D.hera5eb7f7b
I D.hera5eb7f7b
C rmail amy
```

This indicates that user **uucp** on system **hera** is sending mail to user **amy**, who is also working on system **hera**.

¦ Copyright IBM Corp. 1985, 1991

6.8.7.3 Scheduling Work in the Spooling Directory

When users issue BNU commands to copy files and execute remote commands, the files containing these work requests are queued for transfer in the local **/usr/spool/uucp/system_name** directory. The BNU daemon **uusched** schedules the transfer of these files.

This section contains information about the **uusched** daemon and about the way in which the **Maxuuscheds** file works with that daemon to limit the number of remote systems that **uucico** can contact any one time.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.8.7.3.1 The uusched Daemon

This program schedules the transfer of files that are queued in the spooling directory, **/usr/spool/uucp**, on the local system.

The **uusched** daemon first randomizes the queued work in the spooling directory and then starts the **uucico** daemon with the **-ssystem_name** option. This flag tells **uucico** the system for which the selected job is scheduled.

You can start **uusched** manually, but the daemon is generally started automatically at specified intervals by the **uudemon.hour** shell script, which is stored in **/usr/adm/uucp**. The shell script, in turn, is started periodically by the **cron** daemon, based on instructions in the file **/usr/spool/cron/crontabs/uucp**.

For information about the **uudemon.hour** shell script, see "Calling File-transport Programs (uudemon.hour)" in topic 6.8.8.3.

You start the **uusched** daemon manually by issuing the following command:

uusched [**options**]

Note: Either you must be in the **/usr/lib/uucp** directory when you issue the **uusched** command, or you must enter the full path name of the command: **/usr/lib/uucp/uusched**.

This command uses the following flags:

-udebug_level
-xdebug_level

Pass Debug Level to uucico (**-udebug_level**)

This flag passes the **-xdebug_level** specification on to the **uucico** daemon, which then produces debugging output about the file-transport activities.

Debug Level (**-xdebug_level**)

This flag produces debugging information about the progress of the **uusched** activity. The valid range for the debugging level is 0 to 9, with a default of 5. Higher numbers produce more detailed debugging information.

When starting **uusched** manually, use the following form to run the daemon as a background process:

```
/usr/lib/uucp/uusched &
```

¦ Copyright IBM Corp. 1985, 1991

6.8.7.3.2 Limiting the Number of Scheduled Jobs

The file **/usr/adm/uucp/Maxuuscheds** limits the number of remote systems that the **uucico** program can contact at any one time. This file is used in conjunction with the **uusched** daemon and the lock files in the **/etc/locks** directory to determine the number of systems currently being polled. You use this file to help manage system resources and load averages.

The **Maxuuscheds** file contains an ASCII number that you can change in order to meet the needs of your installation; the default is 2. In general, the larger the number, the greater the potential load on the local system.

The **Maxuuscheds** file requires no configuration and no maintenance unless the system on which it is installed is utilized frequently and heavily by users on remote systems.

¦ Copyright IBM Corp. 1985, 1991

6.8.8 Running Automatic Maintenance Routines

The BNU software includes four shell scripts that run maintenance routines automatically. These include the following:

uudemon.admin	**uudemon.hour**
uudemon.cleanup	**uudemon.poll**

These shell scripts reside in the directory **/usr/adm/uucp**. They are started periodically by the **cron** daemon, based on instructions in the file **/usr/spool/cron/crontabs/uucp**.

The automatic maintenance routines are not enabled when you install BNU. You must edit the **crontabs/uucp** file, removing the comment character (#) from the beginning of each individual line that governs running the particular maintenance shell script. The changes made to this file will not take effect unless the changed file is installed or the system is restarted. Therefore, to uncomment certain lines in this file, use the **crontab** command while logged in as the BNU Administrator (**uucp** is the example login ID) to get a copy of the file, edit the copy, and then use the **crontab** command to install the changed file. For example:

```
crontab -1 > new.crontab
```

gets a copy of the **uucp** user's **crontab** file into **new.crontab**.

```
vi new.crontab
```

uses **vi** to edit (delete comment characters) from the copy.

```
crontab new.crontab
```

installs the changed file.

See *AIX Operating System Commands Reference* for more information about **cron** and **crontab**.

Subtopics
6.8.8.1 Getting Status Information (uudemon.admin)
6.8.8.2 Cleaning Up Files and Directories (uudemon.cleanup)
6.8.8.3 Calling File-transport Programs (uudemon.hour)
6.8.8.4 Polling Remote Systems (uudemon.poll)

¦ Copyright IBM Corp. 1985, 1991

6.8.8.1 Getting Status Information (uudemon.admin)

The **uudemon.admin** shell script mails status information about the BNU activities to the **uucp** login ID at intervals specified in the **crontabs/uucp** file. The shell script executes both the **uustat -p** and the **uustat -q** commands.

The **-p** flag instructs **uustat** to run a **ps -flp** (process status: full, long list of specified process IDs) for all PID numbers in the lock files.

The **-q** flag lists the jobs currently queued to run on each system; these jobs are either waiting to execute or in the process of executing. If a status file exists for the system, its date, time, and status information are reported.

You should execute the **uudemon.admin** shell script at least once a day. To run it automatically, uncomment the following line in the uucp crontab file by deleting the pound sign (#) that precedes this line:

```
48 8,12,16 * * * /bin/sh "/usr/adm/uucp/uudemon.admin > /dev/null"
```

The **48** notation represents minutes, the **8,12,16** notation represents hours based on the 24-hour clock, and the three asterisks (* * *) are placeholders representing the day of the month, the month of the year, and the day of the week. This line therefore instructs the **cron** daemon to run the **uudemon.admin** shell script at **48** minutes past the hours **8**, **12**, and **16**--that is, at 8:48 a.m., 12:48 p.m., and 4:48 p.m.

Note: Remember that these run intervals are just defaults. You can change the times at which **cron** executes **uudemon.admin** to fit the needs of your site simply by changing these time intervals.

Use the **pg** command to examine the **uudemon.admin** script, which is located in the directory **usr/adm/uucp**.

¦ Copyright IBM Corp. 1985, 1991

6.8.8.2 Cleaning Up Files and Directories (uudemon.cleanup)

As described in "Cleaning Up the Spooling Directories" in topic 6.8.6.2, the **uudemon.cleanup** shell script cleans up the BNU spooling directories and log files. The script deletes files in the spooling directory that are as old as, or older than a specified number of days, and removes empty directories.

The **uudemon.cleanup** shell script also updates archived log files, removing log information more than three days old. The script removes log files for individual computers from the **usr/spool/uucp/.Log** directory, merges them, and places them in the directory **usr/spool/uucp/.Old**, which contains old log information.

After performing the cleanup operations, the script mails the **uucp** login ID a summary of the status information gathered during the current day.

You can instruct **cron** to run the **uudemon.cleanup** shell script daily, weekly, or at longer intervals, depending on the amount of **uucico** and **uuxqt** transactions that occur on the local system.

To run this script automatically, uncomment the following line in the **/usr/spool/cron/crontabs/uucp** file by deleting the pound sign (#) that precedes this line:

```
45 23 * * * /bin/sh -c "/usr/adm/uucp/uudemon.cleanup > /dev/null"
```

The **45** notation represents minutes, the **23** notation represents hours based on the 24-hour clock, and the three asterisks (* * *) are placeholders representing the day of the month, the month of the year, and the day of the week. This line therefore instructs **cron** to run the **uudemon.cleanup** shell script at **45** minutes after hour **23**--that is, at 11:45 p.m.

Remember that these run intervals are just defaults. You can change the times at which **cron** executes **uudemon.cleanup** to fit the needs of your site simply by changing these time intervals.

Note: AIX allots BNU a specified amount of storage space for any one particular log file; the number of blocks is determined by the default **ulimit**. If the **uudemon.cleanup** script fails to execute because the **ulimit** is set too low for the requirements of the local system, delete the line shown above from the **crontab** file and add the following entry to the root **crontab** file:

```
45 23 * * * ulimit 5000; /bin/su uucp -c "/usr/adm/uucp/uudemon.cleanup \
> /dev/null"
```

You can use the **pg** command to examine the **uudemon.uucleanup** script, which is located in the directory **usr/adm/uucp**.

¦ Copyright IBM Corp. 1985, 1991

6.8.8.3 Calling File-transport Programs (uudemon.hour)

The **uudemon.hour** shell script is used in conjunction with the **Poll** file, the **uudemon.poll** shell script, and the **/usr/spool/cron/crontabs/uucp** file to initiate calls to remote systems.

Specifically, **uudemon.hour** calls various programs involved in transferring files between systems at specified hourly intervals. It calls the following:

The **uusched** daemon, which first searches the spooling directory on the local system for command files that have not been transferred to the specified remote system, and then schedules the transfer of those files

The **uuxqt** daemon, which searches the spooling directory for execute files that have been transferred to the local system but not yet processed on that system.

You can instruct **cron** to run the **uudemon.hour** shell script at specified hourly intervals. The frequency at which you run the script depends entirely on the amount of file-transfer activity originating from the local computer. If users on the local system initiate a large number of file transfers, you may need to specify that **cron** should start **uudemon.hour** several times an hour. If the number of file transfers originating from the local system is low, you can probably specify a start time once every four hours, for example.

To run this script automatically, uncomment the following line in the uucp crontab file by deleting the pound sign (#) that precedes this line:

```
25,55 * * * * /bin/sh -c  "/usr/adm/uucp/uudemon.hour > /dev/null"
```

The **25,55** notation represents minutes, and the four asterisks (* * * *) are placeholders representing the hour of the day, the day of the month, the month of the year, and the day of the week. This line therefore instructs **cron** to run **uudemon.hour** at **25** minutes past the hour and again at **55** minutes past the hour--for example, at 8:25 and 8:55 a.m., 9:25 and 9:55 a.m., and so on.

Again, note that these run intervals are defaults, which you can change to meet the needs of your site simply by changing the time intervals. For example, to run the shell script once every four hours, enter the number **4** in the time-interval field.

You can use the **pg** command to examine the **uudemon.hour** script, which is located in the directory **usr/adm/uucp**.

¦ Copyright IBM Corp. 1985, 1991

6.8.8.4 Polling Remote Systems (uudemon.poll)

As mentioned above, the **uudemon.poll** shell script is used in conjunction with the **Poll** file, the **uudemon.hour** shell script, and the uucp crontab file to initiate calls to remote systems.

The **uudemon.poll** shell script performs the following actions:

Polls the systems listed in the **Poll** file (**/usr/adm/uucp/Poll**)

Creates command files for the systems listed in the **Poll** file.

The time at which you run **uudemon.poll** depends on the time at which you run **uudemon.hour**. You generally schedule the polling shell script to before the hourly script. This schedule enables **uudemon.poll** to create any required command files before **cron** runs **uudemon.hour**.

Instruct **cron** to run **uudemon.poll** about five to ten minutes before running **uudemon.hour**. To run this script automatically, uncomment the following line in the uucp crontab file by deleting the pound sign (#) that precedes this line:

```
20,50 * * * * /bin/sh -c "/usr/adm/uucp/uudemon.poll > /dev/null"
```

As was the case with the **uudemon.hour** script, the **20,50** notation represents minutes, and the four asterisks (* * * *) are placeholders representing the hour of the day, the day of the month, the month of the year, and the day of the week. This line therefore instructs **cron** to run **uudemon.poll** at 20 minutes past the hour and again at 50 minutes past the hour--for example, at 8:20 and 8:50 a.m., 9:20 and 9:50 a.m., and so on.

Again, these run intervals are defaults. You can change the times at which **cron** executes **uudemon.poll** to correspond to the times you set up for **uudemon.hour**. The defaults specified in **/usr/spool/cron/crontabs/uucp** run **uudemon.poll** five minutes before running **uudemon.hour**.

You can use the **pg** command to examine the **uudemon.poll** script, which is located in the directory **usr/adm/uucp**.

¦ Copyright IBM Corp. 1985, 1991

6.8.9 Handling Common Problems

Much of the material in this chapter has dealt with handling problems that users may encounter when working with the BNU facility. This section deals briefly with the following situations:

Full spooling directorie

Untransferred file

Outdated **Systems** files

Faulty automatic call units (ACUs) and modem

Login failures

Subtopics
6.8.9.1 Full Spooling Directories
6.8.9.2 Untransferred Files
6.8.9.3 Outdated Systems Files
6.8.9.4 Faulty Automatic Call Units (ACUs) and Modems
6.8.9.5 Login Failures

¦ Copyright IBM Corp. 1985, 1991

6.8.9.1 Full Spooling Directories

If users at your site often transfer files and execute commands remotely using the BNU facility, the file system that spools incoming or outgoing work requests can run out of space. This makes it impossible to send jobs to, or receive jobs from remote systems.

The inability to receive files from remote systems is possibly the more serious of the two problems. When file space for the incoming work does become available, the local system may be flooded with a backlog of work sent from remote systems.

This type of problem can be solved by one of three mechanisms:

1. If the spool directories are filling up because cleanup is not done frequently enough, modify the line in the **uucp crontab** file that starts the **uudemon.cleanup** shell script. Change the numbers in the first two fields of the cleanup line to decrease the intervals between the times when the **cron** daemon executes **uudemon.cleanup**.

   For detailed information about cleaning up the spooling directories and modifying the cleanup shell script, see the following:

   "Cleaning Up the Spooling Directories" in topic 6.8.6.2

   "Cleaning Up Files and Directories (uudemon.cleanup)" in topic 6.8.8.2.

2. If the problem persists, it probably indicates that too little space has been allocated to the spool directories. Assign more disk space.

3. If the problem continues and all possible space has been allocated to spool directories on the existing UUCP gateway, UUCP spooling for the cluster must be shared. Provide one or more new cluster sites with modems and set them up as UUCP gateways. Allocate space for more spool directories on these new gateways.

¦ Copyright IBM Corp. 1985, 1991

6.8.9.2 Untransferred Files

If you need to transfer files from a local to a remote system immediately, rather than waiting for the execute time specified in the **/usr/spool/cron/crontabs/uucp** file, you can manually issue either the **uucico** command or the **uuxqt** command.

Use the **uucico** daemon to manually transfer data (**D.***) files. Use the **uuxqt** daemon to manually transfer execute (**X.***) files.

Note: If you wish to monitor the process of the file transfer, do **not** issue either command with the ampersand (&) character, which instructs AIX to run the specified command as a background process.

For detailed information about these procedures, see the following:

"Monitoring a File Transfer" in topic 6.8.6.1.4

"Transporting Copy Requests" in topic 6.8.7.1.

¦ Copyright IBM Corp. 1985, 1991

6.8.9.3 Outdated Systems Files

It can sometimes be difficult to maintain current **Systems** files when the users at your site communicate regularly with users at a remote site.

This can be a costly problem if your site communicates with the remote computers over telephone lines. For example, a modem and line may be unavailable for a significant length of time while a local system attempts to contact a remote system. This is particularly frustrating if the user cannot reach the remote system simply because the phone number of its modem has been changed or the system manager at the remote site has issued a new login ID or password for that system.

The best way to deal with this problem is to try to avoid it by communicating regularly with the system managers at remote sites. Be sure to contact the administrators of remote computers that are part of your network whenever you change a telephone number, a login ID, or a password--and request that these individuals send you the same information when they make similar changes to the configuration files of the computers at their installations.

For detailed information about the **Systems** file, see "Customizing the Systems File" in topic 6.8.5.2.3.

¦ Copyright IBM Corp. 1985, 1991

6.8.9.4 Faulty Automatic Call Units (ACUs) and Modems

The ACUs used at your site, as well as the incoming modems, may occasionally cause problems that make it difficult for users to contact other computers or to receive files.

You can generally identify such problems relatively easily because the status report generated with the **uustat** command includes the number of times that BNU has tried to reach the designated system and the reason why the contact failed.

If you suspect that the problem is caused by a bad transmission line, use the **cu** command to monitor the line. Issue **cu** to connect to a different system than the one listed in the status report as having problems, but specify the suspected line. If the second system also has trouble connecting over the specified line, check the hardware.

> For information about using the **cu** command, see the BNU chapter in *AIX Operating System Commands Reference*. For information about the command itself, see the **cu** entry in *AIX Operating System Commands Reference*.
>
> For information about the hardware involved in establishing a remot connection, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.
>
> For information about ACUs, see "Setting Up Devices" in topic 6.8.5.2.1.

If users on specific local systems at your site are having problems communicating with specific remote systems using modem connections, you should also make certain that the configuration files on all the involved systems are set up correctly. For information about configuring these files, see "Setting Up Remote Communication" in topic 6.8.5.2.

¦ Copyright IBM Corp. 1985, 1991

6.8.9.5 Login Failures

Two of the most common reasons for login failures result from the following:

The appropriate tty device has either not been created at all, or ha not been created properly.

The expect-send sequence in the login field in the **Systems** file needs additional data in order to establish a reliable connection to a certain remote system.

Subtopics
6.8.9.5.1 Problems with a tty Device
6.8.9.5.2 Problems with an expect-send Sequence
6.8.9.5.3 Troubleshooting Connection Problems

¦ Copyright IBM Corp. 1985, 1991

6.8.9.5.1 Problems with a tty Device

If the tty device has not been created, BNU will probably display the following error message when a user attempts to contact a remote system:

```
Call failed:  NO DEVICES AVAILABLE
```

If the tty device has not been properly created, BNU will probably display this error message:

```
Call failed:  CAN'T ACCESS DEVICE
```

You then need to perform one of the following actions:

1. If the device does not exist, create the appropriate tty device with the **devices** command.

2. If the device has not been created properly, check the value of the **dvam** parameter for that tty and, if necessary, change it with the **devices** command.

3. If the proper device exists, either the direct or the modem connection may be wrong. Depending on the configuration, check either or both of these connections, and then check the configuration files on both the local and the remote systems.

For detailed information about these operations, see the following:

For information about the **devices** command, see the description of that command in *AIX Operating System Commands Reference*.

For information about creating and managing the devices involved i remote communications, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

For information about debugging connection problems, se "Monitoring a File Transfer" in topic 6.8.6.1.4 and "Troubleshooting Connection Problems" in topic 6.8.9.5.3.

¦ Copyright IBM Corp. 1985, 1991

6.8.9.5.2 Problems with an expect-send Sequence

You may occasionally need to change the **expect-send** character strings in the **Login** field in the **Systems** file, depending on the way in which the local system or the remote system is initialized.

Note: If you are having problems that appear to be related to expect-send characters, you can manually start the **uucico** daemon with the **-x** (debug) flag to monitor the communication, as described in "Monitoring a File Transfer" in topic 6.8.6.1.4 and "Troubleshooting Connection Problems" in topic 6.8.9.5.3.

If you or your users experience login failures when trying to connect to a remote computer specified correctly in the **Systems** file, you may need to add several additional characters to the appropriate line in the entry representing that system. These additional characters precede the data included in the login field. You may find it useful to use the **cu** command to connect to the modem without dialing. This allows you to experiment with the dialing login sequences necessary.

Suppose your **Systems** file contains the following entry:

```
hera  Any  hera  1200  -  in:--in: uucp  word: sysuucp
```

This specifies a hardwired connection to system **hera** (entered in both the system-name and caller fields) at a transmission rate of **1200** bps. The first expect string, **in:--in:**, contains the login prompt issued by **hera**, and the first send string contains the **uucp** program login ID sent by the local system. The second expect string contains **hera**'s password prompt, represented by **word**, and the second send string contains the local system's response, **sysuucp**.

If this entry is producing login failures for which you cannot account in any other way, try including one of the following standard strings in the relevant line in the **Systems** file:

1. **"" \n**

2. **"" \r\d\r\d\r**

Add one of the strings to the entry and see if that solves the problem. If it does not, delete that string, add the other string, and try again. Your entry in the **Systems** file will look like one of the following examples:

```
hera  Any  hera  1200  -  "" \n in:--in: uucp  word:  sysuucp
```

or

```
hera  Any  hera  1200  -  "" \r\d\r\d\r  in:--in: uucp  word: sysuucp
```

If you continue to have login problems, check the appropriate entries in the **Devices**, **Systems**, and **Permissions** files to make certain that the remote system is correctly specified.

For additional information about these files, see the following:

"Setting Up Devices" in topic 6.8.5.2.1

"Customizing the Systems File" in topic 6.8.5.2.3

¦ Copyright IBM Corp. 1985, 1991

"Customizing the Permissions File" in topic 6.8.5.2.5.

For information about debugging file-transfer and connection problems, see "Monitoring a File Transfer" in topic 6.8.6.1.4 and "Troubleshooting Connection Problems" in topic 6.8.9.5.3.

¦ Copyright IBM Corp. 1985, 1991

6.8.9.5.3 Troubleshooting Connection Problems

As described in "Monitoring a File Transfer" in topic 6.8.6.1.4, you use a form of the **uucico** command to debug a connection problem:

```
/usr/lib/uucp/uucico -r1 -ssystem_name -x9
```

where **-r1** specifies the server mode, **-ssystem_name** is the name of the remote system to which you are trying to connect, and **-x9** specifies the highest debug level, which produces the most detailed debugging information.

Expect-send debugging output produced by the **uucico** command can come either from information in the **Dialers** file, or from information in the **Systems** file.

Note: To set up a connection with a remote system, you must be familiar with the login sequence of that system. For detailed information about expect-send sequences, see "Expect-Send Characters in Login Fields" in topic 6.8.5.2.3.

Problems in the Dialers File: When establishing a connection between a local and a remote system using a telephone line and modem, BNU consults the **Dialers** file. (BNU also checks the **Systems** file to make sure it contains a listing for the specified remote computer.) Each entry in the **Dialers** file is a line that contains a series of expect-send sequences, like the following:

```
hayes   =,-,  ""  \dAT\r\c OK \pATDT\T\r\c CONNECT
```

This example specifies a Hayes modem:

The **=,-** notation translates the telephone number: any **=** represents wait for dial tone, and any **-** represents pause.

The **""** notation tells the modem to wait for nothing; continue with the rest of the string.

The **\dAT** notation instructs the modem to delay for **AT** (the Hayes Attention prefix).

The remainder of the line in the **Dialers** file for the Hayes modem contains similar information. If the line in the **Dialers** file is not set up correctly for the specified modem, BNU will probably display the following error message:

```
DIALER SCRIPT FAILED
```

1. First, make sure that both the local and the remote modems are turned on, that they are both set up correctly, and that the telephone number of the remote modem is correct.

2. Then check the **Dialers** file and make sure the information is correctly specified for the local modem. If possible, you should also check the **Dialers** file on the remote system.

3. If you continue to have connection problems, check the documentation that came with your modem to make sure you have used the correct expect-send sequence characters in the **Dialers** file.

¦ Copyright IBM Corp. 1985, 1991

For detailed information about the hardware used in establishin remote communications, see "Managing Ports, Cables, and Modems on a PS/2" in topic 5.6.

For detailed information about configuring the **Dialers** file, see "Setting Up Dialers" in topic 6.8.5.2.2.

Problems in the Systems File: When establishing any type of connection, BNU consults the **Systems** file, which specifies the way in which the local computer communicates with a designated remote computer.

For example, suppose your local **Systems** file contains the following entry for communicating with the remote system **venus**:

```
venus  Any  venus  1200  -  "" \n in:--in: uucp1  word: passuucp
```

If users on the local system are having trouble connecting to **venus**, you can issue the **uucico** command with the debugging flag to produce debugging information about the connection:

```
/usr/lib/uucp/uucico -r1 -svenus -x9
```

Note: When entered with the appropriate flag, the **ct** and **cu** commands also produce debugging information about remote connections.

Following is an example of the debugging output produced when the **uucico** command (or **ct** or **cu**) uses the entry in the **Systems** file shown above to connect the local system to the remote system **venus**:

```
expect:  ""
got it
sendthem (^J^M)
expect (in:)
^M^Jlogin:got it
sendthem (uucp1^M)
expect (word:)
^M^JPassword:got it
sendthem (passuucp^M)
imsg >^M^J^PShere^@Login Successful:  System=venus
```

These entries represent the following information:

expect: ""	The local system should not wait for any information from the remote system.
got it	Acknowledgement.
sendthem (^J^M)	The local system sends the remote system a carriage return and a new line.
expect (in:)	The local system expects to receive the remote system login prompt, which ends in the character string **in:**.
^M^Jlogin:got it	The local system receives the remote login prompt.
sendthem (uucp1^M)	The local system sends the **uucp1** login ID to the remote system.

¦ Copyright IBM Corp. 1985, 1991

expect (word:) The local system expects to receive the remote system password prompt, which ends in the character string **word:**.

^M^JPassword:got it The local system receives the remote password prompt.

sendthem (passuucp^M) The local system sends the password for the **uucp1** login ID to the remote system.

imsg >^M^J^PShere^@Login Successful: System=venus
The local system is successfully logged in to the remote system **venus**.

¦ Copyright IBM Corp. 1985, 1991

6.9 Chapter 9. Overview of the Message Handling Package

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.9.1 Contents

¦ Copyright IBM Corp. 1985, 1991

6.9.2 About This Chapter

This chapter provides a brief description of the files, commands, and features of the Message Handling (MH) package. This chapter also describes some of the managing tasks associated with using the MH package.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System*, and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

6.9.3 Understanding the Message Handling Package

The Message Handling (MH) package contains a collection of commands that enable you to create, distribute, receive, view, process, and store messages. You can also verify user IDs, address formats, and date formats. You can run MH commands as separate programs, use MH commands with other MH and AIX commands, and create programs that call MH commands.

The MH package uses the AIX file and directory system. MH commands store messages as individual files with numerical file names, and group messages into directories. A directory of messages is called a ***folder***. Each folder contains some standard files to support the MH functions, as well as its actual messages. Since the messages are AIX files and the folders are AIX directories, you can use the AIX commands to manage messages and folders. For example, you can use the **chmod** command to change the access permissions on a folder, or you can run **cron** to purge old or inactive messages from a group of folders.

Note: If you want other users in your cluster to read your messages, file names, directory names, and characters, use ASCII characters only.

The MH package does not provide a message transport facility. MH commands provide input to the **sendmail** command, and rely on the transport facilities associated with **sendmail**.

All restrictions regarding MBCS and ASCII conventions apply to Message Handling. See Chapter 6, "Managing the Electronic Mail System" in topic 6.6 for more information.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.9.3.1 Standard Directories

Files supporting the MH package are stored in several directories. The following paragraphs describe these directories and the types of MH files that they contain:

Directory	Description
/usr/lib/mh	Contains files that describe the system's MH defaults. You can alter many of these files to set your own system defaults.
$HOME	Contains the user's MH profile **.mh_profile**. Users can modify their own **.mh_profile** to set their own MH defaults.
user_mh_directory	Contains the user's message folders and usually the user's **context** file. The user can choose any name for this directory. The chosen name will be shown as an entry in the user's **mh.profile** file.

¦ Copyright IBM Corp. 1985, 1991

6.9.3.2 Files in the Supporting Database

The following files are used by the MH system:

File	Description
$HOME/.mh_profile	The MH user profile.
user_mh_directory/**draft**	The default draft file.
$HOME/.newmail	The location of the user's mail drop.
/usr/lib/mh/maildelivery	The default local delivery instructions file.
$HOME/.maildelivery	The user's local delivery instructions file. This file must have permissions set so that it is owned, readable, and writable by the user.
/usr/lib/mh/MailAliases	The default mail alias file.
/usr/lib/mh/mtstailor	The MH tailor file.
$HOME/.forward	The user's local mail forwarding file. This file must have permissions set so that it is owned, readable, and writable by the user.
/usr/lib/mh/components	The default message form for the **comp** command. Each user can create a default message form and store that form in the file **user_mh_directory/components**.
/usr/lib/mh/distcomps	The default message skeleton for the **dist** command. Each user can create a default message skeleton and store that message skeleton in the file **user_mh_directory/distcomps**.
/usr/lib/mh/forwcomps	The default message skeleton for the **forw** command. Each user can create a default message skeleton and store that message skeleton in the file **user_mh_directory/forwcomps**.
/usr/lib/mh/digestcomps	The default message skeleton for the **forw** command when the **-digest** flag is specified. Each user can create a default message skeleton for digests and store that message skeleton in the file **user_mh_directory/digestcomps**.
/usr/lib/mh/mhl.forward	The default message filter for the **forw** command when the **-format** flag is specified. Each user can create a default filter and store that filter in the file **user_mh_directory/mhl.forward**.
/usr/lib/mh/mhl.format	The default message template. Each user can create a default message template and store that template in the file **user_mh_directory/mhl.format**.

¦ Copyright IBM Corp. 1985, 1991

/usr/lib/mh/replcomps	The default reply template. Each user can create a default reply template and store that template in the file **user_mh_directory/replcomps**.
/tmp/prompter*	A temporary copy of a message. This file is created when the editor **prompter** is used.

¦ Copyright IBM Corp. 1985, 1991

6.9.3.3 User Commands

The following commands are the user-level programs in the MH package package:

Command	Description
ali	Lists mail aliases and their addresses.
anno	Annotates messages.
ap	Parses and reformats dates.
burst	Explodes digests into messages.
comp	Composes a message.
dist	Redistributes a message to additional addresses.
dp	Parses and reformats addresses.
folder	Selects and lists folders and messages.
folders	Lists folders and messages.
forw	Forwards messages.
Command	**Description**
inc	Incorporates new mail.
mark	Creates, modifies, and displays message sequences.
mhl	Produces formatted listings of messages.
mhmail	Sends or receives mail.
mhpath	Prints full path names of messages and folders.
msgchk	Checks for messages.
msh	Creates an MH shell.
next	Shows the next message.
packf	Compresses the contents of a folder into a file.
pick	Selects messages by content, and creates and modifies sequences.
prev	Shows the previous message.
refile	Files messages in other folders.
repl	Replies to a message.
rmf	Removes a folder.
rmm	Removes messages.
scan	Produces a one line per message scan listing.
send	Sends a message.
show	Shows messages.
sortm	Sorts messages.
vmh	Invokes a visual interface for use with MH commands.
whatnow	Invokes a prompting interface for draft disposition.
whom	Lists the addresses of the proposed recipients of a message and verifies the addresses.

AIX Operating System Commands Reference provides a detailed description for each of these MH commands.

¦ Copyright IBM Corp. 1985, 1991

6.9.3.4 Other MH Commands

The following MH commands can be used by other programs but should not be run directly by users:

Command	Description
conflict	Searches for alias and password conflicts.
install-mh	Initializes the MH environment.
post	Delivers a message.
prompter	Invokes a prompting editor.
rcvdist	Sends a copy of incoming messages to additional recipients.
rcvpack	Saves incoming messages in a packed file.
rcvstore	Incorporates new mail from standard input into a folder.
rcvtty	Notifies the user of incoming messages.
slocal	Processes incoming mail.
spost	Delivers a message.

AIX Operating System Commands Reference provides a detailed description for each of these MH commands.

¦ Copyright IBM Corp. 1985, 1991

6.9.3.5 Message Components

Each message contains a ***header*** and a ***body***. The header provides date, address, and subject information. The body provides the text of the message.

The ***header*** can contain various ***components***, or fields. These components may be generated by such sources as the MH package (for example, **From:**), the message transport system (for example, **Date:**), the sender (for example, **To:**), and the recipient (for example, **Replied:**). Each header component starts with the name of the component and is followed by a colon and the information in that component. If the component information requires more that one line, you must begin each continuation line with a space character. The first line of a component can not begin with a space character. The following list describes some of the header components:

Header Component	**Description**
To:	Lists the addresses or aliases of the primary recipients of the message.
cc:	Lists the addresses or aliases of the "carbon copy" recipients of the message.
Bcc:	Lists the addresses or aliases of the "blind carbon copy" recipients of the message. Although persons listed in this component receive a copy of the message, the MH package removes this component from the sent copy of the message.
Subject:	Describes the subject or contents of the message.
Date:	Reports the date the message was sent.
From:	Reports the sender of the message.

The **mh-mail** section in *AIX Operating System Technical Reference* contains a more complete list of header components.

The ***body*** of a message consists of the text of the message. The body follows the header. While you compose a message, a line of dashes separate the header from the body. Do not remove this separator line. When you send the message, the MH package converts this line of dashes to an empty line.

Note: A message can contain MBCS characters, but the end user will be able to read it only if the same locale exists as the sender.

¦ Copyright IBM Corp. 1985, 1991

6.9.4 Installing the MH Package

The MH package files are part of the Extended Services feature of the AIX Operating System. For information about how to install Extended Services programs, refer to *Installing and Customizing the AIX Operating System* manual for your machine.

The first time each user runs a MH command, that command calls **install-mh**, which creates a MH profile for that user. See the **install-mh** section in *AIX Operating System Commands Reference* for more information about this command.

¦ Copyright IBM Corp. 1985, 1991

6.9.5 Performing Routine Maintenance Tasks

The MH package requires that you perform a few routine maintenance tasks. Since the MH package creates backup files, you will need to remove files periodically. You may also want to verify addresses and check alias files for duplicate aliases. The following sections discuss these maintenance tasks.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.9.5.1 Removing Messages and Folders

The MH package provides you with two commands for removing messages and folders. The **rmm** command removes messages, and the **rmf** command removes folders.

The **rmm** command does not actually delete messages. It renames messages by placing a comma (**,**) in front of the message file names. You can still use AIX commands to manipulate these file, but the comma makes the messages unavailable to the MH package. Periodically you should delete these files. You can place an entry in your **crontab** file and use the **crontab** command to automatically delete all files having file names beginning with a comma. The **rmf** command removes folders, including all the files within folders.

Since messages are AIX files and folders are AIX directories, you can use AIX commands (such as **rmdir**, and **rm**, and **del**) to remove messages and folders.

See *AIX Operating System Commands Reference* for more information about the commands: **rmf**, **rmm**, **rmdir**, **rm**, **del**, and **crontab**.

¦ Copyright IBM Corp. 1985, 1991

6.9.5.2 Checking for Invalid Addresses

The MH command **whom** expands address headers into sets of addresses and optionally verifies whether the addresses are valid. A valid address is an address that has a format acceptable to the mail transport system. Valid address headers do not imply that a message is deliverable. See the **whom** section in *AIX Operating System Commands Reference* for more information.

¦ Copyright IBM Corp. 1985, 1991

6.9.5.3 Checking for Duplicate Aliases and for Inappropriate Mail Drops

The MH command **conflict** checks for duplicate aliases and for inappropriate mail drops. **conflict** is not a user command, but is designed to be run by other programs (such as **mhmail**). See the **conflict** section of *AIX Operating System Commands Reference* for more information.

¦ Copyright IBM Corp. 1985, 1991

6.9.6 Understanding MH Defaults

The MH package has a multi-layered default structure. When you enter a MH command, the system searches for arguments and default settings in the following sequence:

1. The system accepts all valid arguments and flags from the command line (or from wherever the command was initiated).

2. The system searches **$HOME/.mh_profile** for the entry **command:**, where **command** is the name by which the program was invoked. If this profile entry exists, the system accepts all arguments that do not cancel the previously stated command line instructions.

   Note: If you invoke a MH command by specifying a link to the command, the system searches for the profile entry having that link as its name. If you invoke the command by specifying an alias, the system searches for the profile entry having the actual command as its name, rather than the alias as its name.

3. The system searches **$HOME/.mh_profile** for other entries that are applicable to the command and that have not already been overridden by previously stated arguments. If any such entries exist, the system accepts the arguments to those entries.

4. The system uses the current state information (such as the current folder and the current message) recorded in *user_mh_directory*/**context** for any arguments that have not been stated.

5. The system accepts system-wide defaults for any remaining unspecified arguments.

When you specify a relative path name for your MH profile, the MH programs interpret the full path name as beginning at the current directory. If you specify a relative path name for any other MH files, the MH programs interpret the full path names for those files as beginning at **user_mh_directory**.

¦ Copyright IBM Corp. 1985, 1991

6.9.7 Tailoring MH Profiles and Format Files

The MH package allows you to tailor MH system files and user files. The following sections provide information about tailoring MH files.

Subtopics
6.9.7.1 The User Profile
6.9.7.2 Format Files

¦ Copyright IBM Corp. 1985, 1991

6.9.7.1 The User Profile

When you first run an MH command, MH creates the file **.mh_profile** in your **$HOME** directory. You can create personal MH default values by modifying this file. The following list shows some of the MH values that can be set in **$HOME/.mh_profile:**

Profile Entry	**Description of the Entry**
Path:	Sets the user directory for MH transactions. This entry must appear in **$HOME/.mh_profile**. When an MH command creates **$HOME/.mh_profile**, the MH system automatically creates a **Path:** entry with the default value **Mail**. Throughout the MH documentation, **user_mh_directory** refers to the value of the **Path:** entry.
context:	Sets the name of the MH context file. Unless a full path name is specified, this file resides in **user_mh_directory**. The default file name is **user_mh_directory/context**.
Editor:	Specifies the initial editor to be used by the **comp**, **dist**, **forw**, and **repl** commands. The default editor is **prompter**.
Msg-Protect:	Sets the protection level for message files. The protection level must be specified in octal form. The default is **644**.
Folder-Protect:	Sets the protection level for folder directories. The protection level must be specified in octal form. The default is **711**.
cmd:	Specifies the flags to be used whenever the MH command **cmd** is run. You can use this entry to override **.mh_profile** settings for particular commands. For example, the following entry specifies that the **comp** command is to invoke the **e** (INed editor) command regardless of the editor specified in the **Editor:** entry: comp: -editor e
lasteditor-next:	Specifies the editor to be used for re-editing a draft after initial editing with **lasteditor**. When **lasteditor** is used as the initial editor by the **comp**, **dist**, **forw**, and **repl** commands, the default editor for re-editing the draft is the editor specified in this profile entry. This editor takes effect at the **whatnow** level of the **comp**, **dist**, **forw**, and **repl** commands. For example, the following profile entry states that the **e** (INed editor) command is the default re-editing command to be invoked for drafts initially edited using **prompter**. prompter-next: e
Alternate-Mailboxes:	Specifies the addresses that belong to you. The **repl**

¦ Copyright IBM Corp. 1985, 1991

command uses this profile entry to determine which addresses to include in replies. The **scan** command uses this entry to determine which messages originated from you. When specifying addresses in this entry, you must separate each address with a comma and list the official host names for the addresses (local nicknames are not resolved to their official host names). If you do not specify the host name for an address, **repl** and **scan** regard that address on all host systems as belonging to you. The asterisk (*****) can be used at either or both ends of the host name and the mailbox to indicate wild-card matching. The default is your user ID.

Draft-Folder: Specifies the default draft folder for the **comp**, **dist**, **forw**, and **repl** commands.

MailDrop: Specifies your maildrop for use with the **inc** command. The **$MAILDROP** environment variable supersedes this profile entry. The default is **/usr/mail/$USER**.

Signature: Specifies your mail signature for use with the **send** command. The **$SIGNATURE** environment variable supersedes this profile entry.

See the **mh-profile** section in *AIX Operating System Technical Reference* for a more information about the **.mh_profile** and a more complete listing of the profile entries that you can define.

Entries in the **.mh_profile** file should be only those that maintain static values. Default values for entries such as **current_folder** should not appear in this file, since they are updated dynamically and appear in the **context** file.

¦ Copyright IBM Corp. 1985, 1991

6.9.7.2 Format Files

The MH package enables you to maintain multiple formats for different types of messages. You can maintain formats for new messages, for redistributed messages, for replies to messages, for forwarded messages, and for message digests. You can create and modify system formats and personal formats. The section **mh-format** in *AIX Operating System Technical Reference* describes the MH format strings that you can place in a format file.

¦ Copyright IBM Corp. 1985, 1991

6.9.8 Using Special Features of the MH Package

The MH package provides you with several special features. The following sections discuss two of these features: your ability to specify messages in many ways and your ability to maintain drafts of messages.

Subtopics
6.9.8.1 Specifying Messages
6.9.8.2 Creating and Using Message Drafts

¦ Copyright IBM Corp. 1985, 1991

6.9.8.1 Specifying Messages

Many MH commands allow you refer to particular messages. You can specify messages in the following ways:

By stating the numerical name of the message. When you store message in a folder, the MH package provides the message with a numerical file name. You can use this numerical name when referring to that message.

By stating a MH keyword that indicates a message. You can use th following keywords to specify messages:

Keyword	Description
first	The first message in the folder. The lowest message number is **1**, but a folder does not have to contain a message by this number.
prev	The message that is numerically previous to the current message.
cur	The current message.
.	The current message.
next	The message that is numerically next after the current message.
last	The last message in the folder.
new	A new message. This is equivalent to the message following the **last** message in the folder. You can not use this keyword when specifying a range of messages
all	All of the messages in the folder.

For each of these keywords, if the folder is empty, the keyword represents message **1**.

By stating the sequence name that represents one or more messages Each sequence is associated with a particular folder and can be used to refer to messages in that folder only. You can specify message sequences that are designated as public sequences, message sequences that you define, and message sequences that the MH package system keeps track of for you.

By describing a range of messages

Subtopics
6.9.8.1.1 Defining a Message Sequence
6.9.8.1.2 Using Sequences Defined by the MH Package
6.9.8.1.3 Specifying a Range of Messages

¦ Copyright IBM Corp. 1985, 1991

6.9.8.1.1 Defining a Message Sequence

You can use the **pick** and **mark** commands to define a message sequence. The name you choose for a sequence must contain ASCII alphabetic characters only and can not conflict with one of the MH keywords. For example, you can define a sequence named **frombrenda**, but you can not define a sequence named **new**. You can define a maximum of about 10 sequences for each folder.

See the **pick** and **mark** sections of the *AIX Operating System Commands Reference* for more information about defining message sequences.

¦ Copyright IBM Corp. 1985, 1991

6.9.8.1.2 Using Sequences Defined by the MH Package

The MH package optionally keeps track of the messages that you previously specified and the messages that you have incorporated into a folder, but that you apparently have not seen. You can specify that you want the MH package to keep track of these messages by defining the entries **Previous-Sequence:** and **Unseen-Sequence:** in your **.mh_profile**.

For example, you can place the following definitions in **$HOME/.mh_profile**:

```
Previous-Sequence:  pseq
Unseen-Sequence:  unseen
```

These definitions enable you to refer to the sequences **pseq** and **unseen** for each of your folders. The sequence **pseq** refers to the last message or sequence of messages that you specified. The sequence **unseen** refers to the messages for which you have run the **inc** command, but for which you have not run the **show** command.

¦ Copyright IBM Corp. 1985, 1991

6.9.8.1.3 Specifying a Range of Messages

You can specify a range of messages in the following ways:

By specifying the explicit range of the messages in the sequence. Yo can specify an explicit range by placing a dash (**-**) between the message identifier beginning the range and the message identifier ending the range. For example, the following command requests a scan listing of messages **9** through **last** inclusive:

```
scan 9-last
```

When you specify an explicit range, the range must contain messages.

By specifying the general range of messages in the sequence. You ca specify a general range by placing a colon (**:**) between the message identifier beginning or ending the range and the number of messages in the range. If the message identifier is **prev** or **last**, the message identifier ends the range. Otherwise, the message identifier begins the range. For example, the following command requests a scan listing of the last five messages:

```
scan last:5
```

The following example requests a scan listing of the first three messages:

```
scan first:3
```

You can use the plus (**+**) and minus (**-**) symbols to change (or explicitly state) the direction of the range. The **+** directly following the **:** indicates that the range begins with the message identifier. For example, the following command requests a scan listing of the message **prev** and the three messages following **prev**:

```
scan prev:+4
```

The **-** directly following the **:** indicates that the range ends with the message identifier. For example, the following command requests a scan listing for the five messages preceding message **23** and for message **23**:

```
scan 23:-6
```

You can not use the keyword **new** when specifying a range. If you specify a message that is greater than the **last** message in the folder, the MH package interprets that message as the message number following **last**.

¦ Copyright IBM Corp. 1985, 1991

6.9.8.2 Creating and Using Message Drafts

When you create a message using the commands **comp**, **dist**, **forw**, and **repl**, the MH package enables you to save the message in a folder until you are ready to send the message. You can specify a name for this folder using the **-draftfolder** flag. Until you send the message, the message is considered a ***message draft***. When you send the message, the MH package renames the message draft by placing a comma before its file name. This renaming of the message removes the message from active draft status. (The profile entry **rmmproc:** is not consulted for this automatic renaming.)

A message draft is similar to other MH messages. You can store several message drafts in a folder, you can create sequences containing message drafts, and you can run MH commands (such as **refile**, **rmm**, and **scan**) with a message drafts as an argument.

The **-draftfolder** flag does not change the current folder or the current message.

¦ Copyright IBM Corp. 1985, 1991

6.10 Chapter 10. Managing the IBM AIX Network File System

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.10.1 Contents

¦ Copyright IBM Corp. 1985, 1991

6.10.2 About This Chapter

The IBM AIX Network File System (NFS) is a distributed file service that allows users to share files with computers in networks that include a variety of machines and operating systems. This chapter provides the information you need to install and manage the Network File System software. It includes the following information:

Software and hardware required to install and use the Network Fil System program

What the Network File System program is and how it work

The Network File System program software components (NFS and NIS) an what they do

How to install the Network File System progra

How to configure the NFS software componen

How to configure and use the Network Information Services (NIS network service.

The AIX Operating System features Multibyte Character Set (MBCS) support. For more information on MBCS capabilities, see information in this publication, the *Guide to Multibyte Character Set (MBCS) Support*, *Using the AIX Operating System,* and other references in the AIX document set.

¦ Copyright IBM Corp. 1985, 1991

6.10.3 Information Requirements

Before reading about the Network File System, it is important that you understand the principles of AIX system management described in the following topics:

"The File System - Background for System Management" in topic 1.1.4 and "The Base AIX File Systems" in topic 1.1.5

"Managing User Accounts" in topic 1.2.5

"Information about File Systems - The /etc/filesystems File" in topic 1.2.7

"Backing up Files and File Systems on AIX PS/2" in topic 1.2.9

Principles of remote communication using Transmission Contro Protocol/Internet Protocol (TCP/IP). See *AIX TCP/IP User's Guide*.

Adding machines to a communication network. See the section o communication planning in *AIX/370 Planning Guide*.

¦ Copyright IBM Corp. 1985, 1991

6.10.4 Overview of Hardware and Software Required for the Network File System

Before you can install the Network File System program, the following software and hardware must be installed and running on each system that is to be on the network:

Software

IBM AIX PS/2 Operating System, Version 1.2.1

IBM AIX/370 Operating System, Version 1.2.1

IBM Transmission Control Protocol/Internet Protocol (TCP/IP) program

Hardware

A network adapter for use with Ethernet or Token-Ring
Cables and connectors.
An ARTIC card

Note: NFS is also supported over X.25 data network. See the documentation packaged with that LPP for more specific information.

Refer to the following publications for information about installing the software and hardware:

Installing and Customizing the AIX PS/2 Operating System
Installing and Customizing the AIX/370 Operating System
AIX/370 Planning Guide.
AIX TCP/IP User's Guide

¦ Copyright IBM Corp. 1985, 1991

6.10.5 Overview of the IBM AIX Network File System

The IBM AIX Network File System (NFS) is a distributed file system that allows users to access files located on remote computers in the network. With the Network File System program, users can share files and have access to additional disk space.

The computers that make their files and resources available for remote access are called ***servers***. The computers (or the processes they run) that access the remote resources are considered ***clients***. A computer can be both a server and a client in the Network File System environment.

The IBM AIX Network File System program contains two software components, NFS and Network Information Services (NIS). They are installed together as a single package.

NFS file systems use their own protocol. NFS provides a transparent link for both file names and file contents. Extended-character files can be freely exchanged between systems.

The entire composite minidisk or AIX is included in the Global File System (gfs).

NFS software is distributed by IBM under a cluster-wide licensing agreement. Thus all sites in an AIX cluster will typically be running AIX with TCF (if purchased) and all will normally run NFS. A network, however, may also contain machines running operating systems other than AIX.

AIX machines that run NFS software have the capacity to share file systems with any machines that are running NFS, even those running under some other operating system. Any machine which is NFS licensed and running the NFS software has the ability to function as an NFS client for any file system, anywhere in the network, local or remote. It can also function as an NFS server, offering its own file systems to others.

NFS is a feature found in several operating systems, and it allows otherwise dissimilar operating systems to share file systems. If the AIX cluster has network connections to a remote machine running under some other operating system that offers NFS, that machine may act as an NFS server, allowing its file system(s) to be used by the AIX cluster.

An AIX cluster behaves as if it were a single machine, running in either client or server mode. Mounting a remote file system on any site in the cluster gives every site in the cluster access to that file system. When any remote machine mounts the shared TCF file system from any site in the cluster, it gains access to all files mounted throughout the cluster.

In actual use, every site in the AIX cluster or network has access to a single, large shared file system, which functions as if it were present on each site that is operating.

File locking: When you are using NFS a greater possibility exists of two or more processes modifying a file concurrently. Use *file locking* to ensure that changes are made to a file without outside interference. The AIX Operating System allows many processes simultaneous access to a file through the use of the **fentl** and **lockf** system calls. These system calls allow a program to lock and unlock portions of an open file. For more information, see the *AIX Programming Tools and Interfaces* (Chapter 8, "Using File Locking") and *AIX Technical Reference* (the **fcntl**, **flock**, **lockf**

¦ Copyright IBM Corp. 1985, 1991

commands).

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.10.5.1 gfs Numbers and Synchronization in a TCF Environment

In AIX, every mounted file system is identified by the global file system (gfs) number. In a normal file system, this number is hardcoded in the superblock of the physical device where that file system actually resides. For a remote NFS file system, however, there is no such hardcoding; an arbitrary assignment is made by a System Administrator.

Normally, this gfs number is assigned by writing it into the appropriate stanza of the **/etc/filesystems** file of the machine that performs the mount. The **/etc/filesystems** file is a replicated appendage to the root, but it is a symbolic link to **<LOCAL>/filesystems**; therefore, every site will have its own, unique copy. The System Administrator alters the local copy of this file to indicate which file system(s) are to be mounted by that site. Each such file system is represented by a separate stanza in **/etc/filesystems**.

This gfs number can also be automatically assigned by the system itself. Automatic assignment of gfs number takes place whenever a file system that has no such number is mounted by an NFS client. Thus, every NFS file system in the AIX cluster that either has been mounted or is considered important enough that other sites might want to mount it, will have a unique identifying number. This way there can be no confusion amongst them.

This numbering system, however, is relevant only to NFS clients in the AIX cluster. It is not seen by remote sites outside the AIX cluster, and remote systems that are not running AIX with NFS have no such numbering system. The gfs numbering system is a client-only, AIX-specific concept. It is unknown outside the AIX cluster, and even unknown to NFS servers within the cluster. The client takes care of all such record keeping, allowing the server to be stateless. A more thorough discussion of statelessness will be found later in this chapter.

The System Administrator is responsible for assigning a gfs number to every file system which needs to be mounted by more than one site. This unique number is present in the **/etc/filesystems** file of every machine authorized to mount that file system.

There may, however, be other machines connected to the AIX cluster which are not running AIX but are running NFS. These machines may act as NFS servers for the AIX cluster, and cluster sites may act as NFS servers for those remote machines. Thus file systems may be shared universally but only by machines running NFS software.

A key problem with NFS file system utilization is the lack of inter-client synchronization. Because each client caches data, a given client may have an out-dated view of a file. A TCF cluster behaves as if it were a single client. AIX ensures that each site has the most recent version of the file. This is done by declaring one of the sites as the Current Synchronization Site (CSS). This site is responsible for alerting other sites within the cluster that a file they may be caching is out of date.

The operating system distinguishes every file system from every other by its gfs number. Each mounted file system in a TCF environment is assigned a particular machine to serve as its CSS. If any file system should somehow be assigned two gfs numbers, it would be considered two different file systems by the operating system and might well be entrusted to two separate CSS's. Even if the same machine should be assigned as CSS, it would treat these two gfs numbers as separate file systems and synchronization would fail, just as it would fail if the file system were

¦ Copyright IBM Corp. 1985, 1991

mounted by two unrelated NFS client machines.

File systems from remote NFS machines may be available for mounting in the AIX cluster, but, for one reason or another, gfs numbers may not have been preassigned to them. If any two sites should perform separate concurrent mounts on such a remote NFS file system, each mounting it on separate mount points, the operating system would assign separate gfs numbers for each mount. It would then have no way to know that this is exactly the same file system and would have no way to assure that the changes being made to it by the two sites were synchronized.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.2 Two Types of Mounts Allowed by NFS in a TCF Environment

It, therefore, becomes necessary to introduce the following distinction. An NFS client machine may perform two types of mounts: administered and nonadministered.

Administered vs. nonadministered mounts is a client-only concept. It pertains to the way the client machines coordinate their use of the file system. The gfs number is written into the **/etc/filesystems** file of the client only. The NFS server does not see this number or know of its existence.

An administered mount is the mounting of a file system which has had a permanent, unique gfs number assigned by a System Administrator. The client machine performing such a mount will be one of the AIX sites running NFS software. The file system being mounted may be anywhere, within the AIX cluster or outside it.

A nonadministered mount has had no such number previously assigned. Once again, the client machine performing such a mount is one of the AIX sites; we know this because administered vs. nonadministered is, by definition, a client-only, cluster-oriented concept. The file system being mounted may be a remote NFS file system, one that resides on a machine outside the AIX cluster, possibly running some other operating system. The file system may get sufficiently little usage that no number has ever been assigned. It may even be a file system from some other site within the AIX cluster, perhaps a file system newly created or for which the system Administrator did not anticipate a demand.

No matter what its origin, the system searches for a gfs number in the **/etc/filesystems** file of the machine attempting the mount. The system finds either no gfs number or the number 0, and automatically assigns its own.

When any file system is mounted, whether its gfs Number has been administered or assigned by the system, AIX uses the gfs number to uniquely identify it. The operating system immediately replicates a record of this mount throughout the cluster. Every machine in the cluster may then access that file system. The file system appears to them to be simply another part of the overall file tree of the AIX cluster.

Subtopics
6.10.5.2.1 Administered Mounts
6.10.5.2.2 Nonadministered Mounts

¦ Copyright IBM Corp. 1985, 1991

6.10.5.2.1 Administered Mounts

When a client machine mounts a file system which has had a gfs number assigned to it, this is considered an administered mount.

The stability of this access will vary, however, according to the kind and number of mounts that have been done on that file system. There are two distinct mounting scenarios.

First, each site that wants access to the material can perform its own independent mount, direct from the server where the NFS file system actually resides.

Second, each site can gain access via a single mount done by a single machine. That is, once a single AIX site, acting as client, has mounted the file system, all others in the cluster may use those files just as they would use any other segment of the overall network file tree. The gfs number of the file system has been copied throughout the mount tables of all sites in the cluster, and it seems to be simply another segment of the shared file system that appears to be available cluster-wide.

This second method, which could be called "access by intermediary," is easy enough to do but offers less than optimum stability. If the machine that actually does the mount should go down, all other units trying to access that material would lose their pathway to it. The directory on which those files had been mounted would remain, but attempts to read or write to the files would fail. A listing of files would show an empty directory. Optimum stability is achieved by multiple administered mounts.

When the mounts have been administered ahead of time by a System Administrator, each machine in the AIX cluster which has had its **/etc/filesystems** file so altered is permitted to mount that file system for itself. The latter option makes its access more secure, at least in the sense that it becomes tolerant of the loss of the machine that first performed the mount.

This multiple mounting of a single file system does not establish new and separate software paths to the mounted file system. It simply sets up a failsafe system in which each site that does the mount stands ready to take over the duties of the CSS if the CSS should go down. Thus "superstability" is achieved by replication of potential CSS responsibilities. The synchronization of reads and writes to that file system is thus assured. Note that "replication" is used in a new and different sense in this NFS context than in the rest of AIX. In actuality, the only thing that is being replicated here is the potentiality of assuming responsibility for synchronization. Keep the following points in mind:

- Only an administered file system can be mounted by several sites i the cluster at once.

- If a multi-site mount of a single file system has been made, that fil system must also be unmounted by multiple sites.

The sole exception to the second rule above is as follows: a System Administrator can place the number 0 in the **/etc/filesystems** stanza for a particular file system. This appears to be a gfs number but is really a request for the system to choose its own number. This is equivalent to a nonadministered mount.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.2.2 Nonadministered Mounts

The purpose of a nonadministered mount is to save the System Administrator the trouble of assigning a unique gfs number to every single one of the ever changing and constantly growing number of file systems available on a complex network. This way, if a user becomes aware of some interesting material on some remote site available on the network, he can simply issue the mount command and use that material. He need not wait for the System Administrator to assign a number. The NFS software automatically picks a number for him. The only restriction is that this mount is considered nonadministered and may not be done on more than one site at a time.

These nonadministered mounts are, in a sense, temporary. They depend entirely upon the one machine that performed the mount and all others make their access to that material through this sole mounting site. When that machines logs off the network, crashes, or unmounts the file system, all others lose access to it.

When no gfs number appears in the file's stanza in the **/etc/filesystems** file, or no such stanza appears in the file, the operating system considers this a nonadministered mount and assigns its own gfs number. If other sites were to mount that same file system themselves, the system would assign new gfs numbers and would no longer be able to synchronize changes to the file system.

What actually happens when a second site tries to perform a nonadministered mount varies according to where on the cluster file tree one tries to mount it. If an attempt is made to mount the file system on the same mount point where it was mounted by the first machine, that attempt will be detected by the system and forbidden. If, however, one tries to mount it elsewhere, the system will be fooled and that attempt will be successful. Nevertheless, this should not be attempted, as it lays one open to the synchronization problems described above. If one needs to have the same material available at two different points on the file tree, the way to do it is with a symbolic link.

Nonadministered mounts may not be replicated throughout the AIX cluster; that is, multiple concurrent mounts of the same file system may not be made. The mounted file system remains available to the rest of the AIX cluster only as long as the single mounting machine remains in service and/or until the mounting machine unmounts it.

Only one machine in the cluster may perform a nonadministered mount of any particular remote file system. That is, the same remote file system should not be mounted by two sites in the cluster concurrently unless a gfs number has been assigned and it has been written into the **/etc/filesystems** file of each site that is to perform the mount.

It is possible to mount the same file system in two separate places on the file tree of the same machine. This is an unwise choice in actual practice, however. Once again, the system has no way to know that these apparently separate file systems are actually one and the same, and no way to synchronize changes made to those files.

There is, however, one way to do so safely. NFS file systems can be mounted read-only. This is a completely separate operation from the granting of file permissions, which can only be done once for each file in the file system. Mounting a file system as read-only affects the whole file system, but only as it is mounted at that particular mountpoint. Thus it would be possible to mount a remote file system on one mountpoint

¦ Copyright IBM Corp. 1985, 1991

as read-only, with the very same file system being mounted on some other mountpoint as full-access. Thus one copy of the file system would be available in one place for the public to read, while the same material is mounted elsewhere and being actively changed by a more select group.

This system has its own inherent problems. The data caching routines would not detect changes being made to the files because the system will treat them as two separate file systems. The public access files would not be kept automatically up to date with the most recent changes. If this is not crucial and group access permissions are too hard to accomplish, a read-only, double-mounted file tree may be acceptable. Nevertheless, group permission arrangements are usually the optimum solution.

It should be pointed out that the AIX implementation of NFS allows a great deal of latitude. Our classic picture of the mounting process is one of machines in an AIX cluster acting as clients and mounting file systems from remote NFS servers. Other arrangements are also possible.

AIX machines can act as NFS servers for remote machines. This is common enough that you may well have occasion to do it.

Two other arrangements are also possible in theory, though unlikely in practice.

Sites in an AIX cluster can act as servers for other machines in the same cluster; that is, an AIX site can mount a file system from another machine within the cluster. It is even possible for a single machine to act as both server and client in the same relationship; that is, a machine can mount one of its own file systems, placing it at some new point in its own local file tree.

Such mountings are extremely rare because there is usually no reason to perform them. They are possible but not advised.

One final use for NFS is to strengthen the bonds between AIX clusters. The AIX Operating System has a 32-site limit, but there may well be more than 32 machines in a single organization. In this case, two or more AIX clusters can still process a single shared file system through a judicious use of NFS mounts.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.3 NFS and LAN Configuration in a TCF Environment

This section describes NFS and LAN configuration for users in a TCF environment.

In an AIX cluster, all machines have access to all file systems stored anywhere in the network, and access to those files is transparent to the user. He or she may or may not know where his files are actually stored and will often give it no thought.

System Administrators, however, do need to consider where the files are stored, which machines are likely to access which files, and approximately how often. The issue is efficiency of access.

Depending on the configuration of the LAN, some machines will be able to access remote files in a single step, while other sites will gain access in a two-step process, through intermediaries. Consider Figure 10-1 below.

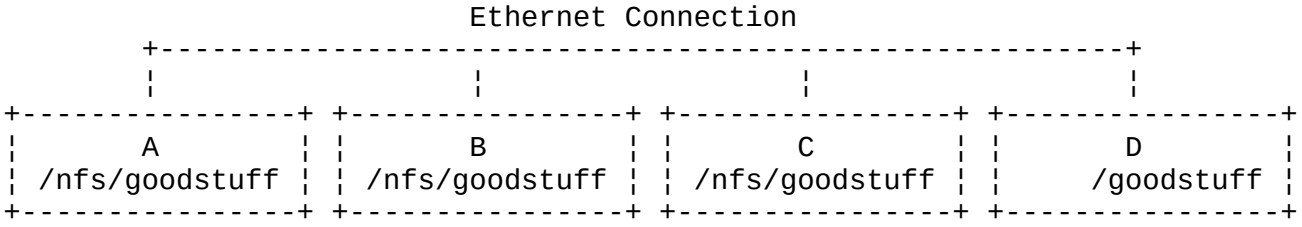


Figure 10-1. Example of an Ethernet Connection

Site D in Figure 10-1 above represents the machine where file system **/goodstuff** physically exists. We are assuming here that D is acting as the NFS server and that site C has mounted D's file system **/goodstuff** onto the directory **/nfs/goodstuff**.

Every site in the network has access to file system **/goodstuff**, but only site C has actually performed a mount of **/goodstuff**. C has chosen to mount the file system in the conventional way, mounting it in its own file tree at **/nfs/goodstuff**.

It is customary to mount a file system on an empty directory which has been created especially for that purpose and given a matching name, but this is not required. The file system can be mounted anywhere on the file tree. The names do not need to match. It can even be mounted on a directory with files in it, but this is not advised. The files in that directory will become inaccessible for the duration of the mount. In Figure 10-1, C has gained a software link to **/goodstuff** and can use its material.

When your AIX system is running TCF, Transparent Distributed Processing is achieved, in part, by the replication of the root directory and its immediate appendages on every machine throughout the cluster. This is what gives the impression of a single file shared file system, all of which appears to be present on each site. We are further assuming that **/nfs/goodstuff** is part of this replicated root system.

Thus, since **/nfs/goodstuff** exists on every site in the cluster, and C has mounted **/goodstuff** on **/nfs/goodstuff**, then **/goodstuff** appears to exist on every site and any machine can access its material. Every machine has automatically gained the same software link to **/goodstuff**.

¦ Copyright IBM Corp. 1985, 1991

Nevertheless, only C is considered to have a direct software pathway. This is because C is the Current Synchronization Site. In all cases, only one site in the cluster has this direct path, even if other sites should perform mounts of the same file system. This simply means that, by convention and for purposes of synchronization, only C is allowed to perform on those files any operations which change any of the statistics of the file (for instance changing the permissions). Any other site that wishes to do so must request C to do it on their behalf, though this is transparent to the user.

Sites A and B gain access to **/goodstuff** files through C. They can perform only a few simple operations on **/goodstuff** files "directly" (that is, without communicating with C). Reads and writes will be done in a single step. Any other operations by A or B on **/goodstuff** files require participation from C. They rely on C to establish and maintain a fully synchronized logical connection with D. Directory lookups, change-inode operations, and synchronization of data caching can only be done with the participation of site C.

Change-inode operations include anything that changes the "statistics" of the file--the information inscribed in the inode. They would include change-owner, change-group, change-mode, and so forth. A more thorough discussion of data caching follows later in this section.

Site C can perform any of these operations in a single step when doing them on its own behalf.

Site C is acting as the CSS (Current Synchronization Site), assuming the task of guaranteeing that each site reads the most recently updated version of the files and that no two sites attempt to write to the file at the same moment.

Let us now assume that D represents a remote data base, and a System Administrator is trying to decide which machine should be used as a front end to the data base. Referring again to Figure 10-1, the answer is simple. It does not matter. Any of the three machines, A, B, or C can be chosen. They all have ethernet pathways to D, and they all have the **/nfs/goodstuff** directory in their replicated root. Thus they all have equal potential to mount the **/goodstuff** file system and act as its CSS.

A second possibility is represented by Figure 10-2. The physical layout is the same, but in this case, **/nfs/goodstuff** is not part of the replicated root. It is a local filesystem on site C. We are assuming that **/nfs/goodstuff** is a directory devoted to goodstuff operations, so the file system **/goodstuff** certainly belongs there logically. Yet only Site C has immediate access to this directory because it is local. All other sites can read that directory only by going through C.

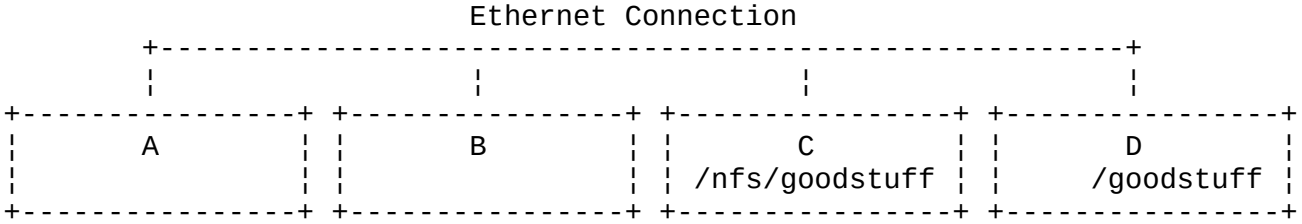


Figure 10-2. Example of an Ethernet Connection

Therefore, in this case, the choice of which machine to use to perform the mount and function as CSS is easy. In terms of efficiency, site C is the

¦ Copyright IBM Corp. 1985, 1991

only practical choice.

But refer now to Figure 10-3. Here is another LAN configuration. D is acting as a Gateway between two separate Ethernet networks. A, B, and C constitute an AIX cluster. Site C of this cluster is also connected to the second network, which includes D and E. This second network may be running AIX or some other operating system.

Assume that, once again, D is a remote NFS server offering a data base and that a System Administrator on the first network is trying to decide which machine to use as its front end.

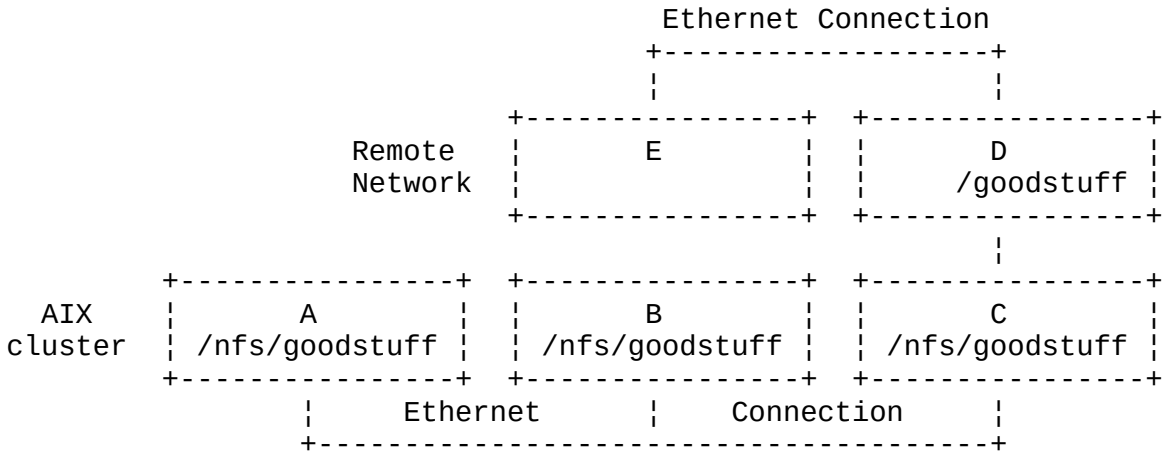


Figure 10-3. Example of an Ethernet Connection

In this case, there is only one answer. Only C has a hardware linkage to D, so only C should perform the mount. After that, A and B can access the **/goodstuff** files, but C should always act as CSS. This is a performance issue only. Either A or B could do the mount. It would merely be inefficient to do so because all requests would have to be routed through C and D anyway.

Refer now to Figure 10-1 again. The remote site D also has the potential to mount file systems from any of the other three machines. D can take its choice of any of the other three to use as NFS server for that operation. The most efficient way of doing so would be to select the machine where the files actually reside. Either of the other two will work but not as efficiently.

If the file system that D wants to mount is on site A and if B is selected as server, then many procedures that D might want to perform will have to be two-step operations. In contrast, if A is selected as server, all file processing actions are one-step operations.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.4 Daemons

Most of NFS is part of the kernel of the AIX Operating System, but there are important segments of NFS, called daemons which are separately running programs. Typically these daemons are started when the machine comes up, and they remain running constantly thereafter. Their function is to stand by at a port or a socket and wait for incoming transmissions. When such a transmission is detected, they usually perform a single system call and then go back to listening at the port.

A socket is a kernel data structure, an addressed area of memory within the kernel. A "port number" is assigned to each socket and messages can be sent to various programs by specifying the appropriate port number. The message is written into the associated memory area (the socket) and the daemon program reads this area periodically for messages. At the coding level, this is what is meant by the figurative phrase "a daemon listening at a port."

In its role as an NFS server, each machine has one port through which all service requests come. An NFS client, on the other hand, has multiple sockets, each with its own port number; each such port is used only to communicate with the single port on an NFS server. NFS service requests are always port-to-port communications.

A daemon is a special form of utility program, a noninteractive system application, as opposed to a user application. The daemon is a highly specialized program in that it is not, itself, part of the kernel; yet nearly all its work is done in and by the kernel.

The NFS daemon, for example, is found on an NFS server, and is responsible for handling remote client requests. The **biod** daemon is found on a client site, and is responsible for read-ahead and write-behind work done in data buffers.

Any machine that has local file systems that may be needed by other machines in the network can be configured at startup as an NFS server. Each server has a daemon, who stands ready to serve the needs of those other sites, which might at any moment request access to the files they have mounted. This NFS daemon stands by a certain port, listening for client requests.

At some point, a process on one of the client machines wants to read a remote file on the server. The process makes all read requests to its own kernel. The kernel is unable to fulfill this particular request because the requested files are on a remote NFS server. The process sends out the request on one of its own ports, roughly in the form of "I would like to read such-and-such block of such-and-such file handle". Then the process stands by its port to await a reply.

On the NFS server site, the NFS daemon receives this request, issues the proper system call(s), reads the data, and packages it appropriately. Then he responds, roughly, "I have read it for you and here it is" and sends it back over the net.

Thus **nfsd**, which is the NFS service daemon, does the work for clients.

The waiting process receives the data via its own port and the incoming information is stored in one or more buffers. The first block is read into a buffer by the process itself. If read ahead has been requested, the next block is read into another buffer by the **biod** daemon (buffered

¦ Copyright IBM Corp. 1985, 1991

input/output daemon).

The **biod** performs this read ahead on behalf of the client process. When the process wants to write data, the **biod** can also perform write behind. A more thorough discussion of buffered I/O follows in the next section.

When a machine is to act as server, the **nfsd** daemons are usually initialized at startup time. Typically two to four of them stand by at the network port, listening for requests. They form a queue, taking turns listening at the port at various intervals, (some more frequently than others, perhaps) but all of them sharing the use of the port. When a request comes in, the appropriate daemon hears it, services the request by issuing his own particular system call, finishes the service and gets back in line for more listening.

When a machine is to act as client, its **biod**s are also initiated at startup, but they sleep inside the kernel until asynchronous I/O is requested. Then they are awakened, one at a time. Each performs its I/O job and then goes back to sleep.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.5 Data Caching in a TCF Environment

A good operating system is constructed using a principle called "program locality," which states that programs have a strong tendency to read their next data from the immediate area of the last read. Since data access from disk is much slower than data access from RAM, the process will be speeded up considerably if the operating system moves a larger chunk of data into memory than the program asks for. There will then be a high probability that the program's next data request will be for something in this "data cache". Most such data requests can then be satisfied from high-speed memory rather than from disk.

Data caching is a built-in feature of the AIX Operating System. When NFS was added, its implementation required the addition of **biods**. One of the responsibilities assumed by the CSS is to monitor this data caching activity.

AIX is a complex, multisite, multiuser, multitasking environment. There are often many, many processes running on many sites concurrently. Since file systems are being shared, and since data is being read into buffers in large aggregates, there is a high probability that one process will read a block of material into a buffer on its own site, and some part of this data will be immediately changed by another process working in its own buffer on some other site. If the first process continues merrily reading its own buffer, it will be reading outdated information.

Part of the function of the CSS is to invalidate any buffers that contain information which has changed and force them to be refreshed with current data. It keeps a complete record of which processes are reading and which are writing. It controls each site's use of data caching.

NFS uses the **biod** daemons to perform what is called read ahead and write behind.

If a process requests that two bytes be read from a certain file, the file system will read a whole block into one of the buffers. It may also perform an asynchronous read of the next block, assuming that the process is likely to need this data next. This is read ahead and, in the case of an NFS file system, it is done by the **biod** daemons.

Writing is handled in like manner. Assume that you have opened a new file on a remote NFS server and you are writing to that file. The process that does the writing channels its output into a series of buffers and the contents of the buffers may be held there for some time. This holds the material available for additions or changes, as often happens with programs like word processing. When the buffers are full, or when the process ends and the file is closed, the file system flushes these buffers out to the server. The **biod** daemons handle this write behind. The process has already written the material to buffer; the daemons come along behind and write it to disk.

The **biod** daemons are only used for NFS. They allow a user process to continue with other work rather than waiting for the NFS server to signal its readiness and then receive the contents of the buffers. These latter actions are handled by the **biod** daemons and constitute what is called write-behind.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.6 Remote Procedure Calls in a TCF Environment

The RPC (Remote Procedure Call) system allows clients to work with remote file systems as if they were local. The client host executes a process which sends service requests to the server host. The request is fulfilled and the results are sent back. The net result is as if the client had executed the process for itself, totally within its own address space.

A number of server daemons cooperate in providing remote RPC service. These are programs or processes which run on behalf of remote NFS clients. They do jobs on the server host which have been requested by some process running on the client host.

Typically, there are a number of programs involved in most of the NFS services: the **portmapper**, the **inetd**, and the server daemons themselves.

These server daemons can be brought into existence in one of two ways:

They can be started explicitly in a startup script

They can be started by another daemon called the **inetd**.

In order for the latter option to take place, the **inetd** must be properly configured when it is started up. This is done by writing the name of the daemon to be started into the **inetd**'s configuration file, **/etc/inetd.conf**.

The interaction between these various daemons is best illustrated by an example.

Suppose someone wants to find out the number of users on a remote site. He runs the program called **rusers**. This **rusers** program contacts the **portmapper** daemon on the remote site to find out which port to use in order to make contact with the **ruser** daemon. The portmapper returns the port number. The **ruser** program on the client site then sends a message to that port number.

If the **ruser** daemon has already been initialized and is listening at its port, it simply carries out the service. If, however, it has not yet been started, it needs to be brought up. This is the job of another daemon.

If the **ruser** daemon is listed in the **inetd** configuration file, then **inetd** is responsible for starting it. The first time such a request is received, **inetd** will dispatch (start up) that particular server daemon. The ruser daemon will then finish the transaction and return the number of users to the client.

This particular interaction usually occurs only once. Once started, the **ruser** daemon usually keeps running. It receives and fills all subsequent requests for its services directly. All subsequent service requests go direct to **ruserd**. A server that is brought up manually, by a startup script, receives all requests directly, including the first.

The sole job of the **inetd** command is to start servers when the first request is received. The **inetd** command does not come into play unless two conditions are present:

1. A service request has been received.

2. The daemon responsible for servicing that request is asleep.

¦ Copyright IBM Corp. 1985, 1991

Whenever a server daemon is brought up, its first action is to register itself with the **portmapper**. It reports to the **portmapper** that it can handle such-and-such program numbers and that it will be standing by at such-and-such port number. When a request is received for one of these program numbers, the **portmapper** daemon passes along the appropriate port number, and the client program sends its request to that port.

Note: The locale of the client environment should be the same as the default locale of the server environment. If the locales do not match, unexpected results may occur. For more information about MBCS on remote file systems, see the *Guide to Multibyte Character Set (MBCS) Support*.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.7 Dispatching

When the service daemon receives a request, it dispatches (executes) the appropriate procedure; a server daemon may have more than one procedure it can perform, but they are relatively simple programs.

Note that there are two separate uses of the word, dispatch, used with respect to daemons. The **inetd** dispatches (starts up) other daemon processes, and each daemon process is said to dispatch (execute) its own internal functions.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.8 The NFS Component

When the NFS component is installed, files located on network servers can be accessed through the **mount** command. Unlike remote copy or transfer functions, users are presented with complete file systems that contain the actual files instead of copies of the files. They can read and write to the remote files as if they were located on their local machine.

Subtopics
6.10.5.8.1 How NFS Works
6.10.5.8.2 Using NFS for Remote File Service

¦ Copyright IBM Corp. 1985, 1991

6.10.5.8.1 How NFS Works

NFS is designed to work across networks without being dependent on machine types, operating systems, or network architectures. It achieves this independence through the use of ***remote procedure calls*** defined through the ***Remote Procedure Call*** protocol built on top of an ***eXternal Data Representation*** protocol.

NFS uses remote procedure calls to communicate between the computer on which the user is logged (the client) and the computer on which the file is located (the server). Using remote procedure calls enables users to work with remote files as if they were local. The remote procedure calls are defined through routines contained in the Remote Procedure Call (RPC) protocol. RPC makes it possible for the client to invoke a caller process that can have a process located on the server execute the procedure call as if the caller process had executed the call in its own address space. See *AIX Operating System Technical Reference* and *AIX Programming Tools and Interfaces* for more detailed information on RPC and the RPC routines.

The eXternal Data Representation (XDR) protocol provides a uniform way of representing data types over a network. The XDR routines are used by RPC to standardize the representation of data in remote communications. This resolves situations, such as different byte ordering, that can occur between communicating machines. XDR converts the parameters and results of each RPC service provided. See *AIX Operating System Technical Reference* for more detailed information on XDR and the XDR routines.

NFS uses authentication mechanisms built into the RPC facility for AIX file protection. ***Authentication*** is a means of verifying the user of an information system or resource. RPC includes a slot for the authentication parameters on every remote procedure call so that the caller can identify itself to the server. See *AIX Operating System Technical Reference* for more detailed information on the authentication routines.

NFS and RPC are not dependent on the data transport used to carry remote procedure call data. In the IBM AIX Network File System environment, NFS and RPC use User Datagram Protocol/Internet Protocol (UDP/IP) and Transmission Control Protocol/Internet Protocol (TCP/IP) as the transport protocols for carrying the message data between communicating programs. UDP/IP is the default protocol used by RPC, but the default value can be changed to TCP/IP in RPC routines when necessary. See *AIX Programming Tools and Interfaces* for information on changing the default values for RPC routines.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.8.2 Using NFS for Remote File Service

The NFS interface is designed to provide ***stateless*** file service. In a ***stateless*** file service environment, the server does not have to keep track of its transactions with clients, completed transactions, or files operated on from one transaction to the next. Because servers do not have to keep track of the state of each file, there is no **open** operation in NFS itself. NFS relies on the client to remember the information for later NFS uses.

An example of a stateless server operation is in the use of the **mount** command. A client can use the **mount** command to build a view of a file system located on its local devices. The client can use the **mount** command with NFS extended capabilities to build a view of a file system located on the network. This is a stateless operation because the command gets only the information needed for the client to establish contact with a server. It does not require the server to keep any information.

An NFS server can also be a client of another server. Servers can only answer requests about their own remote mounts. They cannot serve as messengers between a client and another server.

Users can use AIX commands to read and write to remote files, and to read and set file attributes on remote files. Traditional AIX commands can also be used to read remote directories, and to create and remove remote files.

IBM AIX Network File System does not support access to remote special files (including the device drivers in **/dev**).

¦ Copyright IBM Corp. 1985, 1991

6.10.5.9 The NIS (Network Information Services) Component

Included in the IBM AIX Network File System software is the NIS (Network Information Services), commonly referred to as the NIS. The NIS is a service used to administer system information on networked machines. The NIS is structured as a centralized network look-up facility comprising NIS data bases that provide system information, such as passwords and host names, to Network File System users. For users with different locales, use ASCII characters only for IDs, site names, domains, and so forth.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.10.5.9.1 NIS Domains and Hosts

A group of hosts that share a NIS data base belong to the same domain. The hosts are usually grouped together in the domain for a common reason, such as belonging to members of the same work group at a particular location. Each NIS host is assigned to a domain when the system starts. The default value for the domain is specified in the **/etc/rc.nfs** file. The person who manages your system (or a user with superuser authority) can assign a host to a different domain or change the name of the domain with the **domainname** command. The domain name must be set on all host that intend to use the NIS. See *AIX Operating System Commands Reference* for information on the **domainname** command.

For users with different locales, use ASCII characters only for IDs, site names, domains, and so forth.

The host that maintains the NIS data base for a domain is called the ***NIS master service***. To balance the NIS processing task load as well as provide services in the event the NIS master server is unavailable, additional hosts can be designated as ***NIS slave servers***. The NIS slave servers maintain exact replicas of the master NIS data base. Changes to the NIS data base are made on the NIS master server, and then ***propagated*** (transferred) to the NIS slave servers.

Note: To ensure integrity and reliability of NIS data throughout the domain, the NIS service algorithm requires that changes to the data base be made only on the NIS master server.

An application's access to data served by the NIS is independent of the relative locations of a NIS client and server. A NIS client accesses system information maintained in the NIS data base by making a remote procedure call to a NIS server. The NIS server searches its local data base and returns the requested information to the NIS client.

¦ Copyright IBM Corp. 1985, 1991

6.10.5.9.2 NIS Maps

Data base information is maintained in ***NIS maps***. Each NIS map is created by associating an index ***key*** with a ***value***. For example, the information in the NIS master service's **/etc/hosts** file is used to create a NIS map that uses each host name as a key. The value associated with each key contains the name, aliases, and Internet address of the host. The key and value pairs, also known as ***records***, that are created from the entries in **/etc/hosts** comprise the NIS host name map.

The NIS maps are created using the **makedbm** utility, which converts input into dbm format files. A NIS map consists of two dbm files: **map.key.pag** and **map.key.dir**. For example, the host name map in IBM AIX Network File System consists of files called **hosts.nam.pag** and **hosts.nam.dir**. The file with the **.pag** extension contains the key and value pairs, while the file with the **.dir** extension serves as an index for large **.pag** files. The files are stored in a directory under the **/etc/yp** directory that corresponds to the name of the domain they serve. For example, maps for a domain known as **publications** would be found in the **/etc/yp/publications** directory.

Note: The **/etc/yp/YP_MAP_XLATE** table is used to resolve map naming convention differences among systems. For example, the host name map in IBM AIX Network File System **hosts.nam**, is converted to **hosts.byname** for use on other systems. This table can be appended to include entries for maps specific to your site. See *AIX Operating System Technical Reference* for information on the **/etc/yp/YP_MAP_XLATE** table.

Default NIS Maps: Program routines are automatically redirected from a local ASCII file to the domain-based NIS map for most information maintained by the default NIS maps. Routines that check password or group information can be directed to consult the local ASCII file before consulting the NIS map.

The default NIS maps maintain values in the same format as the ASCII files from which they are derived. For example, each entry in the **/etc/passwd** file has seven fields that are separated by colons. The values in the NIS password name map have the same seven fields in the same order and format (colons separating the fields). When programs are redirected to the password name map, they find the data in the same format as the **/etc/passwd** file.

¦ Copyright IBM Corp. 1985, 1991

6.10.6 Components of IBM AIX Network File System

The system calls, daemon programs, utility functions, and commands installed with each component of the IBM AIX Network File System (NFS and Network Information Services) are discussed in the following section.

Subtopics
6.10.6.1 NFS System Calls, Utilities, and Commands
6.10.6.2 Network Information Services Software Component

¦ Copyright IBM Corp. 1985, 1991

6.10.6.1 NFS System Calls, Utilities, and Commands

The **async_daemon** and **nfssvc** system calls are added to the kernel when NFS is installed. The **async_daemon** system call performs asynchronous block input/output operations for NFS clients. The **nfssvc** system call loops continuously in the server to perform the server operations. See *AIX Operating System Technical Reference* for more information on these system calls.

The **rpc** file that contains the symbolic names for RPC program numbers is added to the **/etc** directory. This file provides users with readable information about the services available at RPC servers. See *AIX Operating System Technical Reference* for more information on this file.

The NFS daemons and utility programs handle the remote file system mounts and remote file operations, as well as other general network services. Some of the daemons are added to the **/etc** directory. The daemons are started from the command line or by a system process, such as the **inetd** daemon. They can also be invoked from a start-up shell script, such as **/etc/rc.nfs**. Once started, they run continuously on the server awaiting service requests. A brief description of each NFS and RPC daemon follows:

Started from the rc.nfs file:

nfsd Network file system daemons that handle client file system requests.

biod Buffered input/output daemons that asynchronously process data contained in buffer caches to facilitate the **read-ahead** and **write-behind** processes in response to NFS client requests. Asynchronous blocking of data into caches increases the access speed of requests for the information contained in the buffers.

yppasswdd Updates passwords in the Network Information Services password data base. See "Starting the rpc.yppasswdd Daemon" in topic 6.10.10.5.4 for detailed information.

lockd Handles remote NFS file locking.

statd Network status monitor responsible for file locking across server crashes.

Started from the rc.tcpip file:

portmap Maps RPC program numbers to port numbers. When a client makes a remote procedure call to a specific program, it contacts the **portmap** process (also known as the **portmapper**) on the NFS server to determine the port where the request should be sent.

Started from the inetd daemon (in the inetd.conf file):

mountd Answers client file system mount requests.

rexd Answers remote requests for execution of programs.

rstatd Returns performance statistics obtained from the kernel.

rusersd Services requests for information on system users.

¦ Copyright IBM Corp. 1985, 1991

sprayd Records status of data packets sent over the network by the **spray** command.

rwalld Services requests to write messages to network users.

See *AIX Operating System Commands Reference* for information specific to each daemon.

The NFS commands can be used to obtain information about remote files and to control remote file system operations made in the network. A brief description of each command follows:

nfsstat Lists statistics about NFS and RPC. Users with superuser authority can use this command to reinitialize the statistics registers.

onrpc Runs a command on a remote computer. The superuser cannot use the **onrpc** command.

rpcgen Compiles the input to remote procedure calls into the C language code for implementation.

rup Lists the host status of remote machines.

rusers Lists users logged in on remote machines.

rwall Writes messages to all users over the network.

showmount Lists clients that have remotely mounted a file system from a specific host.

spray Sends a stream of data packets to a specified host machine to test the system.

You must use ASCII characters for host names.

See *AIX Operating System Commands Reference* for more information on each command.

¦ Copyright IBM Corp. 1985, 1991

6.10.6.2 Network Information Services Software Component

When the NIS is running, client requests for system information are actually broadcast on the network and answered by the NIS. The NIS service looks up the information in the data base located on a NIS server, and returns the information to the client.

The NIS user and administrative commands are added to the appropriate directories when the NIS is installed. A brief description of each command follows:

domainname Lists the name of the current NIS domain system for a NIS host. A user with superuser authority can use this command to change and set the domain.

makedbm Creates the NIS data base maps.

ypbind Establishes a connection that enables a NIS client process to communicate with the **ypserv** process.

ypcat Lists the contents of NIS maps.

ypinit Builds and installs the NIS data base on the NIS servers.

yppasswd Allows users to change NIS passwords from any NIS host.

yppoll Identifies the version of a NIS map on a NIS server.

yppush Forces propagation of a changed NIS map from the NIS master service to each NIS slave server.

ypserv Looks up information in the NIS data bases.

ypset Points the **ypbind** process at a particular server. When using point-to-point links, the **ypset** command must be used (assuming you are not the yp server). This includes the X.25 LPP and CTCs.

ypwhich Identifies which machine is the NIS server of a NIS client.

ypxfr Requests propagation of a NIS map from the NIS master service. (This command is issued from NIS slave servers.)

Note: When using point-to-point links the command **ypset** must be used (assuming you are not the NIS server). This includes the X.25 LPP and CTCs.

See *AIX Operating System Commands Reference* for more detailed information on each NIS command.

The NIS utilities that handle the functions involved with the client requests are located in **libc.a**. A brief description of each NIS utility and its function follows:

Enumerating Map Values

yp_all Returns all values in the map in a single request by transferring the entire map from server to a client.

yp_first Returns the first key-value pair from a map.

¦ Copyright IBM Corp. 1985, 1991

yp_next	Returns the next key-value pair in a map. Attributes can be set to determine how many times **yp_next** is read.

Working With Domains and Maps

yp_get_default_domain	Returns the name of a node's domain.
yp_master	Returns the machine name of the NIS master service for a map.
yp_match	Returns the value associated with a passed key.
yp_order	Returns the order number for a map. An order number is the date (in UNIX seconds) the map was built.

Error Information

yperr_string	Returns a pointer to an error message string that is null-terminated but contains no period or newline character.
ypprot_err	Translates a NIS protocol error code into a readable error code. The error code can be used as input to the **yperr_string** routine.

See *AIX Operating System Technical Reference* for more detailed information on the NIS client utilities.

¦ Copyright IBM Corp. 1985, 1991

6.10.7 Installing the IBM AIX Network File System

IBM AIX Network File System is shipped as a licensed program. The installation procedures conform to the standards and conventions for all AIX licensed programs as described in the *AIX Programming Tools and Interfaces*. Be sure no other system activity is going on when the program is being installed on your system.

Starting a Network File System Installation

1. To start the installation, type the **installp** command at the command line.

Note: To cancel the installation, enter **quit**.

NFS makes it possible to use both System/370 and PS/2 or PS/55 workstations as NFS servers and clients.

Network Information Services makes it possible to consolidate system information onto a centralized data base and use the Network Information Services to administer the information from the data base.

A Note about Reinstalling NFS: If you reinstall NFS on top of itself, the system backs up the following files that you may have altered:

/etc/rc.tcpip

/etc/rc.nfs

/etc/inetd.conf

The **/usr/lpp/nfs** directory serves as the root for these backups.

The NFS install program then reconfigures the real files in exactly the same manner as if it were installing NFS for the first time.

¦ Copyright IBM Corp. 1985, 1991

6.10.8 Configuring NFS on Your System

This section discusses how to configure an NFS network the first time you install it.

Note: Only users with superuser authority can configure NFS.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.10.8.1 Planning Your NFS Implementation

Before you begin, you must plan your NFS implementation. Consider which nodes or workstations should be servers, and which ones should be set up as clients only. NFS servers ***export*** file systems to NFS clients. To ***export*** a file system means to make it available for use by other machines (clients) on the network. Access to exported file systems can be restricted to specified clients. Remember that a workstation can be both a server and a client.

Configuring NFS the First Time

When you have decided on the NFS configuration of your network, do the following on all network nodes or workstations that are to be set up as NFS servers and clients:

1. Edit the **/etc/rc** file to start the **/etc/rc.tcpip** and **/etc/rc.nfs** programs.

2. Edit the **/etc/rc.tcpip** file to start the **portmap** and **inetd** daemon programs.

3. Customize each workstation designated as an NFS server as follows:

    a. Edit the **/etc/inetd.conf** file to start the **mountd** daemon and any other RPC daemons you plan to run for your NFS implementation. See "NFS System Calls, Utilities, and Commands" in topic 6.10.6.1 for the list of RPC daemons available.
    b. Edit the **/etc/rc.nfs** file to start the **nfsd lockd**, **statd**, and **biod** daemons.
    c. Edit the **/etc/exports** file.

4. Customize each workstation designated as an NFS client as follows:

    a. Establish default NFS mounts if appropriate.
    b. Edit the **rc.nfs** file to start the **biod** daemon and to include a **mount** command that automatically mounts remote files when the system starts.
    c. If you plan to run RPC daemons, edit the **inetd.conf** file to start the daemons. See "NFS System Calls, Utilities, and Commands" in topic 6.10.6.1 for the list of RPC daemons available.

5. Shut down the system and reboot to start NFS.

Note: If your network contains a variety of machines from different manufacturers, confirm that the maximum data packet size is consistent among the configured machines. Search the **/etc/net** file for the network device you are using for NFS and adjust the **inetlen** and **r_inetlen** entries.

Subtopics
6.10.8.1.1 Editing /etc/rc on All NFS Servers and Clients
6.10.8.1.2 Editing /etc/rc.tcpip On All NFS Servers and Clients

¦ Copyright IBM Corp. 1985, 1991

6.10.8.1.1 Editing /etc/rc on All NFS Servers and Clients

The **/etc/rc** file contains commands that the system reads and implements when it starts, including commands to start the TCP/IP and NFS programs. The TCP/IP program must be properly configured on all NFS servers and clients. See *AIX TCP/IP User's Guide* for information on TCP/IP.

Starting the TCP/IP Interface Program: Search the **/etc/rc** file for the following entry:

```
sh /etc/rc.tcpip
```

If there is a comment (#) symbol at the beginning of the entry, delete the #. This directs the system to read the file containing startup instructions for TCP/IP.

Starting the NFS Program: There is also an entry in **/etc/rc** for the NFS program. Search the file for the following:

```
sh /etc/rc.nfs
```

If there are comment (#) symbols at the beginning of each line of the entry, delete the # at the beginning of each line. Uncommenting the lines directs the system to read the file containing startup instructions for NFS the next time the system starts.

¦ Copyright IBM Corp. 1985, 1991

6.10.8.1.2 Editing /etc/rc.tcpip On All NFS Servers and Clients

The **/etc/rc.tcpip** file contains entries for the **portmap** and **inetd** daemons, which are required for NFS.

Starting the portmap Daemon: The **portmap** daemon keeps track of which RPC services each RPC server handles and the ports the servers are listening on. Search the **/etc/rc.tcpip** file for the following entry:

```
if [-x /etc/portmap]
then
        /etc/portmap
        sleep 3
fi
```

If comment (#) symbols appear at the beginning of each line in the entry, delete the # from the beginning of each line. Uncommenting the lines causes the **portmap** daemon, sometimes referred to as the **portmapper**, to start the next time **rc.tcpip** is run.

Starting the inetd Daemon: The **inetd** daemon is the internet server process. Search the **/etc/rc.tcpip** file for its entry:

```
if [-x /etc/inetd]
then
        /etc/inetd
fi
```

If comment (#) symbols appear at the beginning of each line in the entry, delete the # from the beginning of each line. Uncommenting the lines causes the **inetd** daemon to start the next time **rc.tcpip** is run.

Note: The **portmap** entry must appear before the **inetd** entry in the **etc/rc.tcpip** file. Make note of this if you must manually enter the lines for any reason.

¦ Copyright IBM Corp. 1985, 1991

6.10.8.2 Customizing an NFS Server

Customizing each workstation designated as an NFS server involves editing the **/etc/inetd.conf**, **/etc/rc.nfs** and **/etc/exports** file.

Subtopics
6.10.8.2.1 Editing /etc/inetd.conf
6.10.8.2.2 Editing /etc/rc.nfs
6.10.8.2.3 Editing the /etc/exports File

¦ Copyright IBM Corp. 1985, 1991

6.10.8.2.1 Editing /etc/inetd.conf

The **mountd** daemon and other daemons required for remote communication are started from the **/etc/inetd.conf** file.

Starting the mountd Daemon: The **mountd** daemon is called by the **mount** command to verify that the file system can be exported by the requesting client. In order for the remote mount to succeed, **mountd** must be active on the server. Search **/etc/inetd.conf** for the following **mountd** entry:

```
rpc dgram udp wait root /etc/rpc.mountd 100005 1
```

If a comment (#) symbol appears at the beginning of this line, delete the #. Uncommenting the line causes the **mountd** daemon to start the next time **/etc/inetd** is run.

For more information about the **mountd** daemon, see **rpc.mountd** in *AIX Operating System Commands Reference*.

Note: If the **inetd** daemon is running on your network and it is possible that **mountd** is already active, you can enter the following at the command line to see if **mountd** is active:

```
rpcinfo -u localhost 100005
```

Replace the **localhost** parameter with the host name of the server you are customizing. If you receive a message that the program is **ready and waiting**, **mountd** is already uncommented.

Starting Other RPC Daemons: In addition, search the file for any other RPC daemons you plan to activate. RPC daemons are identified by lines that look similar to the **mountd** entry and contain the **rpc** field. If comment (#) symbols appear before the entries you plan to activate, remove the # from the beginning of each entry. See "NFS System Calls, Utilities, and Commands" in topic 6.10.6.1 for information on these daemons.

¦ Copyright IBM Corp. 1985, 1991

6.10.8.2.2 Editing /etc/rc.nfs

The network file system daemon (**nfsd**) and block I/O daemon (**biod**) programs are started from the **/etc/rc.nfs** file.

Starting the nfsd Processes: The network file system daemon (**nfsd**) processes carry remote procedure call requests to the kernel for processing. Search the **/etc/rc.nfs** file for the following lines:

```
if [-y /etc/nfsd] && [-f /etc/exports]
then
        /etc/nfsd 3
        echo 'nfsd'
fi
```

The number in the third line (**3** in the example) determines the number of **nfsd** processes that are active. The load on the server determines the number of processes to assign. For an average load, three **nfsd** processes should be available to execute remote procedure calls.

If these lines do not appear, add them to the file after the following entry:

```
echo Starting RPC/NFS daemons...
```

Starting the biod Processes: The block I/O daemon (**biod**) processes handle data transmission between NFS servers and clients. Search the **/etc/rc.nfs** file for the following lines:

```
if [-x /etc/biod ]
then
/etc/biod 3
        echo 'biod'
fi
```

If these lines do not appear, add them to the file following the **nfsd** entry.

Note: Like the **nfsd** entry, the number in the third line determines the number of **biod** processes that are started. The load on the server determines the number of processes to assign. For an average load, three **biod** processes should be available to handle NFS server and client data transmissions.

¦ Copyright IBM Corp. 1985, 1991

6.10.8.2.3 Editing the /etc/exports File

The **/etc/exports** file specifies the file systems minidisks that are available for mounting remotely. A server can only export its own file systems. A server cannot act as a messenger between clients and the file systems of other servers.

File systems can be made available to all clients, or client access to each file system can be controlled by listing the host names or authorized network groups (identified in the **/etc/netgroup** file) that are allowed to remotely mount each file system in the **/etc/exports** file. The **/etc/netgroup** file consists of lists that define network group specific patterns. See *AIX Operating System Technical Reference* for more information on the **netgroup** file.

Using a text editor, create **/etc/exports** and list by full path name each file system that will be exported. Align each file system name at the left margin. The path names must be the mount point of a local or TCF-accessible file system.

For file systems that can be mounted by all clients in the network, do not type any names following the file systems. If you are controlling client access, type the host names (or the **netgroup** names) following the name of the file system leaving white space between each name. If you need more than one line to list the names, start each new line with white space.

Note: You can add comments anywhere within the file by inserting a comment symbol (#) to signal the start of your comment text. The comment text extends to the end of the line on which the symbol appears.

The following shows examples of entries that can be created for an **/etc/exports** file:

```
/u                              # export to the world
/tmp        mach1 mach2 mach3   # export to only these machines
/usr        clients             # export to netgroup called clients
```

In this example, the first entry specifies that all clients can mount the **/u** directory. The second entry specifies that only the machines named **mach1**, **mach2**, and **mach3** can mount the **/tmp** directory. The third entry specifies that only the machines found in the **netgroup** called **clients** can mount the **/usr** file system.

¦ Copyright IBM Corp. 1985, 1991

6.10.8.3 Customizing an NFS Client

Customizing each workstation designated as an NFS client involves editing the **/etc/rc.nfs** file to start the **biod** processes, and establishing default NFS mounts. In addition, you can edit the **/etc/inetd.conf** file to start any RPC daemons that you plan to run on your NFS client workstation.

Subtopics
6.10.8.3.1 Establishing the Default NFS Mounts
6.10.8.3.2 Editing /etc/rc.nfs
6.10.8.3.3 Editing /etc/inetd.conf

¦ Copyright IBM Corp. 1985, 1991

6.10.8.3.1 Establishing the Default NFS Mounts

Default NFS mounts refer to the remote file systems that are mounted automatically when the system starts. To establish the default NFS environment for clients, list the remote file systems that should be mounted automatically when the system starts in the **/etc/filesystems** file on each client. After you have set up the default NFS environment, the mounts are made automatically when the system starts. The specified file systems do not have to be mounted from the command line each time the system reboots. However, other file systems can be mounted remotely from the command line if the appropriate conditions exist for the client. See "Mounting Files Remotely From the Command Line" in topic 6.10.9.3 for detailed information.

Editing /etc/filesystems: The remote file systems that NFS clients attempt to automatically mount when the system starts are identified in the **/etc/filesystems** file. The **/etc/filesystems** file is organized in stanzas. Stanzas are lines of attributes grouped together by common functions. In the case of NFS mounts, the first line of each stanza should be the mount point of the file system being mounted. The mount point is followed by a colon (**:**). The attributes associated with the mount are listed on subsequent lines. See the file formats section in *AIX Operating System Technical Reference* for more detailed information on the **filesystems** file.

Using a text editor, add stanzas to **/etc/filesystems** to define each NFS mount. The following attributes are required for the stanzas you create that pertain to the NFS mounts:

dev = "host:NFSdir"
For NFS mounts, **host** specifies the host machine on which the remote file system resides. **NFSdir** specifies the path name of the remote file system being mounted. Use ASCII characters for **NFSdir** to ensure successful communication across different locales.

vol = "mount_point"
The mount_point specifies the local directory on which the remote file system will be mounted. This should be the same as the first line of the stanza.

mount = false
NFS mounts must use the false mount attribute.

check = false
NFS mounts must use the false check attribute.

site = sitename
When TCF is installed every stanza must contain this attribute.

In addition to these required attributes, you can use the following in stanzas for NFS mounts:

type = type_name
Defines the file system being mounted as part of the **type_name** mount group. This parameter is used in conjunction with **mount-t**, which mounts groups of specified file systems at the same time.

ftype = nfs
NFS mounts must use the nfs ftype attribute.

¦ Copyright IBM Corp. 1985, 1991

gfs = gfs_number
NFS administered mounts must contain a unique non-zero **gfs_number** specified for this attribute.

options = options
The options specified below are a list of comma-separated words. For additional options see the **mount** command in *AIX Operating System Commands Reference (Volume II)*.

The defaults of the options specific to NFS file systems are:

fg,retry=10000,timeo=7,retrans=3,port=NFS_PORT,hard

(with defaults for **rsize** and **wsize** set by the kernel).

bg If the first mount attempt fails, try the mount again in the background.

fg Try the mount again in the foreground.

retry=n Set the number of times to try the mount.

rsize=n Set the **read** buffer size to the number of bytes specified by **n**.

wsize=n Set the **write** buffer size to the number of bytes specified by **n**.

timeo=n Set the NFS timeout to the tenths of a second specified by **n**. Use this option to avoid situations that can occur in networks where the server load can cause inadequate response time.

retrans=n Set the number of NFS retransmissions to the number specified by **n**.

port=n Set the server port to the number specified by **n**.

soft Return an error if the server does not respond.

hard Continue to try the request until the server responds.

intr Allow keyboard interrupts on hard mounts.

ro Read-only.

The **bg** and **fg** options cause the **mount** command to run in the background (**bg**) or foreground (**fg**) if the server's **mountd** process does not respond. The system attempts to transmit each request the number of times specified in the **retry** option before it gives up. Once the file system is mounted, each NFS mount request made in the kernel waits for a response during the time specified in the **timeout** option. If no response arrives, the time-out is multiplied by two, and the request is transmitted again. When the number of retransmitted attempts for a mount request that specifies the **soft** option have been sent with no reply, an error is returned. When the number of retransmitted attempts for a mount request that specifies the **hard** option have been sent with no reply, the system displays a message and retries the request.

Note: If a **soft** mount is attempted as read-write, use the **hard** option

¦ Copyright IBM Corp. 1985, 1991

along with the **rw** (read-write) option to avoid error conditions that can conflict with applications.

When you specify a **hard** mount, it is possible that the process can hang while waiting for a response. To be able to interrupt the process and end it from the keyboard, use the **intr** option in the mount options. The number of bytes in a **read** or **write** request can be set with the **rsize** and **wsize** options. If you do not set these options, the kernel automatically sets them.

The system uses the following as defaults:

```
fg
retry=10000
timeo=7
retrans=3
port=NFS_PORT
hard
```

This means that the system runs the NFS mount request in the foreground. Once the file system is mounted, each NFS mount request made in the kernel waits seven tenths (0.7) of a second for a response. If no response arrives, the time-out is multiplied by 2 and the request is retransmitted. When the request has been retransmitted three times with no reply, the file system prints a message and retries the request. See *AIX Operating System Commands Reference* for more detailed information on the **mount** command.

The following is an example of an NFS stanza that could appear in **/etc/filesystems**. In this example, you are setting up the NFS client known as **mach1** and have already established the mount points for the nodes in your network. The following stanza can be created for an automatic remote mount:

```
  /u/jdoe:
          dev = "mach2:/u/jdoe"
          vol = "/u/jdoe"
          mount = false,readonly
          check = false
          options = soft
          type = nfs_mount
          gfs = 100
          ftype = nfs
          site = mach1
```

This stanza directs the system to mount the remote directory **/u/jdoe** over the local mount point of the same name. The filesystem is mounted as read-only, and since it is mounted as **soft**, an error is returned in the event the server does not respond. By specifying the type as nfs_mount, the system attempts to mount **/u/jdoe** (along with any other file systems that are specified in the type = nfs_mount group) when the command **/etc/mount/ -t nfs_mount** is issued. If rc.nfs file issues this command then this file system will be mounted automatically when the system starts; see below.

The kernel will use gfs 100 for this mount. As this is an administered mount in a TCF environment, other TCF sites may include this stanza in their **/etc/filesystems** file with mach1 changed for each site. The system administrator should assign unique gfs numbers to administered NFS mount stanzas. No other filesystem in the cluster should refer to this gfs

¦ Copyright IBM Corp. 1985, 1991

number. The **/etc/fsmap** should be examined, since it describes all filesystems in the cluster. This stanza may be added to the **/etc/filesystems** file of any mach1, then the following command should be invoked:

```
syncfsmap mach1
```

This will update the **/etc/fsmap** file and make a record of the use of gfs 100.

¦ Copyright IBM Corp. 1985, 1991

6.10.8.3.2 Editing /etc/rc.nfs

NFS file system mounts can be made automatically when the system starts if **/etc/rc.nfs** contains a mount command for remote file systems.

Adding a mount Command Sequence for Automatic Remote Mounts: To mount NFS file systems automatically whenever the system is restarted, edit the **/etc/rc.nfs** file to include a mount command for remote file systems.

Note: File systems that are mounted automatically must be defined in the **/etc/filesystems** file discussed in "Editing /etc/filesystems" in topic 6.10.8.3.1.

Search the **/etc/rc.nfs** file for the following line:

```
/etc/mount -v -t nfs_mount
```

These lines are commented out by default. Uncomment one of these lines by removing the comment (#) from the beginning of the entry. Replace **nfs_mount** with the **type_name** you define in **/etc/filesystems**. When the **/etc/rc.nfs** file is executed, the system attempts to mount every file system that contains the stanza attribute **type = type_name**.

Establishing the Local Mount Points: Confirm that the mount points for NFS mounts you defined in **/etc/filesystems** exist. If they do not exist, you need to create them. Using the **mkdir** (make directory) command, create the mount points in the following form:

```
mkdir dirname
```

¦ Copyright IBM Corp. 1985, 1991

6.10.8.3.3 Editing /etc/inetd.conf

If you plan to use any of the RPC daemon programs to assist in remote communications between the client and other network machines, you must edit the **/etc/inetd.conf** file to activate the appropriate daemons. The **/etc/inetd.conf** file is shipped with the RPC daemon entries in place, but some of the RPC daemons may be commented out by default. In order to activate the RPC daemons you plan to use, search the file for the existing entries. The RPC daemon entries can be identified by a field listed as **rpc**. Remove the comment (#) symbols that appear before the entries to activate the daemons. See "NFS System Calls, Utilities, and Commands" in topic 6.10.6.1 for the list of daemons available and *AIX Operating System Commands Reference* for information on each daemon.

If the **inetd** process is already running on the workstation and you have made changes to the **/etc/inetd.conf** file, send a SIGHUP (signal hangup) to the **inetd** process. SIGHUP causes the **inetd** process to reread the **inetd.conf** file so the changes you have made can be implemented. To issue SIGHUP, type the following at the command line:

```
kill -1 pid
```

The **pid** is the process ID number for the current **inetd** process that is running.

¦ Copyright IBM Corp. 1985, 1991

6.10.8.4 Shutting Down and Rebooting Each System

After you have finished customizing the NFS server and client workstations, shut down and reboot each system by entering the following at the command line:

```
shutdown -r
```

When the system comes back up, the NFS environment you have configured is available.

Note: If the Network Information Services network service is already configured in your network, the order in which you reboot the machines is important. It is recommended that you reboot the NIS master server first, followed by the NIS slave servers and the NIS clients. This is explained in greater detail in "Configuring the Network Information Services on Your System" in topic 6.10.10.

¦ Copyright IBM Corp. 1985, 1991

6.10.9 Maintaining NFS

Once you have configured NFS on your system, you can make changes to the configuration without going through the entire NFS installation procedure. This section discusses maintenance tasks, such as adding network daemons, mounting individual file systems from the command line.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.10.9.1 Changing the Number of Network Daemons

You can change the number of active **nfsd** processes on NFS servers, and the active number of **biod** processes on both NFS servers and clients, at the command line. Use the following format:

```
/etc/daemon #
```

Replace the **daemon** parameter with **nfsd** or **biod**, and the # with the number of processes you want to have active. When you press **Enter**, the new number of daemon processes are available. You do not have to reboot the system to activate the daemons. For example, to increase the number of **biod** daemons currently active from two to five, enter the following:

```
/etc/biod 5
```

When you press **Enter**, there are five **biod** daemons available.

¦ Copyright IBM Corp. 1985, 1991

6.10.9.2 Changing the inetd.conf file

If the **inetd** process is already running on the workstation and you have made changes to the **/etc/inetd.conf** file, send a SIGHUP (signal hangup) to the **inetd** process. SIGHUP causes the **inetd** process to reread the **inetd.conf** file so the changes you have made can be implemented. To issue SIGHUP, type the following at the command line:

```
kill -1 pid
```

The **pid** is the process ID number for the current **inetd** process that is running.

¦ Copyright IBM Corp. 1985, 1991

6.10.9.3 Mounting Files Remotely From the Command Line

A remote file system that is not mounted automatically can be mounted at an NFS client from the command line using the **mount** command. The file system containing the directory to be mounted must be listed in the **/etc/exports** file on the server from which it is being mounted, and the local mount point must exist. In addition, the user must be authorized to use the **mount** command. (See *AIX Operating System Commands Reference* for details on using the mount command.)

Mounting a Specific NFS File System Defined in /etc/filesystems: To mount a specific NFS file system defined in **/etc/filesystems**, enter the following:

```
/etc/mount dirname
```

The **dirname** parameter is the mount point (full path name) of the file system being mounted. For example, a file system called **/reports/week1** can be mounted from the command line if the stanza for the file system exists in the **/etc/filesystems** file, and the file is listed in the server's **/etc/exports** file with no restrictions. To mount the file system, enter the following:

```
/etc/mount /reports/week1
```

Mounting Selected NFS File Systems Defined in /etc/filesystems: If there are groups of multiple remote file systems that you would like to access at one time, you can define each of the files in a group with the same **type_name** in their stanza in **/etc/filesystems**. To mount selected NFS file systems as a group, enter the following:

```
/etc/mount -t type_name
```

The **type_name** parameter refers to the unique name you have defined to represent a group of file systems.

For example, the file systems maintained by a special department, such as the engineering department, could be defined with the **type** attribute in the appropriate **/etc/filesystems** stanzas as follows:

```
type = engineering
```

Then, using the following command, you can instruct the system to mount all the file systems with the **engineering** type attribute:

```
/etc/mount -t engineering
```

For file systems that belong to multiple groups and require mounting on a daily basis, you can add **/etc/mount -t** entries for those mount groups to the end of **/etc/rc.nfs**. This causes the system to attempt to mount the file systems associated with the groups you had defined whenever the system is restarted.

Note: It is not necessary that the directory the user mounts be exactly the same as the directory exported by the server. The user must mount either the exported directory, or a sub-directory of it (provided it resides on the same file system or minidisk).

Mounting NFS File Systems Not Defined in /etc/filesystems: To mount a

¦ Copyright IBM Corp. 1985, 1991

remote NFS file system that is not defined in /etc/filesystems, enter the following at the command line:

```
/etc/mount [-vfrs] [-0 options] hostname:directory mnt_point
```

For users with different locales, use ASCII characters for the host names to ensure that the files can be accessed successfully.

The **hostname** parameter identifies which server the file system is being mounted from. The **options** parameter specifies the options available to NFS mounts, such as read-only (**ro**), read-write (**rw**), hard, or soft. The **dirname1** parameter identifies the name of the remote file system being mounted, while **dirname2** identifies the name of the local directory **dirname1** is being mounted on (the mount point).

For example, the **/reports/week1** file system exists on the server called **mach1** and is listed in **/etc/exports** with no restrictions. You want to mount the file system on the client **mach2** as read-only and as a soft mount. To mount the file system on your local directory **/summary/reports**, enter the following:

```
/etc/mount -o ro,soft mach1:/reports/week1 /summary/reports
```

¦ Copyright IBM Corp. 1985, 1991

6.10.9.4 Superuser Privileges on Mounted File Systems

To prevent unauthorized access to NFS servers, NFS does not allow a user on a client to exercise superuser privileges on files in a mounted file system. The root user ID on the client (**0**) is mapped to the kernel variable **nobody** (-2, or unsigned 65534) when performing file operations in a mounted file system.

Subtopics
6.10.9.4.1 Restrictions on Superuser from a Client

¦ Copyright IBM Corp. 1985, 1991

6.10.9.4.1 Restrictions on Superuser from a Client

The following examples demonstrate restricted privileges for the superuser on an NFS mounted file system. First, as a non-privileged user from a client, create two files in a mounted file system on which the user has write permission. Modify the permissions on each file as shown:

```
cd mounted_filesystem
touch demo1
touch demo2
chmod 700 demo1
chmod 777 demo2
```

Next, become the superuser on the client and attempt to access the files you created as an ordinary user by entering the following at the command line:

```
touch demo1
```

Although you are logged in as the superuser, you are not permitted to modify this file. File access is prohibited because the user ID **0** has been mapped to **65534** for file operations on the mounted file system. No one but root on the host exporting the file system and the file's owner can modify the file.

In the case of **demo2**, the superuser on the client can access this file because the **write** permission for **others** is set.

Programs that run as root (**setuid root** programs) have similar restrictions. A user running a **setuid root** program from a client is not allowed to do privileged operations on files in NFS mounted file systems. Also note that changing ownership of a file in a mounted file system to **root** means that it will be owned by root on the server, not root on the client.

¦ Copyright IBM Corp. 1985, 1991

6.10.10 Configuring the Network Information Services on Your System

This section discusses how to configure the NIS for the first time in your network.

Note: Only users with superuser authority can configure and modify Network Information Services.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.10.10.1 Planning Your NIS Implementation

Before you begin, you must plan your Network Information Services implementation. Consider which nodes or hosts are going to use NIS. When you have determined the hosts that will be using NIS, assign them to NIS domains. A NIS domain refers collectively to a group of a hosts. In many cases, a single domain is sufficient for an NFS network.

In each NIS domain, system and user account information is consolidated in a set of data bases, also called ***NIS maps***. It is recommended that you create and maintain all NIS maps for one NIS domain on a single host. The host on which you maintain a NIS map is called the NIS master server. One or more hosts should be designated to maintain exact replicas of the NIS maps on the NIS master server. These hosts are called NIS slave servers. Use at least one NIS slave server per domain to help balance the NIS processing task load, and to provide NIS services in the event the NIS master server is not available. The remaining hosts should be designated as NIS clients. NIS clients do not maintain any NIS maps. They query the NIS servers for system and user account information. The NIS clients do not have to make a distinction between querying the NIS master server or a NIS slave server.

+--- **Configuring Network Information Services** ----------+
¦ ¦
¦ When you have decided on the NIS configuration of your network, do the ¦
¦ following: ¦
¦ ¦
¦ 1. On all NIS hosts, (the NIS master server, NIS slave servers, and ¦
¦ NIS clients), add the **/etc/yp** directory to the **PATH** variable. ¦
¦ ¦
¦ 2. On all NIS hosts (the NIS master server, NIS slave servers, and ¦
¦ NIS clients), edit the **/etc/rc.nfs** file as follows: ¦
¦ ¦
¦ a. Change the default name of the domain to the one you have ¦
¦ selected. ¦
¦ b. Start the **ypbind** process. ¦
¦ ¦
¦ 3. On each NIS host, set the domain with the **domainname** command. ¦
¦ ¦
¦ 4. Customize the NIS master server as follows: ¦
¦ ¦
¦ a. Edit the **/etc/rc.nfs** file to start the **ypserv** process. ¦
¦ b. Edit the input files that are used to create NIS maps. ¦
¦ c. Create the NIS maps with the **ypinit -m** command. ¦
¦ d. If the users in your NIS network can change their own ¦
¦ passwords, edit the **/etc/rc.nfs** file to start the **yppasswdd** ¦
¦ daemon. ¦
¦ ¦
¦ 5. Customize each NIS slave server as follows: ¦
¦ ¦
¦ a. Edit the **/etc/rc.nfs** file to start the **ypserv** process. ¦
¦ b. Edit selected system files to direct requests to the NIS ¦
¦ service. ¦
¦ c. Replicate the NIS maps from the NIS master server with the ¦
¦ **ypinit -s** command. ¦
¦ ¦
¦ 6. Customize each NIS client by editing selected system files to ¦
¦ direct requests to the NIS service. ¦
¦ ¦
+---+

¦ Copyright IBM Corp. 1985, 1991

6.10.10.2 Modifying the PATH Variable

The **/etc/yp** directory must be added to the **PATH** variable for system users. The **PATH** variable contains a list of the directory names that identify to the startup shell and other commands the locations of the executable files (programs or commands) for the system. Using the method described in "Tailoring the User Environment" in topic 1.2.6, find the standard **PATH** variable for users and modify it to contain the **/etc/yp** directory.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.3 Editing /etc/rc.nfs on All NIS Hosts

The **/etc/rc.nfs** file contains NFS and NIS commands that the system implements when it starts. To customize your NIS environment, you must modify the default domain name and the **ypbind** entries that already exist in **/etc/rc.nfs**.

Changing the Default Name of the Domain: Search the file for the following entry:

```
/bin/domainname domain
```

The **domain** parameter is often specified as **ibm** by default in this file. Replace **domain** (or **ibm**) with the name of the NIS domain in which the NIS host resides. The next time **/etc/rc.nfs** runs, the domain is set to the name you supply.

Starting ypbind: The **ypbind** process searches for a NIS host that has NIS maps to match a NIS client's request. Search the file for the following entry:

```
if [ -f /etc/ypbind ] ; then
    /etc/ypbind ; echo ' ypbind\c'   >/dev/console
fi
```

In most cases, these lines are commented out by default. If the lines are preceded by a comment (#) symbol, remove the # that precedes each line. The next time **/etc/rc.nfs** runs, the **ypbind** process starts.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.4 Setting the NIS Domain

The NIS domain must be set on each NIS host before the NIS maps can be created and distributed. Set the domain by entering the following at the command line:

```
domainname domain
```

Replace the **domain** parameter with the same one you used in the **/etc/rc.nfs** file.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.5 Customizing the NIS Master Server

Customizing the NIS master server involves editing the input files used for NIS maps, creating the NIS maps to generate a master NIS data base, and starting the NIS processes that consult the maps upon NIS client request. In addition, if the users in your NIS network can change their own passwords, you must also edit the **/etc/rc.nfs** file to start the **yppasswdd** daemon.

Subtopics
6.10.10.5.1 Editing the NIS Map Input Files
6.10.10.5.2 Creating the NIS Maps on the NIS Master Server
6.10.10.5.3 Starting the ypserv and ypbind Processes
6.10.10.5.4 Starting the rpc.yppasswdd Daemon

¦ Copyright IBM Corp. 1985, 1991

6.10.10.5.1 Editing the NIS Map Input Files

The NIS maps are usually constructed from text files that contain information in a standardized format. The AIX files used to make the default NIS maps include the following:

/etc/passwd (or a specially designated NIS password account file)
/etc/group (or a specially designated NIS group account file)
/etc/hosts
/etc/rpc
/etc/protocols
/etc/services.

In addition, default NIS maps are created from the following files if they are available on the workstation:

/etc/ethers
/etc/networks
/usr/lib/aliases.

It is important to create and maintain all information that may be required by any host in the NIS domain on the NIS master server. For example, if there are any special RPC program entries that are required by other hosts in the domain, the entries must be added to the **/etc/rpc** file on the NIS master server, even if the NIS master server does not use those programs.

After the NIS maps are installed on the NIS servers, system processes that ordinarily access these text files are redirected to the appropriate NIS map for the information. The exception to this involves the **/etc/passwd**, **/etc/group**, and **/etc/hosts** files.

The **/etc/hosts** file is consulted by the system startup processes before the NIS is activated, but subsequent queries for host information are redirected to a NIS map. In order to provide selected user and group information to the entire domain while maintaining certain information for local use only, system processes that require user and group information first consult the **/etc/passwd** and **/etc/group** files on the local machine. If the processes encounter a special NIS escape sequence in the file before their query has been answered, they are redirected to the NIS map. You can install user and group information in your NIS environment in one of the following manners:

Use **/etc/passwd** and **/etc/group** on the NIS master as the NIS map input files.
Use separate password and group files as the NIS map input files
Do not use NIS for password and group information

Determine which situation is appropriate for your NIS network environment by considering your administrative overhead.

Using /etc/passwd and /etc/group as Map Input: By default, NIS uses the **/etc/passwd** and **/etc/group** files from the NIS master server as input files for the NIS maps that contain password and group information.

All users and groups on the NIS master server are included automatically in the NIS maps. In a network that contains workstations from a variety of manufacturers, the user and group IDs supplied for a workstation may conflict with those on the NIS master server. This can cause file ownership problems if an NFS server exports a directory to an NFS client

¦ Copyright IBM Corp. 1985, 1991

whose password file contains user IDs that match those in the NIS map.

Note: In order for users listed in the NIS password files to change their passwords, the **yppasswdd** must be running on the NIS master server, and they must use the **yppasswd** command.

Warning: Do not put the NIS escape entry in the files you use as the input password and group files for NIS maps. This causes security problems that may affect all workstations on the network.

Using Separate Password and Group Files as NIS Map Input: To permit exclusion of selected entries from the NIS maps, you can create new files, such as **/etc/yp/passwd** and **/etc/yp/group**, that are for NIS users and groups only. Place entries designated for local use only in **/etc/passwd** and **/etc/group**, and place entries designated for NIS use in the new files. To configure the NIS master server in this way, do the following:

1. In **/etc/passwd**, verify that existing entries are local entries only and add the NIS escape sequence (**+::0:0::**) as the last line in the file.

2. In **/etc/group**, verify that existing entries are local entries only and add the NIS escape sequence (**+:**) as the last line in the file.

3. Create new files for the NIS password and group entries. These files should be place in the directory other than **/etc** (for example in **/etc/yp**), should be named **passwd** and group, respectively, and have the same format as **/etc/passwd** and **/etc/group**.

4. Modify the **DIR** variables in the **/etc/rc.nfs** file, so that it names the directory where you place the NIS password and group files. You will be replacing the line **DIR=/etc** with the line *DIR=yp_dir* where *yp_dir* is the directory you chose above in step 3.

Note: If you choose this method of creating separate password and group files for NIS users and groups, you cannot use the adduser or users command to add NIS users or groups. You will instead be required to edit these files using a text editor.

Not Using NIS for Password and Group Information: In a typical NFS environment, home directories are exported over the network so that users can log in from any workstation and access their own working environment. To ensure proper permissions for remotely mounting file systems and to allow access to home directories throughout the network, NFS requires global user ID (UID) and group ID (GID) assignments.

If you choose not to use NIS to manage UID/GID assignments in an NFS environment, you must maintain duplicate password and group accounts for users who need to access multiple NFS hosts. For example, a user on a workstation whose UID is **205** and GID is **35** must have the same UID and GID (**205** and **35**) in **/etc/passwd** on any other work station the user accesses.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.5.2 Creating the NIS Maps on the NIS Master Server

After the text files on the NIS master server have been edited, you can build the default NIS maps. Change to the **/etc/yp** directory and begin the **ypinit** process on the NIS master server by issuing the following at the command line:

```
cd /etc/yp
ypinit -m
```

The system displays a message about quitting the **ypinit** process if it encounters non-fatal errors along with the default answer **NO**. Press **Enter**.

Note: If the system reports errors during the **ypinit** process, run the **ypinit -m** command again after you have resolved the problems.

Next, the system prompts you for the list of hosts you have designated as NIS servers (in addition to the NIS master server). Enter the name of each host you plan to use as a NIS slave server. When you are finished with the list, press **CTRL-D** to continue the **ypinit** process.

As the **ypinit** process continues, it creates a subdirectory in **/etc/yp** that corresponds to the name of the NIS domain you have set. The **ypinit** process then uses the instructions in **/etc/yp/Makefile** on the NIS master server to create the default NIS maps, which are placed in the new subdirectory identified by the NIS domain name. Each NIS map consists of two files called **map.key.pag** and **map.key.dir**. The **map** parameter is the name of the map, such as **passwd** or **group**. The **key** parameter specifies the index criterion, such as **nam** for indexing entries by name or **uid** for indexing entries by user ID. The **.pag** and **.dir** filename extensions are used by the **makedbm** procedure, which is explained later in this section.

Note: To resolve naming convention differences in the **map** and **key** parameters, the NIS processes consult the **/etc/yp/YP_MAP_X_LATE** file for translated names that AIX allows. You can add entries that are specific to your site to this file. See *AIX Operating System Technical Reference* for more information.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.5.3 Starting the ypserv and ypbind Processes

When the **ypinit** process finishes building the NIS maps, start the NIS daemons on the NIS master server.

The **ypserv** process runs as a background daemon on both the NIS master server and NIS slave servers to answer NIS client requests for information. To start **ypserv**, enter the following at the command line:

```
/etc/ypserv
```

If the system can locate the NIS map subdirectory created by the **ypinit** process (the /etc/yp/**domain** directory created in the previous section), **ypserv** starts automatically from the **/etc/rc.nfs** file the next time the system is booted.

The **ypbind** process runs as a background daemon on all NIS hosts to find the NIS server with a NIS map that has the information requested by a client. To start the **ypbind** utility, enter the following at the command line:

```
/etc/ypbind
```

If the entry for **ypbind** is uncommented in the **/etc/rc.nfs** file, **ypbind** starts automatically the next time the system is booted.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.5.4 Starting the rpc.yppasswdd Daemon

If you use the **/etc/passwd** file as the input file or use a separate file (**/etc/yp/passwd**) for your NIS password map, it is important to start the **yppasswdd** daemon on the NIS master server. Without the **rpc.yppasswdd** daemon, users may not be able to change their passwords. If the **rpc.yppasswdd** daemon is running on the NIS master server, users can use the **rpc.yppasswd** command at any NIS host in the domain to change their NIS password. An entry for the **rpc.yppasswdd** daemon is included in the **/etc/rc.nfs** file, but is usually commented out by default.

To activate the **yppasswdd** daemon on the NIS master server, search the **/etc/rc.nfs** file for the following:

```
PASSWD=/etc/passwd
if [ -x /etc/rpc.yppasswdd -a -f $PASSWD ] ; then
    /etc/rpc.yppasswdd $PASSWD -m passwd PASSWD=$PASSWD
    echo ' yppasswdd\c'                 > /dev/console
fi
```

Remove the comment symbols (#) that precede these lines. The next time the system boots, the **rpc.yppasswdd** daemon will run.

Note: Pay close attention to the file listed in the **PASSWD** argument. You can replace the file name (**/etc/passwd** or **/etc/yp/passwd** usually appears) with the name of the file you have selected as the input file for the NIS password maps.

See *AIX Operating System Commands Reference* for more information on the **yppasswd** command, and the **rpc.yppasswdd** daemon.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.6 Customizing NIS Slave Servers

Customizing workstations to be NIS slave servers involves starting the NIS processes that consult the maps, editing the three local access files that are not replaced by NIS maps (**/etc/passwd**, **/etc/group**, and **/etc/hosts**), and replicating the NIS maps from the NIS master server.

Subtopics
6.10.10.6.1 Starting the ypserv and ypbind Processes
6.10.10.6.2 Editing the Local Access Files
6.10.10.6.3 Transferring Maps From the NIS Master Server

¦ Copyright IBM Corp. 1985, 1991

6.10.10.6.1 Starting the ypserv and ypbind Processes

The **ypserv** process runs as a background daemon on both the NIS master server and NIS slave servers to answer NIS client requests for information. To start **ypserv**, enter the following at the command line:

```
/etc/ypserv
```

If the system can locate the NIS map subdirectory created by the **ypinit** process (the /etc/yp/**domain** directory, which is created in "Transferring Maps From the NIS Master Server" in topic 6.10.10.6.3), **ypserv** starts automatically from the **/etc/rc.nfs** file the next the system is booted.

The **ypbind** process runs as a background daemon on all NIS hosts to find the NIS server with a NIS map that has the information requested by a client. To start the **ypbind** utility, enter the following at the command line:

```
/etc/ypbind
```

If the entry for **ypbind** is uncommented in the **/etc/rc.nfs** file, **ypbind** starts automatically the next time the system is booted.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.6.2 Editing the Local Access Files

In most cases, system processes are automatically redirected to NIS maps for information. For example, a process looking up the name of a network service checks the NIS **services.byname** (or **srvcs.nam**) map instead of the **/etc/services** text file on the workstation when the NIS is running. The workstation text files that are replaced by NIS maps are maintained on the NIS master server only.

The following files, however, may contain information for local use only:

/etc/passwd
/etc/group
/etc/hosts

The **/etc/passwd** and **/etc/group** files are augmented rather than replaced by NIS maps. To maintain private information and still obtain information from the NIS maps, special NIS escape entries can be added to the files. The system processes consult the local files first. If they encounter a NIS escape entry before they find the information for which they are searching, the processes are redirected to the NIS service to continue the search.

On a NIS slave server, the **/etc/passwd** file should contain only those user entries that are designated for local use only. In addition, an entry for **root** should always be maintained in the local **/etc/passwd** file to permit logging in for maintenance when NIS is not running. The final entry should be the NIS escape sequence (**+::0:0::**). This entry instructs a system process to consult the NIS map at this point. It should be the last entry so that all local entries are checked before the NIS map is consulted.

The following is an example of the **/etc/passwd** file that could appear on a NIS slave server:

```
root:!:0:0:Sysop:/:/bin/sh
su:!:0:0::/:
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/usr/adm:
adduser:!:0:0::/usr/adm:/etc/adduser
locusr1:!:220:50:Local User 1:/u/locusr1:bin/sh
locusr2:!:221:50:Local User 2:/u/locusr2:/bin/sh
+::0:0::
```

Note: The **!** is a special place holder that directs system processes to check an additional security file. This is explained in detail in the section on creating password files.

The **/etc/group** file should contain only those group entries that are designated for local use only. The final entry should be the NIS escape sequence (**+:**). This entry instructs a system process to consult the NIS map at this point. It should be the last entry so that all local entries are checked before the NIS map is consulted.

The following is an example of the **/etc/group** file that could appear on a NIS slave server:

```
system:!:0:root,su
```

¦ Copyright IBM Corp. 1985, 1991

```
printq:!:9:root
locgrp:!:50:locusr1,locusr2
+:
```

The TCP/IP program consults the **/etc/hosts** file before NIS is initialized. For this reason, an entry that identifies the local Internet address of the host must appear in the file on a NIS slave server. The only other entry required in this file is the **loopback** entry. No other entries are necessary because the NIS map is consulted in place of the local **/etc/hosts** file. The entries in **/etc/hosts** should take the following form:

```
Internet_address    hostname
127.0.0.1           localhost
```

Replace the **Internet_address** parameter with the Internet address of the local host, and the **hostname** parameter with the name of the local host. For example, the **/etc/hosts** file on the NIS slave server called **slave1** with Internet address **192.200.10.10** would look like the following:

```
192.200.10.10       slave1
127.0.0.1           localhost
```

¦ Copyright IBM Corp. 1985, 1991

6.10.10.6.3 Transferring Maps From the NIS Master Server

In NIS, complete replicas of the master NIS data base located on the NIS master server are transferred to the NIS slave servers. The NIS slave servers must be authorized to copy files remotely from the NIS master server. See the **rcp** (remote copy) command in *AIX TCP/IP User's Guide* for detailed information.

To transfer the replicated NIS maps from the NIS master server to the NIS slave servers, change to the **/etc/yp** directory and issue the following at the command line:

```
cd /etc/yp
ypinit -s master
```

Replace the **master** parameter with the name of the NIS master server.

The **ypinit** procedure creates a new subdirectory under **/etc/yp** corresponding to the name of the NIS domain you have set. It transfers exact replicas of the maps maintained on the NIS master server to this subdirectory. For information on the contents of the maps, see "Creating the NIS Maps on the NIS Master Server" in topic 6.10.10.5.2.

¦ Copyright IBM Corp. 1985, 1991

6.10.10.7 Customizing NIS Clients

Customizing NIS clients is similar to customizing each NIS slave server. Like the NIS slave server, customizing workstations to be NIS clients involves editing the three local access files that are not replaced by NIS maps (**/etc/passwd**, **/etc/group**, and **/etc/hosts**). Follow the procedures listed in "Editing the Local Access Files" in topic 6.10.10.6.2.

The final task is to start the **ypbind** process. As discussed previously, the **ypbind** process runs as a background daemon on all NIS hosts to find the NIS server with a NIS map that has the information requested by a client. To start the **ypbind** utility, enter the following at the command line:

```
/etc/ypbind
```

If the entry for **ypbind** is uncommented in the **/etc/rc.nfs** file, **ypbind** starts automatically the next time the system is booted.

¦ Copyright IBM Corp. 1985, 1991

6.10.11 Maintaining the NIS Environment

It is likely that the NIS environment you have configured will require adjustments from time to time. In fact, in large or complex networks, the NIS environment may change many times a day. This section discusses how to maintain NIS using tools that are supplied with IBM AIX Network File System.

Subtopics

¦ Copyright IBM Corp. 1985, 1991

6.10.11.1 Changing the Default NIS Maps

Changing the NIS maps to reflect updated system information may be a common occurrence in your daily maintenance tasks. System information, such as a new user account or a changed password, can require constant updating. To modify most NIS maps, you will edit the ASCII input file on the NIS master server that contains the original NIS map, rebuild the NIS map, and propagate it to the NIS slave servers. The special handling of NIS password maps is the exception.

Warning: Except for user password changes made with the **yppasswd** command, NIS maps should be modified on the NIS master server only. Modifying NIS maps on NIS slave servers can break the NIS service algorithm, which can result in the NIS map data being unreliable.

Subtopics
6.10.11.1.1 Updating a NIS Map
6.10.11.1.2 Updating NIS Passwords

¦ Copyright IBM Corp. 1985, 1991

6.10.11.1.1 Updating a NIS Map

To update the contents of a default NIS map, you must first edit the text file that is used as the input file for the map. For example, if you were adding a new machine to your network, you would edit the **/etc/hosts** file by adding an entry that lists the new machine's Internet address and host name.

After editing the input file, change to the **/etc/yp** directory and use the **make** procedure to rebuild the new map by issuing the following at the command line:

```
cd /etc/yp
make map_type
```

The **map_type** parameter specifies the NIS maps to be constructed from the input file. For example, issuing the command as **make hosts** causes the host name and the host address NIS maps to be rebuilt.

If you have modified several text files or want to confirm that all NIS maps are updated, issue the **make** command without parameters. The **make** procedure automatically evaluates every input file on the NIS master server. If the file has been modified since the latest NIS map for that file was built, the NIS map is rebuilt.

The **make** procedure automatically propagates any new maps to the NIS slave servers by default. If you prefer not to have the NIS maps propagated automatically, add the **NOPUSH** parameter to the **make** command sequence. You can assign any non-null value to **NOPUSH**. For example, to have the NIS host maps rebuilt but not propagated to the NIS slave servers, the following could be entered at the command line:

```
make hosts NOPUSH=noxfr
```

By deferring the transfer process, you can examine the contents of the updated map or test parts of the map before it is transferred.

When you want to transfer the NIS maps, you do not have to use the **make** command again. You can transfer the maps manually with the **yppush** command. The **yppush** command transfers a NIS map to all hosts listed in a map called **ypservers**. To transfer the maps manually, enter the following at the command line:

```
/etc/yp/yppush mapname
```

The **mapname** parameter specifies the name of the NIS map being transferred.

¦ Copyright IBM Corp. 1985, 1991

6.10.11.1.2 Updating NIS Passwords

Users listed in the NIS password map can log in on any host in a particular domain, regardless of whether they are included in the local host's password file. However, unless they are logged in on the NIS master server and the NIS master server's **/etc/passwd** file is being used for NIS map input, users cannot change their password using the AIX **passwd** command. If the **rpc.yppasswdd** daemon program is running on the NIS master server, users can change their passwords from any host in the domain using the **yppasswd** command.

Note: The **yppasswdd** daemon, which runs on the NIS master server only, is usually invoked when the system starts from the **/etc/rc.nfs** file. See "Starting the rpc.yppasswdd Daemon" in topic 6.10.10.5.4 for more information on running **yppasswdd**.

Activate the **rpc.yppasswdd** daemon on the NIS master server by entering the following command at the command line:

```
/etc/rpc.yppasswdd pw_file [-m make_args]
```

The **pw_file** parameter specifies the path name of the file being used as the input file for the NIS password map. The **rpc.yppasswdd** daemon checks the file named by **pw_file** to verify the old password supplied by the user. If verification succeeds (the old password is valid), **rpc.yppasswdd** replaces the old password in **pw_file** with the new one.

The flag for the **make** procedure (**-m**) causes the **rpc.yppasswdd** daemon to invoke the **make** procedure to rebuild the appropriate NIS maps. Include any arguments in place of the **make_args** parameter. Usually, **passwd** is used as an argument to **make**. Variables defined in **/etc/yp/Makefile** can also be assigned as **make** arguments.

In the following example, **/etc/passwd** on the NIS master server is used as the input file for the NIS password name and ID maps. In this case, the **PASSWD** variable passed to the **make** procedure is set to **/etc/passwd** as follows:

```
/etc/rpc.yppasswdd /etc/passwd -m passwd PASSWD=/etc/passwd
```

When a user enters the **yppasswd** command from any host in the NIS domain, the **rpc.yppasswdd** daemon changes the **/etc/passwd** file on the NIS master server and invokes **make** to rebuild and propagate the password maps.

If the **/etc/yp/passwd** file on the NIS master server is used as the input file for the NIS password name and ID maps, the **PASSWD** variable is set to **/etc/yp/passwd** as shown in the following:

```
/etc/rpc.yppasswdd /etc/yp/passwd -m passwd PASSWD=/etc/yp/passwd
```

See *AIX Operating System Commands Reference* for more information on the **yppasswd** command.

¦ Copyright IBM Corp. 1985, 1991

6.10.11.2 Adding a New NIS Server

If your network configuration grows or changes, it may be necessary to add additional NIS slave servers to support the new configuration. Adding a new NIS server to your configuration involves modifying the **ypservers** map, which is also used by the system to determine which NIS hosts receive the replicas of the NIS maps on the NIS master server.

The procedure for modifying the **ypservers** map differs from other default maps because no text file is used as input for this map. Instead, the **makedbm** utility is used to create the modified **ypservers** maps.

Note: The **makedbm** command only applies to servers, not clients. Do not attempt to run **makedbm** on a client.

Modifying the ypservers Map on the NIS Master Server: First, list the contents of the current **ypservers** map using the **makedbm** command with the **-u** flag. The output of this command, along with the echoed name of the new server, is then used as input to another **makedbm** command that creates the new map temporarily called **tmpsvrs**. Change to the **/etc/yp** directory and enter the following:

```
cd /etc/yp
(makedbm -u domain/ypservers ; echo new_server) | makedbm _ tmpsrvrs
```

Note: The **makedbm** command only applies to servers not clients. Do not attempt to run **makedbm** on a client.

The **domain** parameter specifies the name of the NIS domain (and the directory where the NIS maps are kept). The **new_server** parameter specifies the name of the host being added to the **ypservers** map.

You can check the contents of the new map by entering the following at the command line:

```
makedbm -u tmpsrvrs
```

When you have verified that the new server is included in the **tmpsrvrs** map, use the **mv** (move) command to replace the old **ypservers** map files with the new ones. Enter the following at the command line:

```
mv tmpsrvrs.pag domain/ypservers.pag
mv tmpsrvrs.dir domain/ypservers.dir
```

Modifying the ypservers Map on the NIS Slave Server: After you have added the new server to the **ypservers** map on the NIS master server and updated it, run the **ypinit** procedure on the new server to have all of the NIS maps replicated on it. Change to the **/etc/yp** directory and enter the following at the command line:

```
cd /etc/yp
ypinit -s master
```

The **master** parameter specifies the name of the NIS master server.

Note: You must also start the **ypserv** process on the new NIS slave server. See "Customizing NIS Slave Servers" in topic 6.10.10.6 for information.

¦ Copyright IBM Corp. 1985, 1991

6.10.11.3 Creating a New NIS Map

User information requirements at your site may make it necessary to add new maps to service your NIS domain. Standard text files can be created or filtered through processes such as **awk**, **grep**, and **sed**, and passed as input to the **makedbm** utility that creates the NIS maps. Consult the existing **/etc/yp/Makefile** for examples. It is recommended that you use mechanisms similar to those in the **Makefile** when creating the new maps.

After the new map is created on the NIS master server, it must be transferred to the NIS slave servers to maintain data base consistency throughout the network in one of the following ways:

With the **yppush** command. This is issued from the NIS master server to push the new map from the NIS master server to the NIS slave servers.

With the **ypxfr** command. This issued from a NIS slave server to get the new map from the NIS master server.

To provide ongoing maintenance for the new map (such as updating), create an entry for the new map in **/etc/yp/Makefile**.

The following example shows how to create a new map called **udir.nam** that lists the home directories of users in a NIS domain. The new map is created using filtered input from the NIS password file (in this case, **/etc/yp/passwd**). The keys for the map are the user names, and their corresponding values are the users' home directories. To begin creating the map, change to the **/etc/yp** directory and enter the following at the command line:

```
cd /etc/yp
awk '{FS=":" ; OFS="\t" ; print $1,$6}' /etc/yp/passwd |
    \ makedbm - domain/udir.nam
```

The **domain** parameter specifies the current NIS domain.

Note: The **makedbm** command applies to servers only, not clients. Do not attempt to run **makedbm** on a client.

After the new map **udir.nam** is propagated to the NIS slave servers, any NIS client in the NIS domain can query it for information with the **ypcat** or **ypmatch** commands.

Subtopics
6.10.11.3.1 Keeping Maps on NIS Slave Servers Current
6.10.11.3.2 Network Information Services Entries in Non-NIS Files
6.10.11.3.3 Interpolation of Entries Between Local Files and NIS Maps

¦ Copyright IBM Corp. 1985, 1991

6.10.11.3.1 Keeping Maps on NIS Slave Servers Current

To ensure the information in NIS maps is reliable and consistent throughout a NIS domain, all updates to the maps located on the NIS master server must be propagated to the NIS slave servers. When a map on the NIS master server is updated, the **make** procedure automatically executes the **yppush** utility after rebuilding the map. The **yppush** command notifies all NIS slave servers that a map must be transferred, and the **ypserv** process on each NIS slave server invokes the **ypxfr** command to get the updated map. However, a NIS slave server that is out of service when **yppush** sends notice of the new map does not invoke the **ypxfr** command, which means it retains the earlier version of the NIS map when it returns to the network. To prevent such situations, each NIS slave server can be set to request updated maps from the NIS master server at regular intervals using the **cron** process.

Some maps require update verifications more frequently than others. For example, the password and mail alias maps can change many times a day, and must be checked more frequently than the protocols or services maps, which may not change for months at a time.

The **ypxfr** command requests a single map. To avoid having **crontab** entries for each map, you can group **ypxfr** commands for several maps in a shell script, and have the **cron** process execute the shell script at appropriate intervals. Group the maps together according to how often they need updating. The following shell scripts exist in the **/etc/yp** directory:

ypxfr_1perh for checking each hour
ypxfr_1perd for checking once a day
ypxfr_2perd for checking twice a day.

Modify these shell scripts as necessary to meet the requirements of your site.

Use the **crontab** command to have the shell scripts executed automatically at specific intervals.

Note: Before you make new entries for the **ypxfr** scripts, it is important to copy the current **crontab** entries for **root** to a temporary file. This prevents the old entries from being lost when **cron** processes the new file. To create a temporary file to hold the current entries, enter the following at the command line:

```
crontab -l > filename
```

The **filename** parameter specifies the temporary file you have created.

Using a text editor, add your new entries for the **ypxfr** procedures to the temporary file. Modify the execution schedule for each entry according to how often each **ypxfr** script should be run, as in the following examples:

To execute the **ypxfr_1perh** script at 37 minutes after every hour of the day:

```
37 * * * * /etc/yp/ypxfr_1perh
```

To execute the **ypxfr_2perd** script at 11:00 a.m. and 11:00 p.m. each day:

¦ Copyright IBM Corp. 1985, 1991

```
0 11,23 * * * /etc/yp/ypxfr_2perd
```

To execute the **ypxfr_1perd** script at 3:30 a.m. each day:

```
30 3 * * * /etc/yp/ypxfr_1perd
```

Note: It is recommended that the request times differ for each NIS slave server to prevent overloading the NIS master server with requests from all NIS slave servers at the same time.

After the entries have been made, issue the following at the command line to have **cron** read the file and execute the procedures at the appropriate times:

crontab **filename**

Although the **ypxfr** command can also be issued at the command line, this use should be reserved for special circumstances such as when creating a test NIS environment or bringing a NIS slave server into service after it has been off-line.

You can maintain a record of the **ypxfr** activity in a log file you can create and call **/etc/yp/ypxfr.log**. (Remember to trim this file periodically in the same manner as you trim your other administrative logs.) To discontinue **ypxfr** logging, remove the **/etc/yp/ypxfr.log** file.

¦ Copyright IBM Corp. 1985, 1991

6.10.11.3.2 Network Information Services Entries in Non-NIS Files

Two files used to validate remote access from other hosts, **/etc/hosts.equiv** and **/.rhosts**, are not used as input for NIS maps, but these files can have entries that refer to the Network Information Services. To direct system processes to consult NIS from these files, use one of the following:

A single **+** on a line by itself, which directs a system process to consult NIS for all accesses

A **+@** or **-@** appearing before the name of a network group, which permits or restricts access by members of that network group.

Using a single **+** as an entry in the **hosts.equiv** file means that only the **/etc/passwd** file (and associated NIS map, if being used) is consulted to validate users remotely logging in on this system. Users listed in the password file or NIS map are permitted to log in without a password.

Note: It is not recommended to have unrestricted access in the **/.rhosts** file (**.rhosts** for the root user). The single **+** entry effectively permits any root user from another host to log in as root on this system without a password.

The following example demonstrates the use of netgroup entries in the **hosts.equiv** and **/.rhosts** files:

```
+@netgroup1
-@netgroup2
```

The hosts listed in **netgroup1** are considered trusted. This means users attempting to log in remotely from a host in **netgroup1** can log in without a password if they are listed in **/etc/passwd** or the associated NIS map. The hosts listed in **netgroup2**, on the other hand, are not trusted. This means that users attempting to log in remotely from a host in **netgroup2** must supply their password.

See *AIX Operating System Technical Reference* for information on the **netgroup** file format.

¦ Copyright IBM Corp. 1985, 1991

6.10.11.3.3 Interpolation of Entries Between Local Files and NIS Maps

You can direct system processes that consult the local files **/etc/passwd** and **/etc/group** to use certain information from the local file while using other information from the associated NIS map. The plus (**+**) symbol is placed at the beginning of a file entry to direct the system process to use the values from the fields in the associated NIS map for those fields left blank in the entry while using the local values that are specified.

For users with different locales, use ASCII characters only to ensure that files can be accessed successfully.

For example, an entry in the **/etc/passwd** file usually appears in the following form:

user_namepassword:userid:groupid:account_information:home_directory:login_sh

A system process reads the values of these fields locally. To have the system process interpolate entries from the NIS map and the local file, the entry could appear as follows:

+**username**:::**account_information:home_directory**:

In this example, the entry directs the system process to get the password, user ID, group ID, and login shell values from the fields in the associated NIS password map, to use the **additional_data** and **home directory** values listed in the local file. Another way to think of it is that the local values for the **additional_data** and **home_directory** override the values in the NIS map.

Note: The user ID (UID) and the group ID (GID) numerical entries cannot be overridden locally if NIS interpolation is used on a password file or group file entry.

In this example, the following entry in the **/etc/group** file directs the system process to get the group password and GID fields for the review group from the YP, but defines **joe** and **jane** as the members of the group, which overrides any members listed in the corresponding entry in the NIS map:

+review:::joe,jane

Network group entries can also be included by using the **+@** with the network group name. In the following example entry, all users in the **novices** netgroup use the locally defined **GCOS** and login shell fields while other fields are obtained from the NIS for each user in the netgroup:

+@novices::::A Novice User::/usr/local/help_menu

The **+@** and **-@** signifiers can also be used in combination with netgroup names in the **/etc/hosts.equiv** and **/.rhosts** files as discussed in "Network Information Services Entries in Non-NIS Files" in topic 6.10.11.3.2.

¦ Copyright IBM Corp. 1985, 1991

A.0 Appendix A. Printer Control Codes

Table A-1 lists the standard printer control codes that you can use with attached printers. If the printer can perform the function by itself, the system passes the code directly to the printer. If the codes do not work on the printer installed on your system, the printer support system performs one of the following actions:

Tries to emulate the control with functions that the installed printe does have.

Removes the control from the output stream. The function is no performed.

The three code columns in the table show different representations of the same code, depending on how you enter the code into the data stream.

Control Name Shows the name of the control character. In many cases the name is the same as the keyboard key that produces the required ASCII code for the control code.

Hex Code Shows the hexadecimal representation of the control code.

ASCII Code Shows the decimal representation of the control code.

Table A-1. Printer Control Codes

Category	Function Performed	Control Name	Hex Code	ASCII Code
Control:	Provides a null value. Used as a list terminator.	**NUL**	**00**	**0**
	Sounds the buzzer. (2)	**BEL**	**07**	**7**
	Prints the next character as a printable character. The next character is a control with an ASCII value of less than 32.	**ESC ^**	**1B5E**	**27 94**
	Prints more	**ESC \ m n c**	**1B5C m n c**	**27 92 m n c**

Copyright IBM Corp. 1985, 1991

	than one character with an ASCII value that is below 32.			
	Clears the printer memory of all data waiting to be printed following the last received line feed. (1)	**CAN**	**18**	**24**
	Sets resolution for raster image print (**n** indicates a string of control bytes). (1)	**ESC [O n**	**1B5B4F n**	**27 91 79 n**
	Performs a printer power-on reset. (1)	**ESC [K**	**1B5B4B 0100 0**	**27 91 75 1 0 0**
Positionin the	Sets back space.	**BS**	**08**	**8**
Printhead:	Sets horizontal tab.	**HT**	**09**	**9**
	Sets horizontal tabs (**n** is a list of one or more tab positions).	**ESC D n NUL**	**1B44n00**	**27 68 n 0**
	Sets tab stops to power-on settings.	**ESC R**	**1B52**	**27 82**
	Sets line feed.	**LF**	**0A**	**10**

Copyright IBM Corp. 1985, 1991

	Sets reverse line feed.	**ESC]**	**1B5D**	**27 93**
	Starts automatic line feed.	**ESC 5 1**	**1B35 1**	**27 53 1**
	Stops automatic line feed.	**ESC 5 0**	**1B35 0**	**27 53 0**
	Provides a carriage return (no line feed).	**CR**	**0D**	**13**
	Provides a vertical tab.	**VT**	**0B**	**11**
	Sets vertical tabs (**n** is a list of tab positions).	**ESC B n NUL**	**1B42 n NUL**	**27 66 n NUL**
Paper Control:	Provides a form feed.	**FF**	**0C**	**12**
	Sets top of forms. (2)	**ESC 4**	**1B34**	**27 52**
	Ignores end of forms. (2)	**ESC 8**	**1B38**	**27 56**
	Respects end of forms. (2)	**ESC 9**	**1B39**	**27 57**
	Sets skip perforation (2) (**n** is lines to skip).	**ESC N n**	**1B4E n**	**27 78 n**
	Stops skip perforation.	**ESC O**	**1B4F**	**27 79**

Copyright IBM Corp. 1985, 1991

	(2)			
Formatting the Page	Uses 12 characters-per printing.	**ESC :** inch	**1B3A**	**27 58**
Image:	Sets 1/8" line spacing.	**ESC 0**	**1B30**	**27 48**
	Starts **n**/72" line spacing.	**ESC 2**	**1B32**	**27 50**
	Sets **n**/72" line spacing.	**ESC A n**	**1B41 n**	**27 59 n**
	Sets page length. (2) (**n** is lines per page).	**ESC C n**	**1B43 n**	**27 67 n**
	Sets page length (2) (**n** is inches per page).	**ESC C 0 n**	**1B43 0 n**	**27 67 0 n**
	Sets left and right margins (**m** and **n** are column numbers).	**ESC X m n**	**1B58 m n**	**27 88 m n**
	Sets top and bottom margins (**m** and **n** are length of control; **t1 t0** are high/low-order bytes of top margin; **b1 b0** are high/low-order bytes of bottom margin).	**ESC [S m n t1 t0 b1 b0**	**1B5B53 m n t1 t0 b1 b0**	**27 91 83 m n t1 t0 b1 b0**
	Starts automatic	**ESC M 1**	**1B4D 1**	**27 77 1**

Copyright IBM Corp. 1985, 1991

	line justification.			
	Stops automatic line justification.	**ESC M 0**	**1B4D 0**	**27 77 0**
	Starts proportional spacing.	**ESC P 1**	**1B50 1**	**27 80 1**
	Stops proportional spacing.	**ESC P 0**	**1B50 0**	**27 80 0**
	Sets print resolution for draft quality font and **ESC K** image data (**n** indicates the resolution value). (1)	**ESC [0 n**	**1B5B30 0100 n**	**27 91 48 1 0 n**
	Sets presentation surface color (**n** indicates the available colors). (1)	**ESC [L n**	**1B5B4C 0200 00 n**	**27 91 76 2 0 n**
Controllin the Ribbon	Sets color band 1 (yellow).	**ESC y**	**1B79**	**27 121**
	Sets color band 2 (magenta).	**ESC m**	**1B6D**	**27 109**
	Sets color band 3 (cyan).	**ESC c**	**1B63**	**27 99**
	Sets color band 4 (black).	**ESC b**	**1B62**	**27 98**
	Sets automatic ribbon band shift.	**ESC a**	**1B61**	**27 97**

Copyright IBM Corp. 1985, 1991

	Sets the background color for **Text**, **ESC K**, or **ESC [O** printing (**n** indicates the available colors). (1)	**ESC [N n**	**1B5B4E 0100 n**	**27 91 78 1 0 n**
	Sets the foreground color for **Text**, **ESC K**, or **ESC [O** printing (**n** indicates the available colors). (1)	**ESC [M n**	**1B5B4D 0100 n**	**27 91 77 1 0 n**
	Swaps the foreground and background printing color (**n**= 0 indicates normal printing; **n**= 1 indicates reversing background and foreground colors). (1)	**ESC []**	**1B5B5D 0100 n**	**27 91 93 1 0 n**
Selecting Print	Starts double-wide printing.	**SO**	**0E**	**14**
Mode:	Stops double-wide printing.	**DC4**	**14**	**20**
	Starts double-wide continuous printing.	**ESC W 1**	**1B57 1**	**27 87 1**
	Stops double-wide continuous printing.	**ESC W 0**	**1B57 0**	**27 87 0**

Copyright IBM Corp. 1985, 1991

	Starts compressed printing.	**SI**	**0F**	**15**
	Stops compressed printing.	**DC2**	**12**	**18**
	Starts underline printing.	**ESC -1**	**1B2D 1**	**27 45 1**
	Stops underline printing.	**ESC -0**	**1B2D 0**	**27 45 0**
	Starts emphasized printing.	**ESC E**	**1B45**	**27 69**
	Stops emphasized printing.	**ESC F**	**1B46**	**27 70**
	Starts double-strike printing.	**ESC G**	**1B47**	**27 71**
	Stops double-strike printing.	**ESC H**	**1B48**	**27 72**
	Starts superscript printing.	**ESC S 0**	**1B53 0**	**27 83 0**
	Starts subscript printing.	**ESC S 1**	**1B53 1**	**27 83 1**
	Stops superscript or subscript printing.	**ESC T**	**1B54**	**27 84**
	Starts color underline, bypassing white space (**n** defines	**ESC [B 1 n**	**1B5B42 0200 1 n**	**27 91 66 2 0 1 n**

Copyright IBM Corp. 1985, 1991

	the available color). (1)			
	Starts continuous color underline (**n** defines the available color). (1)	**ESC [B 0 n**	**1B5B42 0200 0 n**	**27 91 66 2 0 0 n**
	Stops underline.	**ESC [E**	**1B5B450000**	**27 91 69 0 0**
Selecting Character	**hU**ses PC character set 2.	**ESC 6**	**1B36**	**27 54**
Set:	Uses PC character set 1.	**ESC 7**	**1B37**	**27 55**
	Selects font (**n** specifies the font; varies with printer type).	**ESC I n**	**1B49 n**	**27 73 n**
	Sets graphic set ID (**c** selects graphic set 0, 1 or 2).	**ESC [T 1 0 c**	**1B5B54 1 0 c**	**27 91 84 1 0 c**
Using Bit Image Graphics:	Sets bit graphics (normal (**n** is a string of control bytes).	**ESC K n**	**1B4B n**	**27 75 n**
	Sets graphics dual-half speed (**n** is a string of control bytes).	**ESC L n**	**1B4C n**	**27 76 n**
	Sets bit graphics dual-normal	**ESC Y n**	**1B59 n**	**27 89 n**

Copyright IBM Corp. 1985, 1991

	speed (**n** is a string of control bytes).			
	Sets bit graphics high-half speed (**n** is a string of control bytes).	**ESC Z n**	**1B5A n**	**27 90 n**
	Sets aspect ratio to 1:1.	**ESC n 1**	**1B6E 1**	**27 110 1**
	Sets aspect ratio to 5:6.	**ESC n 0**	**1B6E 0**	**27 110 0**
	Moves carriage to home position.	**ESC <**	**1B3C**	**27 60**
	Moves right **n**/120.	**ESC d n**	**1B64 n**	**27 100 n**
	Moves left **n**/120.	**ESC e n**	**1B65 n**	**27 101 n**
	Starts unidirectional printing.	**ESC U 1**	**1B551**	**27 85 1**
	Stops unidirectional printing.	**ESC U 0**	**1B550**	**27 85 0**
	Sets 7 dot line spacing.	**ESC 1**	**1B31**	**27 49**
	Sets graphics line spacing (**n** is the number of 1/216-inch	**ESC 3 n**	**1B33 n**	**27 51 n**

Copyright IBM Corp. 1985, 1991

	steps).			
	Sets variable space line feed (**n** is the number of 1/216-inch steps).	**ESC J n**	**1B4A n**	**27 74 n**
	Moves vertical presentation down the page in number of dots (**n** indicates how far to move the presentation). (1)	**ESC [U n**	**1B5B55 0100 n**	**27 91 85 1 0 n**
Selecting Printer:	Selects the printer to accept data. (2)	**DC1**	**11**	**17**
	Deselects the printer so as to not receive data. (2)	**DC3**	**13**	**19**
	Sets initialize function on. (2)	**ESC ? 1**	**1B3F 1**	**27 63 1**
	Sets initialize function off. (2)	**ESC ? 0**	**1B3F 0**	**27 63 0**
	Queries a parallel attached printer for identification If the device queried is equal to **n**, this printer deactivates the select line. (2)	**ESC Q n**	**1B51 n**	**27 81 n**

(1) Use with the Color Jet Printer.

(2) Do not use these controls when using the print queue.

Copyright IBM Corp. 1985, 1991

(3) These controls may not work on the installed printer. Use **passthrough** mode to send these codes to the printer.

¦ Copyright IBM Corp. 1985, 1991

GLOSSARY Glossary

access. To obtain data from or put data in storage.

access permission. A group of designations that determine who can access a particular AIX file and how the user may access the file.

account. The log in directory and other information that give a user access to the system.

activity manager. A collection of system-supplied tasks allowing users to manage their activities. Provides the ability to list current activities (Activity List) and to begin, cancel, hide, and activate activities.

administrative event. A specified group of base events for which a user can be audited.

All Points Addressable (APA) display. A display that allows each pel to be individually addressed. An APA display allows for images to be displayed that are not made up of images predefined in character boxes. Contrast with *character display*.

allocate. To assign a resource, such as a disk file or a diskette file, to perform a specific task.

alphanumeric character. Consisting of letters, numbers, and often other symbols; such as, punctuation marks and mathematical symbols.

American National Standard Code for Information Interchange (ASCII). The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

American National Standards Institute. An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary industry standards.

APAR. Authorized Program Analysis Report.

application. A program or group of programs that apply to a particular business area; such as, the Inventory Control or the Accounts Receivable application.

application program. A program used to perform an application or part of an application.

¦ Copyright IBM Corp. 1985, 1991

argument. Numbers, letters, or words that change the way a command works.

ASCII. See *American National Standard Code for Information Interchange*.

attribute. A characteristic. For example, the attribute for a displayed field could be blinking.

audit. To review and examine the activities of a data processing system mainly to test the adequacy and effectiveness of procedures for data security and data integrity.

audit pipe. A chain of filter programs connected by pipes.

audit trail. (1) Data, in the form of a logical path linking a sequence of events, used for tracing the transactions that have affected the contents of a record. (2) Information that allows tracing of the history of things such as a customer account of item record.

auditing class. A list of administrative auditing events that define which security-relevant events are recorded for a user. They are defined by the system administrator in the user database.

authentication. In computer security, a process used to verify the user of an information system or protected resources.

authorized program analysis report (APAR). A report of a problem caused by a suspected defect in a current release of a program.

auto carrier return. The system function that places carrier returns automatically within the text and on the display. This is accomplished by moving whole words that exceed the line end zone to the next line.

backend. The program that sends output to a particular device. There are two types of backends: friendly and unfriendly.

background process. (1) A process that does not require operator intervention that can be run by the computer while the workstation is used to do other work. (2) A mode of program execution in which the shell does not wait for program completion before prompting the user for another command.

backup copy. A copy, usually of a file or group of files, that is kept in case the original file or files are unintentionally changed or destroyed.

backup diskette. A diskette containing information copied from a fixed disk or from another diskette. It is used in case the original information becomes unusable.

¦ Copyright IBM Corp. 1985, 1991

bad block. A portion of a disk that can never be used reliably.

base event. An action for which a user may be audited.

base address. The beginning address for resolving symbolic references to locations in storage.

base name. The last element to the right of a full path name. A filename specified without its parent directories.

Basic Networking Utility (BNU). A utility based on the UNIX to UNIX Copy (UUCP) network facility that provides communications support. It uses hardwired, telephone, or IBM token-ring network or Ethernet lines.

batch printing. Queueing one or more documents to print as a separate job. The operator can type or revise additional documents at the same time. This is a background process.

batch processing. A processing method in which a program or programs process records with little or no operator action. This is a background process. Contrast with *interactive processing*.

binary. (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Involving a choice of two conditions; such as, on-off or yes-no.

bit. Either of the binary digits 0 or 1 used in computers to store information. See also *byte*.

block. (1) A group of records that is recorded or processed as a unit. Same as *physical record*. (2) In data communications, a group of records that is recorded, processed, or sent as a unit. (3) A block is 1024 bytes long. (4) A logical block is 4096 bytes long. The system internally manages file systems using this block size.

block file. A file listing the usage of blocks on a disk.

block special file. A special file that provides access to an input or output device is capable of supporting a file system. See also *character special file*.

BNU. See *Basic Networking Utility*.

bootstrap. A small program that loads larger programs during system initialization.

¦ Copyright IBM Corp. 1985, 1991

branch. In a computer program, an instruction that selects one of two or more alternative sets of instructions. A conditional branch occurs only when a specified condition is met.

breakpoint. A place in a computer program, usually specified by an instruction, where execution may be interrupted by external intervention or by a monitor program.

buffer. (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written.

burst pages. On continuous-form paper, pages of output that can be separated at the perforations.

byte. The amount of storage required to represent one character; a byte is 8 bits.

call. (1) To activate a program or procedure at its entry point. Compare with **load**.

callouts. An AIX kernel parameter establishing the maximum number of scheduled activities that can be pending simultaneously.

cancel. To end a task before it is completed.

carrier return. (1) In text data, the action causing line ending formatting to be performed at the current cursor location followed by a line advance of the cursor. Equivalent to the carriage return of a typewriter. (2) A keystroke generally indicating the end of a command line.

case sensitive. Able to distinguish between uppercase and lowercase letters.

CCITT. For Comite Consultative International Telegraphe¦ et Telephone¦, the organization which oversees data communications in France.

character display. A display that uses a character generator to display predefined character boxes of images (characters) on the screen. This kind of display cannot address the screen any less than one character box at a time. Contrast with *All Points Addressable display*.

character position. On a display, each location that a character or symbol can occupy.

¦ Copyright IBM Corp. 1985, 1991

character set. (1) A finite set of different characters that is complete for a given purpose, for example, the character set in ISO Standard 646, "7-bit Coded Character Set for Information Processing Interchange." (2) An ordered set of unique representations called characters; for example, the 26 letters of the English alphabet, Boolean 0 and 1, the set of symbols in the Morse code, and the 128 ASCII characters. (3) A defined collection of characters. (4) All the valid characters for a programming language or for a computer system. (5) A group of characters used for a specific reason, for example, the set of characters a printer can print.

character special file. A special file that provides access to an input or output device. The character interface is used for devices that do not use block I/O. See also *block special file*.

character string. A sequence of consecutive characters.

character variable. The name of a character data item whose value may be assigned or changed while the program is running.

child. (1) Pertaining to a secured resource, either a file or library, that uses the user list of a parent resource. A child resource can have only one parent resource. (2) In the AIX Operating System, child is a *process* spawned by a parent process that shares resources of parent process. Contrast with *parent*.

C language. A general-purpose programming language that is the primary language of the AIX Operating System.

class. Pertaining to the I/O characteristics of a device. AIX devices are classified as block or character.

client. A computer or process that accesses the data, services, or resources of another computer or process on the network.

cluster. (1) A station that consists of a control unit (a cluster controller) and the terminals attached to it. See also alternate index cluster, base cluster. (2) On an IBM personal computer, a particular measure of space on a diskette that DOS establishes when it formats the diskette; DOS then allocates space to files in cluster increments. For a single-sided diskette, a cluster is a sector. For dual-sided diskettes, a cluster is two consecutive sectors.

code. (1) Instructions for the computer. (2) To write instructions for the computer; to *program*. (3) A representation of a condition, such as an error code.

code segment. See *segment*.

¦ Copyright IBM Corp. 1985, 1991

collating sequence. The sequence in which characters are ordered within the computer for sorting, combining, or comparing.

color display. A display device capable of displaying more than two colors and the shades produced via the two colors, as opposed to a monochrome display.

column headings. Text appearing near the top of columns of data for the purpose of identifying or titling.

command. A request to perform an operation or run a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command.

command interpreter. A program that sends instructions to the kernel; also called an interface.

command line. The area of the screen where commands are displayed as they are typed.

command line editing keys. Keys for editing the command line.

command name. (1) The first or principal term in a command. A command name does not include parameters, arguments, flags, or other operands. (2) The full name of a command when an abbreviated form is recognized by the computer (for example, print working directory for **pwd**).

command programming language. Facility that allows programming by the combination of commands rather than by writing statements in a conventional programming language.

compile. (1) To translate a program written in a high-level programming language into a machine language program. (2) The computer actions required to transform a source file into an executable object file.

compress. (1) To move files and libraries together on disk to create one continuous area of unused space. (2) In data communications, to delete a series of duplicate characters in a character string.

computer virus. A program that can vandalize your files, although it usually does its defined task. It can spread itself to other files and directories on the system.

concatenate. (1) To link together. (2) To join two character strings.

concurrent group set. A list of the IDs of the various groups to which a

¦ Copyright IBM Corp. 1985, 1991

user belongs.

condition. An expression in a program or procedure that can be evaluated to a value of either true or false when the program or procedure is running.

configuration. The group of machines, devices, and programs that make up a computer system. See also *system customization*.

configuration file. A file that specifies the characteristics of a system or subsystem, for example, the AIX queueing system.

consistent. Pertaining to a file system, without internal discrepancies.

console. (1) The main AIX display station for that site. (2) A device name associated with the main AIX display station for that site.

constant. A data item with a value that does not change. Contrast with *variable*.

context search. A search through a file whose target is a character string.

control block. A storage area used by a program to hold control information.

control commands. Commands that allow conditional or looping logic flow in shell procedures.

control program. Part of the AIX Operating System that determines the order in which basic functions should be performed.

controlled cancel. The system action that ends the job step being run and saves any new data already created. The job that is running can continue with the next job step.

coredump. A kernel memory image dump that is given a unique name so that it will not be over-written in case of another failure. This enables the system administrator to analyze the dump and determine the cause of failure at some later time.

crash. An unexpected interruption of computer service, usually due to a serious hardware or software malfunction.

coupler. A device connecting a modem to a telephone network.

¦ Copyright IBM Corp. 1985, 1991

CPU. Central Processing Unit.

CSS. Current Synchronized Site.

current directory. The directory that is the starting point for relative path names.

current line. The line on which the cursor is located.

current working directory. See *current directory*.

cursor. (1) A movable symbol (such as an underline) on a display, used to indicate to the operator where the next typed character will be placed or where the next action will be directed. (2) A marker that indicates the current data access location within a file.

cursor movement keys. The directional keys used to move the cursor.

customize. To describe (to the system) the devices, programs, users, and user defaults for a particular data processing system.

cylinder. All fixed disk or diskette tracks that can be read or written without moving the disk drive or diskette drive read/write mechanism.

daemon. See *daemon process*.

daemon process. A process begun by the root or the root shell that can be stopped only by the root. Daemon processes generally provide services that must be available at all times such as sending data to a printer.

data base. A collection of information used for a specific purpose.

data block. See *block*.

data circuit terminating equipment (DCE). Equipment that provides the signal conversion and coding between the data terminal equipment (DTE) and the line.

data communications. The transmission of data between computers, or remote devices or both (usually over long distance).

data stream. All information (data and control information) transmitted over a data link.

¦ Copyright IBM Corp. 1985, 1991

data terminal equipment (DTE). That part of a data station that serves as a data source, data sink, or both.

dbm. Format for the Network Information Services data base files.

DCE. See Data Circuit Terminating Equipment.

debug. (1) To detect, locate, and correct mistakes in a program. (2) To find the cause of problems detected in software.

default. A value that is used when no alternative is specified by the operator.

default directory. The directory name supplied by the operating system if none is specified.

default drive. The drive name supplied by the operating system if none is specified.

default value. A value stored in the system that is used when no other value is specified.

dependent workstation. A workstation having little or no standalone capability, that must be connected to a host or server in order to provide any meaningful capability to the user.

device. An electrical or electronic machine that is designed for a specific purpose and that attaches to your computer; for example, a printer, plotter, disk drive, and so forth.

device driver. A program that operates a specific device; such as a printer, disk drive, or display.

device manager. Collection of routines that act as an intermediary between device drivers and virtual machines for complex interfaces. For example, supervisor calls from a virtual machine are examined by a device manager and are routed to the appropriate subordinate device drivers.

device name. A name reserved by the system that refers to a specific device.

diagnostic. Pertaining to the detection and isolation of an error.

diagnostic aid. A tool (procedure, program, reference manual) used to detect and isolate a device or program malfunction or error.

¦ Copyright IBM Corp. 1985, 1991

diagnostic routine. A computer program that recognizes, locates, and explains either a fault in equipment or a mistake in a computer program.

directory. A type of file containing the names and controlling information for other files or other directories.

disable. To make nonfunctional.

discipline. Pertaining to the order in which requests are serviced; for example, first-come-first-served (fcfs) or shortest job next (sjn).

disk I/O. Fixed disk input and output.

diskette. A thin, flexible magnetic plate that is permanently sealed in a protective cover. It can be used to store information copies from the disk or another diskette.

diskette drive. The mechanism used to read and write information on diskettes.

display device. An output unit that gives a visual representation of data.

display screen. The part of the display device that displays information visually.

display station. A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or received from the computer.

distributed file system. A file system whose files, directories, and other components are stored on different sites in a particular cluster.

distributed operating system. An operating system where multiple machines cooperate to seem like one machine.

distributed processing. Results when a user involves multiple cluster sites in a single operation-for example by editing a remote file and starting a task on another cluster site using the on, fast, fastsite, and migrate commands.

Distributed Services (DS). A licensed program that allows you to share files with other AIX systems in a network. You can mount the file systems located on other AIX systems to created file trees that are independent of the file systems.

¦ Copyright IBM Corp. 1985, 1991

DTE. See Data Terminal Equipment.

dump. (1) To copy the contents of all or part of storage, usually to an output device. (2) Data that has been dumped.

dump diskette. A diskette that contains a dump or is prepared to receive a dump.

dump formatter. Program for analyzing a dump.

EBCDIC. See *extended binary-coded decimal interchange code*.

EBCDIC character. Any one of the symbols included in the 8-bit EBCDIC set.

edit. To modify the values, form, or format of data.

edit buffer. A temporary storage area used by an editor.

editor. A program used to enter and modify programs, text, and other types of documents and data.

effective ID. The ID, either group or user, that is used to run a process. It can be set by a program to either the real or saved ID.

emulation. Imitation; for example, when one computer imitates the characteristics of another computer.

enable. To make functional.

END OF FILE. The user-definable character used to indicate end-of-file. By default, **END OF FILE** is set to be **Ctrl-D**. For more information about user-definable characters, see the **termio** file format entry in *AIX Operating System Technical Reference*.

enter. To send information to the computer by pressing the **Enter** key.

entry. A single input operation on a workstation.

enumerate. Returns values from a NIS (Network Information Services) data base in a specified order.

¦ Copyright IBM Corp. 1985, 1991

environment. The settings for shell variables and paths set associated with each process. These variables can be modified later by the user.

error-correct backspace. An editing key that performs editing based on a cursor position; the cursor is moved one position toward the beginning of the line, the character at the new cursor location is deleted, and all characters following the cursor are moved one position toward the beginning of the line (to fill the vacancy left by the deleted element).

escape character. A character that suppresses the special meaning of one or more characters that follow.

Ethernet. A physical network medium through which computers in the same or different cluster can communicate and share files.

exit value. A numeric value that a command returns to indicate whether it completed successfully. Some commands return exit values that give other information, such as whether a file exists. Shell programs can test exit values to control branching and looping.

expression. A representation of a value. For example, variables and constants appearing alone or in combination with operators.

extended binary-coded decimal interchange code (EBCDIC). A set of 256 eight-bit characters.

feature. A programming or hardware option, usually available at an extra cost.

field. (1) An area in a record or panel used to contain a particular category of data. (2) The smallest component of a record that can be referred to by a name.

Field Serviceable Unit. The smallest serviceable unit of an LPP.

FIFO. See *first-in-first-out*.

file. A collection of related data that is stored and retrieved by an assigned name.

file locking. AIX Operating System allows many processes to synchronize simultaneous access to a file through the use of the file locking system calls. These system calls allow a program to lock and unlock portions of an open file. The program can use either a read lock or a write lock.

A ***read lock*** prevents any other process from setting a write lock on any portion of the protected area. When a read lock is set on a segment of a file, other processes can also set read locks on that segment or portion

¦ Copyright IBM Corp. 1985, 1991

of it. The file descriptor, on which a read lock is being placed, must have been opened with read access.

The ***write lock*** prevents any other process from setting a read lock or a write lock on any portion of the protected area. Only one write lock and no read locks can exist for a given segment of a file at a given time. The file descriptor on which a write lock is being placed must have been opened with write access.

file name. The name used by a program to identify a file. See also *label*.

filename. In DOS, that portion of the file name that precedes the extension.

file specification (filespec). The name and location of a file. A file specification consists of a drive specifier, a path name, and a file name.

file system. A collection of files and directories stored on logical and physical devices (such as disks) and logically organized in a hierarchical fashion.

filetab. An AIX kernel parameter establishing the maximum number of files that can be open simultaneously.

filter. A command that reads standard input data, modifies the data, and sends it to standard output.

filter programs. Programs designed to accept information from input, process the data, and write the results to standard output.

first-in-first-out (FIFO). A named permanent pipe. A FIFO allows two unrelated processes to exchange information using a pipe connection.

first level interrupt handler (FLIH). A routine that receives control of the system as a result of a hardware interrupt. One FLIH is assigned to each of the six interrupt levels.

fixed disk. A flat, circular, nonremovable plate with a magnetizable surface layer on which data can be stored by magnetic recording.

fixed disk drive. The mechanism used to read and write information on fixed disk.

flag. A modifier that appears on a command line with the command name that defines the action of the command. Flags in the AIX Operating System almost always are preceded by a dash.

¦ Copyright IBM Corp. 1985, 1991

font. A family or assortment of characters of a given size and style.

foreground. A mode of program execution in which the shell waits for the program specified on the command line to complete before returning your prompt.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) The pattern which determines how data is recorded.

formatted diskette. A diskette on which control information for a particular computer system has been written but which may or may not contain any data.

free list. A list of available space on each file system. This is sometimes called the free-block list.

free-block list. See *free list*.

FSU. See *Field Serviceable Unit*.

full path name. The name of any directory or file expressed as a string of directories and files beginning with the root directory.

function. A synonym for procedure. The C language treats a function as a data type that contains executable code and returns a single value to the calling function.

function keys. Keys that request actions but do not display or print characters. Included are the keys that normally produce a printed character, but when used with the code key produce a function instead.

generation. For some remote systems, the translation of configuration information into machine language.

Gid. See *group number*.

global. Pertains to information available to more than one program or subroutine.

global action. An action having general applicability, independent of the context established by any task.

global character. The special characters * and ? that can be used in a file specification to match one or more characters. For example, placing

¦ Copyright IBM Corp. 1985, 1991

a ? in a file specification means any character can be in that position.

Global File System (GFS). The entire composite file system of the AIX cluster. It consists of the root file system from the primary site, plus all the mounted file systems from secondary sites. See global file system number.

Global File System (GFS) Number. In AIX, every mounted file system is identified by the global file system number (GFS). In a normal file system, this number is hardcoded in the superblock of the physical device where that file system actually resides. For a remote NFS file system, however, there is no such hardcoding; an arbitrary assignment is made by a System Administrator.

The operating system distinguishes every file system from every other by its GFS number. Each mounted file system is assigned a particular machine to serve as its current synchronized site (CSS). If any file system should somehow be assigned two GFS numbers, it would be considered two different file systems by the operating system and might well be entrusted to two separate CSS's.

global search. The process of having the system look through a document for specific characters, words, or groups of characters.

global variable. A symbol defined in one program module but used in other independently assembled program modules.

graphic character. A character that can be displayed or printed.

group name. A name that uniquely identifies a group of users to the system.

group number (Gid). A unique number assigned to a group of related users. The group number can be substituted in commands that take a group name as an argument.

hardware. The equipment, as opposed to the programming, of a computer system.

header. Constant text that is formatted to be in the top margin of one or more pages.

header label. A special set of records on a diskette describing the contents of the diskette.

here document. Data contained within a shell program or procedure (also called *inline input*).

¦ Copyright IBM Corp. 1985, 1991

HEX. For hexadecimal. A numbering system that uses 16 digits (hex=6, decimal=10).

hierarchical tree structure. The organization of files similar to tree structured directories with each file like a small branch of a larger branch that represents the file's parent directory. A directory can also be contained in another (higher level) directory, with the parent of all directories represented by the tree's root. (Therefore, it is called the root directory, or just root.)

highlight. To emphasize an area on the display by any of several methods, such as brightening the area or reversing the color of characters within the area.

history. A C-shell mechanism that lists previously executed commands, which can be re-executed with the **!** command.

history file. A file containing a log of system actions and operator responses.

hog factor. In system accounting, an analysis of how many times each command was run, how much processor time and memory it used, and how intensive that use was.

home directory. (1) A directory associated with an individual user after logging in.

home site. The computer that stores the modifiable copy of a user's home directory. This is the cluster site with the primary copy of his home directory if it is replicated. A user typically logs in to the computer that is his home site.

I/O. See *input/output*.

IF expressions. Expressions within a procedure, used to test for a condition.

indirect block. A block containing pointers to other blocks. Indirect blocks can be single-indirect, double-indirect, or triple-indirect.

informational message. A message providing information to the operator that does not require a response.

initial program load (IPL). The process of loading the system programs and preparing the system to run jobs. See *initialize*.

initialize. To set counters, switches, addresses, or contents of storage

¦ Copyright IBM Corp. 1985, 1991

to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

inline input. See *here document*.

inode. The internal structure for managing files in the system. Inodes contain all of the information pertaining to the node, type, owner, and location of a file. A table of inodes is stored near the beginning of a file system.

i-number. A number specifying a particular inode on a file system.

inodetab. An AIX kernel parameter that establishes a table in memory for storing copies of inodes for all active files.

input device. Physical devices used to provide data to a computer.

input file. A file opened by a program so that the program can read from that file.

input list. A list of variables to which values are assigned from input data.

input redirection. The specification of an input source other than the standard one.

input-output file. A file opened for input and output use.

input-output device number. A value assigned to a device driver by the guest operating system or to the virtual device by the virtual resource manager. This number uniquely identifies the device regardless of whether it is real or virtual.

input/output (I/O). Pertaining to either input, output, or both between a computer and a device.

interactive processing. A processing method in which each system user's action causes response from the program or the system. Contrast with *batch processing*.

interface. A shared boundary between two or more entities. An interface might be a hardware component to link two devices together or it might be a portion of storage or registers accessed by two or more computer programs.

interleave factor. Specification of the ratio between contiguous physical

¦ Copyright IBM Corp. 1985, 1991

blocks (on a fixed disk) and logically contiguous blocks (as in a file).

interrupt. (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission. (3) A signal sent by an I/O device to the processor when an error has occurred or when assistance is needed to complete I/O. An interrupt usually suspends execution of the currently executing program.

INTERRUPT. The user-definable character used to kill a process running in the foreground. By default, **INTERRUPT** is the **Del** key. For more information about user-definable keys, see the **termio** file format entry in *AIX Operating System Technical Reference*.

IPL. See *initial program load*.

ISO. International Standards Organization.

job. (1) A unit of work to be done by a system. (2) One or more related procedures or programs grouped into a procedure.

job control. A feature that lets the system accept your commands to stop and start processes (jobs) and move them between background and foreground. The commands **ps** and **jobs** report the status of jobs (each of which is assigned a Process Identification Number, or PID, used to show its process status) and the **kill** command can be used to stop them.

job number. A number assigned to a background process when it is started. The job number is displayed when the process is started and when the **jobs** command is invoked; it can also be used to kill the process.

job queue. A list, on disk, of jobs waiting to be processed by the system.

justify. To print a document with even right and left margins.

kbuffers. An AIX kernel parameter establishing the number of buffers that can be used by the kernel.

K-byte. See *kilobyte*.

kernel. Part of the AIX Operating System. The kernel supervises the input and output, manages and controls the hardware, and schedules the user processes for execution. The memory-resident nucleus of the AIX Operating System containing functions needed immediately and frequently.

kernel parameters. Variables that specify how the kernel allocates

¦ Copyright IBM Corp. 1985, 1991

certain system resources.

keyboard. An input device consisting of various keys allowing the user to input data, control cursor and pointer locations, and to control the dialog between the user and the display station.

keylock feature. A security feature in which a lock and key can be used to restrict the use of the display station.

keyword. One of the predefined words of a programming language; a reserved word.

keyword argument. One type of variable assignment that can be made on the command line.

kill. An AIX Operating System command that stops a process.

kill character. The character that is used to delete a line of characters entered after the user's prompt.

kilobyte. 1024 bytes.

kprocs. An AIX kernel parameter establishing the maximum number of processes that the kernel can run simultaneously.

label. (1) The name in the disk or diskette volume table of contents that identifies a file. See also *file name*. (2) The field of an instruction that assigns a symbolic name to the location at which the instruction begins, or such a symbolic name.

LAN. See Local Area Network.

left margin. The area on a page between the left paper edge and the leftmost character position on the page.

left-adjust. The process of aligning lines of text at the left margin or at a tab setting such that the leftmost character in the line or filed is in the leftmost position. Contrast with *right-adjust*.

library. A collection of functions, calls, subroutines, or other data.

licensed program product (LPP). Software programs that remain the property of the manufacturer, for which customers pay a license fee.

line editor. An editor that modifies the contents of a file one line at a

¦ Copyright IBM Corp. 1985, 1991

time.

linefeed. An ASCII character that causes an output device to move forward one line.

link. A connection between an inode and one or more file names associated with it. Also, referred to as a UNIX link or hard link.

literal. A symbol or a quantity in a source program that is itself data, rather than a reference to data.

load. (1) To move data or programs into storage. (2) To place a diskette into a diskette drive, or a magazine into a diskette magazine drive. (3) To insert paper into a printer.

loader. A program that reads run files into main storage, thus preparing them for execution.

local. Pertaining to a device directly connected to your system without the use of a communications line. Contrast with *remote*.

Local Area Network (LAN). A data network located on the user's premises in which serial transmission is used for direct data communication among data stations.

local cluster site. The site on a cluster that the user is logged in to. The term "local" refers to a TCF cluster site.

<LOCAL> alias. The <LOCAL> alias can translate into different strings on different cluster sites for different processes. When <LOCAL> is the first component of the destination name for a symbolic link, it is replaced with its alias string, normally **/sitename**.

<LOCAL> file system. The part of the root file system hierarchy comprising system directories and files (such as the **/etc/motd** "message of the day" file) defined uniquely on a particular computer in the cluster. These files are not replicated. The name of the <LOCAL> file system appears in response to the **site -1** command.

location transparency. Allows an object to change location without the user's or program's knowledge if that location is not part of the object's name. For example, /u/joc/glossary may have been a file on cycorc last week, but it is a file on pooh this week. Joe does not need to know that the file was on either cyorc or pooh. If, however, Joe wants to find out where the site is located, he may invoke the where command.

locale. Each process operates in its own locale. This is a set of environment variables which the process uses to function in different ways

¦ Copyright IBM Corp. 1985, 1991

according to the demands of different cultural traditions. It includes factors like the language in use, the date-handling conventions and the system of monetary notation.

log. To record; for example, to log all messages on the system printer. A list of this type is called a log, such as an error log.

log in. To begin a session at a display station.

login ID. The ID set by the system for a user after login, but before running any programs.

log in shell. The program, or command interpreter, started for a user at log in.

log off. To end a session at a display station.

log out. To end a session at a display station.

logical device. A file for conducting input or output with a physical device.

loop. A sequence of instructions performed repeatedly until an ending condition is reached.

LPP. See licensed program product.

macro. A set of statements defining the name of, format of, and conditions for generating a sequence of assembler statements from a single source statement.

mailbox. An area designated for storage of mail messages directed to a specific system user.

main storage. The part of the processing unit where programs are run.

maintenance system. A special version of the AIX Operating System which is loaded from diskette and used to perform system management tasks.

major device number. A system identification number for each device or type of device.

mapped files. Files on the fixed disk that are accessed as if they are in memory.

¦ Copyright IBM Corp. 1985, 1991

mask. A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

matrix. An array arranged in rows and columns.

maxprocs. An AIX kernel parameter establishing the maximum number of processes that can be run simultaneously by a user.

MBCS. See *MultiByte Character Set*.

memory. Storage on electronic chips. Examples of memory are random access memory, read only memory, or registers. See *storage*.

menu. A displayed list of items from which an operator can make a selection.

message. (1) A response from the system to inform the operator of a condition which may affect further processing of a current program. (2) Information sent from one user in a multi-user operating system to another.

minidisk. A logical division of a fixed disk.

minor device number. A number used to specify various types of information about a particular device; for example, to distinguish among several printers of the same type.

mode word. An inode field that describes the type and state of the inode.

modem. See *modulator-demodulator*.

modulation. Changing the frequency or size of one signal by using the frequency or size of another signal.

modulator-demodulator (modem). A device that converts data from the computer to a signal that can be transmitted on a communications line, and converts the signal received to data for the computer.

module. (1) A discrete programming unit that usually performs a specific task or set of tasks. Modules are subroutines and calling programs that are assembled separately, then linked to make a complete program. (2) See *load module*.

mount. To make a file system accessible.

¦ Copyright IBM Corp. 1985, 1991

mountpoint. Any directory which has a file system mounted to it.

mountab. An AIX kernel parameter establishing the maximum number of file systems that can be mounted simultaneously.

MultiByte Character Set. A method of encoding a set of glyphs (written symbols) so that the coded set is large enough to encompass all written languages. The MBCS system handles character sets which range in encoding size from one to four bytes.

multiprogramming. The processing of two or more programs at the same time.

multi-user environment. A computer system that provides terminals and keyboards for more than one user at the same time.

multivolume file. A diskette file occupying more than one diskette.

MVS. Multiple Virtual Storage.

National Language support. National Language character support is provided for 8-bit ASCII. It provides National Language Character Set support for IBM console displays, selected World Trade ASCII terminals, and IBM printers with National Language capability.

nest. To incorporate a structure or structures of some kind into a structure of the same kind. For example, to nest one loop (the nested loop) within another loop (the nesting loop); to nest one subroutine (the nested subroutine) within another subroutine (the nesting subroutine).

network. A collection of computers that can communicate with each other. A network can consist of several interconnected computers or one computer with a number of remote terminals connected to it. Any of a variety of communication media can be used, such as RS232, Ethernet, PC Net, etc.

Network File System (NFS). A licensed program that allows users to share files between machines in a network environment by mounting the file systems located on remote machines.

Network Information Services (NIS). A network service that utilizes a centralized data base system to administer system information, such as passwords and machine names.

new-line character. A control character that causes the print or display position to move to the first position on the next line.

¦ Copyright IBM Corp. 1985, 1991

NFS. See *Network File System*.

NLS. See National Language Support.

node. An individual element of a full pathname. Nodes are separated by slashes (/).

null. Having no value, containing nothing.

null character (NUL). The character hex 00, used to represent the absence of a printed or displayed character.

object code. Machine-executable instruction, usually generated by a compiler from source code written in a higher level language. consists of directly executable machine code. For programs that must be linked, object code consists of relocatable machine code.

octal. A base eight numbering system.

open. To make a file available to a program for processing.

operating system. Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

operation. A specific action (such as move, add, multiply, load) that the computer performs when requested.

operator. A symbol representing an operation to be done.

output. The result of processing data.

output devices. Physical devices used by a computer to present data to a user.

output file. A file that is opened by a program so that the program can write to that file.

output redirection. The specification of an output destination other than the standard one.

override. (1) A parameter or value that replaces a previous parameter or value. (2) To replace a parameter or value.

¦ Copyright IBM Corp. 1985, 1991

overwrite. To write output into a storage or file space that is already occupied by data.

owner. The user who has the highest level of access authority to a data object or action, as defined by the object or action.

pad. To fill unused positions in a field with dummy data, usually zeros or blanks.

page. A block of instructions, data, or both.

page space minidisk. The area on a fixed disk that temporarily stores instructions or data currently being run. See also *minidisk*.

pagination. The process of adjusting text to fit within margins and/or page boundaries.

paging. The action of transferring instructions, data, or both between real storage and external page storage.

PANIC. An error message generated by the kernel indicating that an error has occurred which is sufficiently severe to prohibit kernel recovery.

parallel processing. The condition in which multiple tasks are being performed simultaneously within the same activity.

parameter. Information that the user supplies to a panel, command, or function.

parent. Pertaining to a secured resource, either a file or library, whose user list is shared with one or more other files or libraries. Contrast with *child*.

parent directory. The directory one level above the current directory.

partition. See *minidisk*.

password. A string of characters that, when entered along with a user identification, allows an operator to sign on to the system.

password security. A program product option that helps prevent the unauthorized use of a display station by checking the password entered by each operator at sign-on.

path name. The sequential list of directory name(s) that identify the

¦ Copyright IBM Corp. 1985, 1991

location of a particular directory and directory name(s) and file name that identify the location of a particular file, in the file hierarchy.

The path name is displayed in response to the **pwd** (print working directory) command. Each file has a full path name beginning with **/** (designating the root directory) and ending with the file's name. The file's relative path name does not begin with **/**.

pattern-matching character. Special characters such as * or ? that can be used in search patterns in a file specification to match one or more characters. For example, placing a ? in a file specification means any character can be in that position. Pattern-matching characters are also called wildcards.

permission code. A three-digit octal code, or a nine-letter alphabetic code, indicating the access permissions. The access permissions are read, write, and execute.

permission field. One of the three-character fields within the permissions column of a directory listing indicating the read, write, and run permissions for the file or directory owner, group, and all others.

phase. One of several stages file system checking and repair performed by the **fsck** command.

physical device. See *device*.

physical file. An indexed file containing data for which one or more alternative indexes have been created.

physical record. (1) A group of records recorded or processed as a unit. Same as *block*. (2) A unit of data moved into or out of the computer.

PID. See *process ID*.

pipe. To direct the data so that the output from one process becomes the input to another process.

pipeline. A direct, one-way connection between two or more processes.

pitch. A unit of width of typewriter type, based on the number of times a letter can be set in a linear inch. For example, 10-pitch type has 10 characters per inch.

platen. The support mechanism for paper on a printer, commonly cylindrical, against which printing mechanisms strike to produce an impression.

¦ Copyright IBM Corp. 1985, 1991

pointer. A logical connection between physical blocks.

port. (1) To make the programming changes necessary to allow a program that runs on one type of computer to run on another type of computer. (2) An access point for data input to or data output from a computer system. See *connector*.

portmap service (portmapper). A daemon process that matches RPC port numbers to RPC services provided by NFS servers to conduct remote services in the NFS.

position. The location of a character in a series, as in a record, a displayed message, or a computer printout.

positional parameter. A shell facility for assigning values from the command line to variables in a program.

POSIX. Portable Operating System for Computer Environments.

preprocessor. (1) A functional unit that effects preparatory computation or organization. (2) A program that examines the source program for preprocessor statements which are then executed, resulting in the alteration of the source program.

primary copy. Each replicated file system has a copy designated as the *primary copy*, which is the copy that may be modified. It resides on the *primary site* and its purpose is to guarantee that file updates are kept consistent.

primary site. The cluster site that maintains the primary copy of a replicated file system.

print queue. A file containing a list of the names of files waiting to be printed.

printout. Information from the computer produced by a printer.

priority. The relative ranking of items. For example, a job with high priority in the job queue will be run before one with medium or low priority.

priority number. A number that establishes the relative priority of printer requests.

privileged user. The account with superuser authority.

¦ Copyright IBM Corp. 1985, 1991

problem determination. The process of identifying why the system is not working. Often this process identifies programs, equipment, data communications facilities, or user errors as the source of the problem.

problem determination procedure. A prescribed sequence of steps aimed at recovery from, or circumvention of, problem conditions.

procedure. See *shell procedure*.

process. (1) A sequence of actions required to produce a desired result. (2) An entity receiving a portion of the processor's time for executing a program. (3) An activity within the system begun by entering a command, running a shell program, or being started by another process.

process accounting. An analysis of the use each process makes of the processing unit, memory, and I/O resources.

process ID (PID). A unique number assigned to a process that is running.

process transparency. The ability to execute and control tasks on any site in the cluster, regardless of where the user is logged in (to find out where that is, type the site command). The same system calls and commands are used, no matter where the process is located. For example, a remote job is stopped the same way that a local job is stopped.

profile. (1) A file containing customized settings for a system or user (2) Data describing the significant features of a user, program, or device.

program. A set of instructions for the computer to interpret and execute.

prompt. A displayed request for information or operator action.

propagation time. The time necessary for a signal to travel from one point on a communications line to another.

protocol. In data communication, the rules for transferring data.

protocol procedure. A process that implements a function for a device manager. For example, a virtual terminal manager may use a protocol procedure to interpret the meaning of keystrokes.

PR/SM. Process Resource/System Manager.

qdaemon. The daemon process that maintains a list of outstanding jobs and

¦ Copyright IBM Corp. 1985, 1991

sends them to the specified device at the appropriate time.

queue. A line or list formed by items waiting to be processed.

queued message. A message from the system that is added to a list of messages stored in a file for viewing by the user at a later time. This is in contrast to a message that is sent directly to the screen for the user to see immediately.

quit. A key, command, or action that tells the system to return to a previous state or stop a process.

quote. To mask the special meaning of certain characters; to cause them to be taken literally.

random access. An access mode in which records can be read from, written to, or removed from a file in any order.

read only. Pertaining to file system mounting, a condition that allows data to be read, but not modified.

real ID. The ID (user or group) that is set at login.

recovery procedure. (1) An action performed by the operator when an error message appears on the display screen. Usually, this action permits the program to continue or permits the operator to run the next job. (2) The method of returning the system to the point where a major system error occurred and running the recent critical jobs again.

redirect. To divert data from a process to a file or device to which it would not normally go.

reference count. In an inode, a record of the total number of directory entries that refer to the inode.

relational expression. A logical statement describing the relationship (such as greater than or equal) of two arithmetic expressions or data items.

relational operator. The reserved words or symbols used to express a relational condition or a relational expression.

relative address. An address specified relative to the address of a symbol. When a program is relocated, the addresses themselves will change, but the specification of relative addresses remains the same.

¦ Copyright IBM Corp. 1985, 1991

relative addressing. A means of addressing instructions and data areas by designating their locations relative to some symbol.

relative path name. The name of a directory or file expressed as a sequence of directories followed by a file name, beginning from the current directory.

remote. Pertaining to a system or device that is connected to your system through a communications line. Contrast with *local*.

remote cluster site. A site on the cluster that the user is not logged in to. The term "remote" usually refers to a TCF cluster site.

remote procedure call. A request for a service located on another computer in the network.

Remote Procedure Call. The interface NFS uses for remote procedure calls.

replicated root file system. The part of the root file system hierarchy comprising system directories and files found under the root (**/**) and replicated on all sites in the cluster. Replicated root files are not specific to individual cluster sites.

reserved character. A character or symbol that has a special (non-literal) meaning unless quoted.

reserved word. A word that is defined in a programming language for a special purpose and that must not appear as a user-declared identifier.

reset. To return a device or circuit to a clear state.

restore. To return to an original value or image. For example, to restore a library from diskette.

right adjust. The process of aligning lines of text at the right margin or tab setting such that the rightmost character in the line or file is in the rightmost position.

right justify. See right align.

right margin. The area on a page between the last text character and the right upper edge.

right-adjust. To place or move an entry in a field so that the rightmost character of the field is in the rightmost position. Contrast with *left-adjust*.

¦ Copyright IBM Corp. 1985, 1991

root. Another name sometimes used for superuser.

root directory. The top level of a tree-structured directory system.

root file system. The basic AIX Operating System file system, which contains operating system files and onto which other file systems can be mounted. The root file system is the file system that contains the files that are run to start the system running.

routine. A set of statements in a program causing the system to perform an operation or a series of related operations.

RPC. See Remote Procedure Call.

RSCS. (Remote Spooling Communications System) - A VM program that transfers files, usually between real machines. Each file is taken from the spooling system of a particular virtual machine and can then be transferred to the spooling system of any operation system that contains the required protocols for communicating with RSCS.

run. To cause a program, utility, or other machine function to be performed.

run-time environment. A collection of subroutines and shell variables that provide commonly used functions and information for system components.

SCCS. Source Code Control System.

scratch file. A file, usually used as a work file, that exists until the program that uses it ends.

screen. See *display screen*.

scroll. To move information vertically or horizontally to bring into view information that is outside the display screen boundaries.

second level interrupt handler (SLIH). A routine that handles the processing of an interrupt from a specific adapter. AN SLIH is called by the first level interrupt handler associated with that interrupt level.

secondary copy. A read-only copy of the primary copy of a replicated file system. Files in the secondary copy are automatically modified or deleted when the corresponding file in the primary copy is modified or deleted. New files added to the primary copy will be automatically added to the

¦ Copyright IBM Corp. 1985, 1991

secondary copy only if the appropriately **fstore** value has been set.

sector. (1) An area on a disk track or a diskette track reserved to record information. (2) The smallest amount of information that can be written to or read from a disk or diskette during a single read or write operation.

security. The protection of data, system operations, and devices from accidental or intentional ruin, damage, or exposure.

segment. A contiguous area of virtual storage allocated to a job or system task. A program segment can be run by itself, even if the whole program is not in main storage.

separator. A character used to separate parts of a command or file.

server. A computer or process that provides data, services, or resources that can be accessed by other computers or processes in the network.

sequential access. An access method in which records are read from, written to, or removed from a file based on the logical order of the records in the file.

session records. In the accounting system, a record of time connected and line usage for connected display stations, produced from log in and log out records.

set flags. Flags that can be put into effect with the shell set command.

shadow page table. A table that maps real storage allocations (first-level storage) to a virtual machine's virtual storage (third-level storage) for use by the real machine in its paging options.

shared printer. A printer that is used by more than one workstation.

shell. See *shell program*.

shell options. The shell provides two different types of options. *Set options* are put into effect with the **set** command and alter the way the shell runs. *Command line options* are entered on the command line (but not with the **set** command) and alter the way the shell starts.

shell procedure. A series of commands combined in a file that carry out a particular function when the file is run or when the file is specified as an argument to the sh command. Shell procedures are frequently called shell scripts.

¦ Copyright IBM Corp. 1985, 1991

shell program. A program that accepts and interprets commands for the operating system. (There is an AIX shell program and a DOS shell program.)

shell prompt. The character string on the command line indicating that the system can accept a command (typically the **$** character).

shell script. See *shell* procedure.

shell variables. Facilities of the shell program for assigning variable values to constant names.

size field. In an inode, a field that indicates the size, in bytes, of the file associated with the inode.

SNA. See Systems Network Architecture.

sort. To rearrange some or all of a group of items based upon the contents or characteristics of those items.

source code. The input to a compiler or assembler, written in a source language.

source diskette. The diskette containing data to be copied, compared, restored, or backed up.

source program. A set of instructions written in a programming language that must be translated to machine language compiled before the program can be run.

special character. A character other than an alphabetic or numeric character that has special meaning to the shell, such as * and ?.

special file. Special files are used in the AIX system to provide an interface to input/output devices. There is at least one special file for each device connected to the computer. Contrast with *directory* and *file*. See also *block special file* and *character special file*.

spool. A VM program that allocates disk areas to hold files in queues, usually while they await some sort of service. Print jobs, for example, are held in a print queue awaiting service from the printer.

spool files. Files used in the transmission of data among devices.

SRM. System Resource Manager.

¦ Copyright IBM Corp. 1985, 1991

standalone shell. A limited version of the shell program used for system maintenance.

standalone workstation. A workstation that can be used to preform tasks independent of (without being connected to) other resources such as servers or host systems.

standard error. The place where many programs place error messages.

standard input. The primary source of data going into a command. Standard input comes from the keyboard unless redirection or piping is used, in which case standard input can be from a file or the output from another command.

standard output. The primary destination of data coming from a command. Standard output goes to the display unless redirection or piping is used, in which case standard output can be to a file or another command.

stanza. A group of lines in a file that together have a common function. Stanzas are usually separated by blank lines, and each stanza has a name.

statement. An instruction in a program or procedure.

status. (1) The current condition or state of a program or device; for example, the status of a printer. (2) The condition of the hardware or software, usually represented in a status code.

storage. (1) The location of saved information. (2) In contrast to memory, the saving of information on physical devices such as disk or tape. See *memory*.

storage device. A device for storing and/or retrieving data.

string. A linear sequence of entities such as characters or physical elements. Examples of strings are alphabetic string, binary element string, bit string, character string, search string, and symbol string.

su. See *superuser*.

subdirectory. A directory contained within another directory in the file system hierarchy.

subprogram. A program invoked by another program, such as a subshell.

¦ Copyright IBM Corp. 1985, 1991

subroutine. (1) A sequenced set of statements that may be used in one or more computer programs and at one or more points in a computer program. (2) A routine that can be part of another routine.

subscript. An integer or variable whose value refers to a particular element in a table or an array.

subshell. An instance of the shell program started from an existing shell program.

substring. A part of a character string.

subsystem. A secondary or subordinate system, usually capable of operating independently of, or synchronously with, a controlling system.

superblock. The most critical part of the file system containing information about every allocation or deallocation of a block in the file system.

superuser (su). The user who can operate without the restrictions designed to prevent data loss or damage to the system (User ID 0).

superuser authority. The unrestricted ability to access and modify any part of the operating system associated with the user who manages the system. The authority obtained when one logs in as **root**.

supervisor. The part of the AIX/370 Operating System control program that coordinates the use of resources and maintains the flow of processing unit operations.

symbolic link. A mechanism that lets you assign a second name to a file or a directory. There are no restrictions pertaining to source or destination location. A file may be deleted if the only pointers to the file are symbolic links.

system. The computer and its associated devices and programs.

system administrator. The person at a computer installation who designs, controls, and manages the use of the computer system.

system call. A request by an active process for a service by the system kernel.

system customization. A process of specifying the devices, programs, and users for a particular data processing system.

¦ Copyright IBM Corp. 1985, 1991

system date. The date assigned by the system user during setup and maintained by the system.

system dump. A copy of memory from all active programs (and their associated data) whenever an error stops the system. Contrast with *task dump*.

system management. The tasks involved in maintaining the system in good working order and modifying the system to meet changing requirements.

system parameters. See *kernel parameters*.

system primary site. The machine (cluster site) designated to hold the primary copy of the replicated root file system. When files are changed in the replicated root file system, the primary site for the cluster must be available.

system profile. A file containing the default values used in system operations.

system-replicated file system. One that contains files and directories accessed by many users regardless of the users' specific applications. These system files, programs and directories are replicated on different sites in a cluster.

system unit. The part of the system that contains the processing unit, the disk drives, and the diskette drives.

systems network architecture (SNA). A set of rules for controlling the transfer of information in a data communication network.

target diskette. The diskette to be used to receive data from a source diskette.

task. A basic unit of work to be performed. Examples are a user task, a server task, and a processor task.

task dump. A copy of memory from a program that failed (and its associated data). Contrast with *system dump*.

TCF. See *Transparent Computing Facility*.

TCP/IP. See *Transmission Control Protocol/Internet Protocol*.

terminal. An input/output device containing a keyboard and either a display device or a printer. Terminals usually are connected to a

¦ Copyright IBM Corp. 1985, 1991

computer and allow a person to interact with the computer.

text application. A program defined for the purpose of processing text data (for example, memos, reports, and letters).

text editing program. See *editor* and *text application*.

texttab. A kernel parameter establishing the size of the text table, in memory, that contains one entry each active shared program text segment.

Token-Ring network. A network that uses a ring topology, in which tokens are passed in the circuit from node to node. A node ready to send can capture the token and insert data for transmission.

trace. To record data that provides a history of events occurring in the system.

trace table. A storage area into which a record of the performance of computer program instructions is stored.

track. A circular path on the surface of a fixed disk or diskette on which information is magnetically recorded and from which recorded information is read.

Transmission Control Protocol/Internet Protocol. A network protocol that connects to other systems that support TCP/IP. TCP/IP allows a user to send and receive mail and transfer files across a network, print files and run commands on remote systems, and log in to remote systems.

Transparent Computing Facility (TCF). An operating system that automatically allows for data, process, name, location, and semantic transparency. Process transparency is the ability to execute and control tasks on any cluster site no matter where the user program is currently executing.

trap. An unprogrammed, hardware-initiated jump to a specific address. Occurs as a result of an error or certain other conditions.

tree-structured directories. A method for connecting directories such that each directory is listed in another directory except for the root directory, which is at the top of the tree.

Trojan horse. A program that can vandalize your files, although it does its defined task.

truncate. To shorten a field or statement to a specified length.

¦ Copyright IBM Corp. 1985, 1991

trusted communications path. A secure path to the system, invoked with a key sequence and used when entering or changing security-relevant information in the system. For example, when changing passwords or logging in to the system.

trusted computing base. The total of all system components, both hardware and software, that protect data in the system.

trusted program. A program with an ID of root (superuser) known to be free of Trojan horses and computer viruses.

trusted shell. A modified command interpreter that provides a restricted environment to perform administrative tasks in a secure manner.

typematic key. A key that repeats its function multiple times when held down.

typestyle. Characters of a given size, style and design.

TTY. Designates a terminal. On a system with more than one terminal, the TTY field of the process status displayed by the **ps** command indicates which terminal started the process.

Uid. See *user number*.

UNIX link. Or a hard link, is a mechanism that lets you use the **ln** command to assign more than one name to a file. Both the new name and the file being linked to must be in the same file system. A file is deleted when all the UNIX links (including the first link - the original name) to it have been removed.

Unix to Unix Copy (UUCP). A set of programs that allows you to copy files from a local unix system to a remote unix system. See Basic Networking Utility (BNU).

update. An improvement for some part of the system.

user. The name associated with an account.

user account. See *account*.

user ID. See *user number*.

user list. A list, containing the user identification and access levels, of all operators who are allowed to use a specified file or library.

¦ Copyright IBM Corp. 1985, 1991

user name. A name that uniquely identifies a user to the system.

user number (Uid). (1) A unique number identifying an operator to the system. This string of characters limits the functions and information the operator is allowed to use. The Uid can often be substituted in commands that take a user's name as an argument.

user profile. A file containing a description of user characteristics and defaults (for example, printer assignment, formats, group ID) to be conveyed to the system while the user is signed on.

utility. A service; in programming, a program that performs a common service function.

UUCP. See Unix-to-Unix Copy.

valid. (1) Allowed. (2) True, in conforming to an appropriate standard or authority.

value. (1) In Usability Services, information selected or typed into a pop-up. (2) A set of characters or a quantity associated with a parameter or name. (3) In programming, the contents of a storage location.

variable. A name used to represent a data item whose value can change while the program is running. Contrast with *constant*.

VCTC. Virtual Channel-to-Channel.

verify. To confirm the correctness of something.

version. Information in addition to an object's name that identifies different modification levels of the same logical object.

vi. A full screen editor.

virtual device. A device that appears to the user as a separate entity but is actually a shared portion of a real device. For example, several virtual terminals may exist simultaneously, but only one is active at any given time.

virtual machine. (1) A functional simulation of a computer and its associated devices. Each virtual machine is controlled by a suitable operating system (for example, conversational monitor system). VM/370 controls concurrent execution of multiple virtual machines on a single System/370. (2) In VM, a functional equivalent of either a System/370

¦ Copyright IBM Corp. 1985, 1991

computing system or a System/370-Extended Architecture computing system. Each virtual machine is controlled by an operating system. VM controls concurrent execution of multiple virtual machines on a single system.

virtual machine console. A terminal used to control a virtual machine. There is one VM console for every virtual machine running on the real machine. Each simulates the console of a real machine.

virtual storage. Addressable space that appears to be real storage. From virtual storage, instructions and data are mapped into real storage locations.

virtual terminal. Any of several logical equivalents of a display station available at a single physical display station.

VM. A hypervisor (a program that runs operating systems) that runs on System/370 hardware. VM can divide the processor into any number of virtual machines, each of which appears to be a complete System/370 machine.

VM HPO. Virtual Machine High Performance Option.

VM minidisk. VM has the ability to divide a real disk into smaller minidisks, each of which acts like a whole real disk. The smallest size that may be allocated is one cylinder for CKD devices or one block for FBA devices.

VM HPO PMA. Virtual Machine High Performance Option Preferred Machine Assist.

VM/SP. Virtual Machine/System Product.

VM/XA. Virtual Machine/Extended Architecture.

Volume ID (Vol ID). A series of characters recorded on the diskette used to identify the diskette to the user and to the system.

wildcard. See *pattern-matching characters*.

word. A contiguous series of 32 bits (4 bytes) in storage, addressable as a unit. The address of the first byte of a word is evenly divisible by four.

work file. A file used for temporary storage of data being processed.

workstation. A device at which an individual may transmit information to,

¦ Copyright IBM Corp. 1985, 1991

or receive information from, a computer for the purpose of performing a task; for example, a display station or printer. See *programmable workstation* and *dependent workstation*.

working directory. See *current directory*.

wrap around. Movement of the point of reference in a file from the end of one line to the beginning of the next, or from one end of a file to the other.

X.25. A commercial packet network access protocol that specifies three levels of connections. The X.25 physical level, link level, and packet level correspond to the first three layers of the ISO/OSI model.

XDR. A data definition language used as a standard to the RPC routines for remote communications. The internal data representations of various machine types are represented in a uniform format so that networked machines can communicate regardless of their manufacturer or structure algorithm.

¦ Copyright IBM Corp. 1985, 1991

Special Characters

A

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

B

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

C

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

D

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

E

¦ Copyright IBM Corp. 1985, 1991

F

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

G

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

N

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

Q

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

U

V

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991

¦ Copyright IBM Corp. 1985, 1991