Bull

AIX 5L System User’s Guide
Communications and Networks

AIX

ORDER REFERENCE
86 A2 45EM 01

Bull

AIX 5L System User's Guide Communications and Networks

AIX

Software

October 2005

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 45EM 01

Printed in France

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

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

Trademarks and Acknowledgements

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

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

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

Linux is a registered trademark of Linus Torvalds.

Contents

About This Book

This book provides end users with complete information about how to perform such tasks as using communications applications and services for the operating system. This book describes the following communications applications: Mail, Message Handler (MH), Transmission Control Protocol/Internet Protocol (TCP/IP), Basic Networking Utilities (BNU), and Asynchronous Terminal Emulation (ATE). It provides overviews, concepts, and procedures on receiving and sending mail and messages , transferring files (ftp command), printing files from and to a remote system, running commands on other systems, communicating between local and remote systems, and customizing the communications environment. This publication is also available on the documentation CD that is shipped with the operating system.

Highlighting

The following highlighting conventions are used in this book:

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

Case-sensitivity in AIX

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

ISO 9000

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

Related publications

The following books contain related information:

- *AIX 5L Version 5.3 System User's Guide: Operating System and Devices*
- *AIX 5L Version 5.3 System Management Guide: Communications and Networks*

Chapter 1. Introduction to Communication Concepts

This chapter provides a general overview of communication concepts. In addition, several system commands are mentioned to help you determine the current communication environment. Topics discussed include:

Network applications

A *network* is the combination of two or more computers and the connecting links between them. A *physical network* is the hardware (equipment such as computers, cables, and telephone lines) that makes up the network. The software and other controlling devices and conventions make up the *logical network*.

Different types of networks and emulators provide different functions. To assist you in understanding the basic concepts of networking and emulation, this section consists of the following information:

System hardware and software communication support for users

All network communications involve use of hardware and software. *Hardware* consists of the physical equipment connected to the physical network. *Software* consists of the programs, procedures, rules or protocols, and associated documentation pertaining to the operation of a particular system. The system hardware and software communications support is determined by the hardware being used and the software necessary to run that hardware and interface with the network.

The system hardware consists of adapter cards that provide a path or interface between the system software and the physical network. An adapter card requires an input/output (I/O) card slot in the system. The adapter card connects the *data terminal equipment* (DTE) to the *data circuit-terminating equipment* (DCE); that is, it provides physical local addressing to a DTE port.

An adapter card prepares all inbound and outbound data; performs address searches; provides drivers, receivers, and surge protection; supports different interfaces; and in general relieves the system processor of many communications tasks. Adapter cards support the standards required by the physical network (for example, EIA 232D, Smartmodem, V.25 bis, EIA 422A, X.21, or V.35) and at the same time support software protocols (for example, synchronous data link control or SDLC, high-level data link control or HDLC, and bisynchronous protocols).

User and application communications functions

Networks allow for several user and application communications functions. They enable a user to:

- Send electronic mail
- Emulate another terminal or log in to another computer
- Transfer data
- Execute programs residing on a remote node.

The communications network enables one user to communicate with another user through electronic mail. The underlying layers of software and hardware, as well as the physical network, allow a user to generate, send, receive, and process messages, letters, memos, invitations, and data files. These communications

can be to or from any other user who resides on the physical network. Electronic mail has the capability for message annotation, message sequencing, message packing, date sorting, and mail folder management.

Emulating another computer permits users to access applications in other systems as if directly attached to that system. Remote login provides an interactive command line interface between a system based on this operating system and other UNIX-based systems.

A file transfer protocol allows users to access files and directories on remote hosts. Files may be transferred to and from the remote host. Password protection is usually provided as part of the protocol. With a file transfer, there is a client/server relationship between the user initiating the request and the remote system the user is accessing. Often a file transfer protocol includes functions for display and control so that users with read/write access can display, define, or delete files and directories.

Several different protocols have been devised to allow users and applications on one system to invoke procedures and execute applications on other systems. This can be useful for a number of environments, including the offloading of many computer-intensive routines in engineering and scientific applications.

Communicating with other operating systems

Different types of computers can be connected on a network. That is, the computers can be from different manufacturers or be different models from the same manufacturer. The differences in operating systems of two or more types of computers can be bridged with programs.

Sometimes these programs require that another program be previously installed on the network. Other programs may require that such communications connectivity protocols as Transmission Control Protocol/Internet Protocol (TCP/IP) or Systems Network Architecture (SNA) exist on the network.

Mail applications

Before you can use the mail system, you must select a user-agent program, such as one of the following:

- “Mail program (mail)”
- “Message handler (mh)” on page 3
- “bellmail command” on page 4

A user-agent program provides facilities for creating, receiving, sending, and filing mail. In addition, you need a transport-agent program, **sendmail**, which distributes incoming mail from other systems or packages and distributes each outgoing mail item and then transmits it to a similar program in one or more remote systems.

> **Note: mail** and **mh** are incompatible in the way they store mail; you must choose one mail handler or the other.

Mail program (mail)

The **mail** program provides you with a user interface to handle mail to and from both a local network user and a remote system user.

A mail message can be text, entered using an editor, or an ASCII file. In addition to a typed message or a file, you can send:

system message	Informs users the system has been updated. A system message is similar to a broadcast message but is sent on the local network only.
secret mail	Used to send classified information. A secret mail message is encrypted. The recipient must enter a password to read it.

vacation message	Informs users you are on vacation. When your system receives mail in your absence, it sends a message back to the origin. The message states you are on vacation. Any mail you receive while on vacation can also be forwarded.

When you receive mail using the **mail** subcommands, you can:

- Leave the mail in the system mailbox.
- Read and delete the mail.
- Forward the mail.
- Add comments to the mail.
- Store the mail in your personal mailbox (mbox).
- Store the mail in a folder you have created.
- Create and maintain an alias file or a distribution file, which directs the mail and mail messages.

The installation of **sendmail** is automatic.

For more information about the **mail** program, refer to Chapter 2, “Mail Overview,” on page 9.

Message handler (mh)

mh is a collection of commands that enables you to perform each mail processing function directly from the command line. These commands provide a broader range of function than the subcommands of **mail**. Also, because the commands can be issued at any time the command prompt is displayed, you gain power and flexibility in creating mail and in processing received mail. For example, you can read a mail message, search a file or run a program to find a particular solution, and answer the message, all within the same shell.

The **mh** program enables you to create, distribute, receive, view, process, and store messages using the following commands:

ali	Lists mail aliases and their addresses.
anno	Annotates messages.
ap	Parses and reformats addresses.
burst	Explodes digests into messages.
comp	Starts an editor for creating or modifying a message.
dist	Redistributes a message to additional addresses.
dp	Parses and reformats dates.
folder	Selects and lists folders and messages.
folders	Lists all folders and messages in the mail directory.
forw	Forwards messages.
inc	Incorporates new mail into a folder.
mark	Creates, modifies, and displays message sequences.
mhl	Produces a formatted listing of messages.
mhmail	Sends or receives mail.
mhpath	Prints full path names of messages and folders.
msgchk	Checks for messages.
msh	Creates a mail handler (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	Moves files between folders.
repl	Replies to a message.
rmf	Removes folders and the messages they contain.
rmm	Removes messages from active status.

scan	Produces a one-line-per-message scannable listing.
send	Sends a message.
show	Shows messages.
sortm	Sorts messages.
vmh	Starts a visual interface for use with **mh** commands.
whatnow	Starts a prompting interface for draft disposition.
whom	Manipulates mh addresses.

For more information on **mh** commands, refer to the *AIX 5L Version 5.3 Commands Reference, Volume 3*.

bellmail command

bellmail is the original AT&T UNIX® mail command, which handles mail for users on the same system and also for users on remote systems that can be accessed by means of Basic Network Utilities (BNU), sometimes known as the UNIX-to-UNIX Copy Program (UUCP). These programs support only networks of systems connected by dial-up or leased point-to-point communication lines. The command opens a shell whose subcommands allow you to:

- Take data from standard input (typed in or redirected from an existing file), add one or more addresses (supplied as arguments to the command itself) and a timestamp, then append a copy to each addressee's system mailbox file (**/var/spool/mail**/*UserID*).
- Read mail items from your system mailbox file.
- Append mail items to your personal mailbox file (**$HOME/mbox**) or to a specified file.
- Send mail using BNU to a user on another system.
- Automatically redirect all mail from your system mailbox to one on another system by adding a *.forward* statement to the beginning of your system mailbox file.

However, you must have some skill as a UNIX user before you can make full use of this mail handler. For more information, refer to the **bellmail** command in the *AIX 5L Version 5.3 Commands Reference, Volume 1*.

Host emulation applications

An *emulator* is a software application that allows your system to function as if you were using a different terminal or printer.

A *terminal emulator* connects to a host system to access data or applications. Some terminal emulators provide a facility to transfer files to and from the host. Others provide an application programming interface (API) to allow program-to-program communication and automation of host tasks.

A *printer emulator* allows the host either to print files on a local printer or store them in printable form to be printed or edited later.

Several applications are available to allow your system to emulate other types of terminals. The following sections provide information on terminal or printer emulators:

- "TCP/IP commands for emulation"
- "BNU commands for emulation" on page 5
- "Asynchronous terminal emulation (ATE)" on page 5

Note: The **bterm** command emulates terminals in bidirectional (BIDI) mode.

TCP/IP commands for emulation

The Transmission Control Protocol/Internet Protocol (TCP/IP) software includes the **telnet** and **rlogin** commands, which allow you to connect to and access a remote TCP/IP system.

telnet	Allows a user to log in to a remote host by implementing the **TELNET** protocol. It is different from the **rlogin** command in that it is a trusted command. A *trusted* command is one that meets all security levels configured on your computer. Systems that require extra security should allow only trusted commands. Standards for trusted commands, processes, and programs are set and maintained by the U.S. Department of Defense.
tn	Performs the same function as the **telnet** command.
rlogin	Allows a user to log in to a remote host. It is different from the **telnet** command in that it is a *nontrusted* command and can be disabled if your system needs extra security.

For more information about TCP/IP, see Chapter 3, “Transmission Control Protocol/Internet Protocol (TCP/IP) Overview,” on page 41.

BNU commands for emulation

The Basic Networking Utilities (BNU) software includes the **ct**, **cu**, and **tip** commands, which allow you to connect to a remote system that uses the AIX operating system.

ct	Enables a user on a remote terminal, such as a 3161, to communicate with another terminal over a telephone line. The user on the remote terminal can then log in and work on the other terminal. The **ct** command is similar to the **cu** command but not as flexible. For example, you cannot issue commands on the local system while connected to a remote system through the **ct** command. However, you can instruct the **ct** command to continue dialing until the connection is established or to specify more than one telephone number at a time.
cu	Connects your terminal to another terminal connected to either a UNIX or non-UNIX system. After the connection is established, you can be logged in on both systems at the same time, executing commands on either one without dropping the BNU communication link. If the remote terminal is also running under UNIX, you can transfer ASCII files between the two systems. You can also use the **cu** command to connect multiple systems, and commands can then be executed on any of the connected systems.
tip	Connects your terminal to a remote terminal and enables you to work on the remote terminal as if logged in directly. You can use the **tip** command to transfer files to and from the remote system. You can use scripting to record the conversations you have with the **tip** command. **Note:** You must have a login on the remote system to use the **tip** command.

For more information about BNU, see Chapter 4, “Basic Networking Utilities (BNU) Overview,” on page 57.

Asynchronous terminal emulation (ATE)

The Asynchronous Terminal Emulation (ATE) program enables your terminal to connect to most systems that support asynchronous terminals, including any system that supports RS-232C or RS-422A connections. ATE allows the remote system to communicate with your terminal either as an asynchronous display or as a DEC VT100 terminal.

ATE enables you to run commands on the remote system, send and receive files, and check the data integrity in the files transferred between systems. You can also use a capture file to record, or *capture*, incoming data from the remote system. ATE is menu-driven and uses subcommands.

When installed, ATE is accessible only to users who have been registered as a member of the UUCP group by a user with root authority.

For more information about ATE, see “ATE concepts” on page 67.

Communications system commands

This section describes commands available for displaying information that identifies users on your system, the system you are using, and users logged in to other systems.

Displaying your login name

Use the **whoami** command to determine your login name.

whoami Displays the login name being used, similar to the following:

```
denise
```

In this example, the login name is `denise`.

Displaying your system name

Use the **uname** command to determine your system name.

uname -n The **uname** command used with the **-n** flag displays the name of your system if you are on a network. Information similar to the following is displayed:

```
barnard
```

In this example, the system name is `barnard`.

To find the node name of another system, you must request that a user on that system enter the **uname -n** command.

Determining whether your system has access

Use the **host** to determine whether your system has access to information that defines the other system. To access another system on the network, your local system must have access to information that defines the other system. To determine if your local system has this information, enter the **host** command with the name of the other system, as in the following example:

host `zeus` Determines if your system has routing information for system `zeus`.

If your system responds with a message similar to:

```
zeus is 192.9.200.4 (300,11,310,4)
```

then your system has the proper information and you can send a message to system `zeus`. The address `192.9.200.4` is used by the system to route the mail.

If your system does not have information about the requested system, it responds with the following message:

```
zeus: unknown host
```

If you receive an `unknown host` message, then the requested system name:

- Is not correct (check the spelling in the address)
- Is on your network but not defined to your system (contact the person responsible for setting up your network)
- Is on another network (see “Addressing mail to users on a different network” on page 18) and requires more detailed addressing
- Is not connected to your network

You can also receive the `unknown host` message if your network is not operating and your local system depends on a remote system to supply network addresses.

Displaying information about logged-in users

Use the **finger** or **f** command to display information about the current users on a specified host. This information can include the user's login name, full name, and terminal name, as well as the date and time of login.

finger `@alcatraz`

Displays the following information about all users logged in to host `alcatraz`:

```
brown  Console Mar 15 13:19
smith  pts0    Mar 15 13:01
jones  tty0    Mar 15 13:01
```

User `brown` is logged in at the console, user `smith` is logged in from a pseudo teletype line pts0, and user `jones` is logged in from a tty0.

finger `brown` or `brown@alcatraz`

Displays the following information about the user `brown`:

```
Login name: brown
In real life: Marta Brown
Directory:/home/brown  Shell: /bin/ksh
On since May 8 07:13:49 on console
No Plan.
```

Chapter 2. Mail Overview

The **mail** program enables you to receive, create, and send mail to users on a local or remote system. Topics discussed in this section are:

Storing mail

Mail is stored in different ways, depending on the specific situation. When mail is sent to your address, it is stored in a system directory that is specifically for mail. This system directory contains a file for every user on the local system. This directory holds your mail until you do something with it.

The **mail** program uses the following types of mailboxes or folders:

System mailbox

The system mailbox is similar to a post office box: the post office delivers letters addressed to the person who owns that box. Similarly, the system mailbox is a file where messages are delivered to a particular user. If the file does not exist when mail arrives, it is created. The file is deleted when all messages have been removed.

System mailboxes reside in the **/var/spool/mail** directory. Each system mailbox is named by the user ID associated with it. For example, if your user ID is `karen`, your system mailbox is:

```
/var/spool/mail/karen
```

Default personal mailbox

Your personal mailbox is similar to an in-basket in an office. You put mail in the in-basket after you have received it, but before you have filed it.

Each user has a personal mailbox. When you read mail from the system mailbox, and if it is not marked for deletion or saved to a file, it is written to your personal mailbox, **$HOME/mbox** (**$HOME** is your login directory). The **mbox** file exists only when it contains a message.

dead.letter file for incomplete messages

If you need to interrupt a message you are creating to complete other tasks, the system saves incomplete messages in the **dead.letter** file in the **$HOME** directory. If the **dead.letter** file does not exist, the file is created. Later you can edit the file to complete your message.

> **Attention:** Do not use the **dead.letter** file to store messages. The content of this file is overwritten each time an interrupt is issued to save a partial message to the **dead.letter** file.

Mail folders

Folders enable you to save messages in an organized fashion. Using the mail program, you can put a message into a folder from the system mailbox, a personal mailbox, or another folder.

Each folder is a text file. Each folder is placed in the directory you specify in your **.mailrc** file with the **set folder** option. You must create this directory before using folders to store messages. When the directory exists, the mail program creates the folders in that directory as needed. If you do not specify a directory in your **.mailrc** file, folders are created in the current directory. See "Organizing mail" on page 14.

> **Note:** Several programs are available to send and receive mail, including Message Handler (MH) and the **bellmail** program. Which program you use depends on what is installed and configured on your system. For information on your system configuration, contact your system administrator.

Receiving and handling mail

The **mail** program enables you to examine each message in a mailbox and then delete or file a message in a personal mail directory.

The command shell notifies you that mail has arrived. The notification is displayed before the next prompt, provided that the **MAIL** environment variable is set and provided that the interval specified by **MAILCHECK** has elapsed since the shell last checked for mail. The notification message is the value of the **MAILMSG** environment variable. Depending on which shell you are using (bourne, korn, or C shell), the notification is similar to the following:

```
YOU HAVE NEW MAIL
```

This section discusses the concepts and procedures on the following mail tasks:

Starting the mail program

Use the **mail** command to read and remove messages from your system mailbox. Do not use the system mailbox to store messages. Store messages in your personal mailbox and in mail folders.

Prerequisites

The mail program must be installed on your system.

Checking your system mailbox

At your system command line prompt, enter the **mail** command:

```
mail
```

If there is no mail in your system mailbox, the system responds with a message:

```
No mail for YourID
```

If there is mail in your mailbox, the system displays a listing of the messages in your system mailbox:

```
Mail Type ? for help.
"/usr/mail/lance": 3 messages 3 new
  >N    1 karen Tue Apr 27 16:10 12/321 "Dept Meeting"
   N    2 lois  Tue Apr 27 16:50 10/350 "System News"
   N    3 tom   Tue Apr 27 17:00 11/356 "Tools Available"
```

The current message is always prefixed with a greater-than symbol (>). Each one-line entry displays the following fields:

status Indicates the class of the message.
number Identifies the piece of mail to the mail program.
sender Identifies the address of the person who sent the mail.
date Specifies the date the message was received.
size Defines the number of lines and characters contained in the message (this includes the header).
subject Identifies the subject of the message, if it has one.

The status can be any of the following:

N A new message.
P A message that will be preserved in your system mailbox.
U An unread message. This is a message that was listed in the mailbox the last time you used the mail program, but the contents were not examined.
***** A message that was saved or written to a file or folder.

A message without a status indicator means that the message has been read but has not been deleted or saved.

Checking your personal mailbox or mail folder

At your system command line prompt, you can use the **mail** command in the ways shown in the following examples:

mail -f A listing of the messages in your personal mailbox, **$HOME/mbox**, is displayed.

If there is no mail in your personal mailbox, the system responds with the message:

```
"/u/george/mbox": 0 messages
```

OR

```
A file or directory in the path name does not exist
```

mail -f +`dept` In this example, a listing of the messages in the **dept** folder is displayed.

If there is no mail in your mail folder, the system responds with the message:

```
A file or directory in the path name does not exist
```

Displaying the contents of your mailbox

From the mailbox prompt, you can enter mailbox subcommands to manage the contents of the mailbox.

Prerequisites

1. The mail program must be installed on your system.
2. The mail program must be started.
3. There must be mail in your mailbox.

Displaying a message within a specific range of messages

Use the (**h**)eaders subcommand to view a message contained within a specific range of messages. At your mailbox prompt, you can use the **h** subcommand in the ways shown in the following examples:

h Approximately 20 messages are displayed at a time. The actual number displayed is determined by the type of terminal being used and the **set screen** option in your **.mailrc** file. If you enter the **h** subcommand again, the same range of messages is displayed.

h 21 Message 21 and subsequent messages, up to and including message 40 (if you have that number of messages in your mailbox), are displayed. Continue typing the **h** subcommand with the subsequent message number until all messages have been displayed.

h 1 To return to the first group of 20 messages, enter any number within the range of 1-20.

Scrolling your mailbox

Use the **z** subcommand to scroll your mailbox. At your mailbox prompt, you can use the **z** subcommand in the ways shown in the following examples:

z Approximately 20 messages are displayed at a time. The actual number displayed is determined by the type of terminal being used and the **set screen** option in your **.mailrc** file. enter the **z** subcommand again to scroll to the next 20 messages.

z + The plus sign (**+**) argument scrolls to the next 20 messages. Message 21 and subsequent messages, up to and including message 40 (if you have that number of messages in your mailbox), are displayed. Continue typing the **z+** subcommand until all messages have been displayed. The system will respond with the following message:

```
On last screenful of messages.
```

z - The minus sign (**-**) argument scrolls to the previous 20 messages. When you reach the first set of messages, the system will respond with the following message:

```
On first screenful of messages.
```

Searching and displaying information about specific messages

At your mailbox prompt, you can use the (**f**)rom subcommand in the ways shown in the following examples:

f Displays header information for the current message.

f 1 4 7 Displays header information for the specific messages 1, 4, and 7.

f 1-10 Displays header information for a range of messages 1 through 10.

f * Displays all messages.

f ron Messages, if any, from user ron are displayed. The characters entered for an address do not need to exactly match the address; therefore, the request for address ron in either uppercase or lowercase letters matches all of the following addresses:

```
RoN
ron@topdog
hron
rOn
```

f /meet Messages, if any, where the **Subject:** field contains the letters meet are displayed. The characters entered for a pattern do not need to exactly match the **Subject:** field. They must only be contained in the **Subject:** field in either uppercase or lowercase letters; therefore, the request for subject meet matches all of the following subjects:

```
Meeting on Thursday
Come to meeting tomorrow
MEET ME IN ST. LOUIS
```

Displaying the current message number

At your mailbox prompt, you can use the **=** subcommand in the way shown in the following example:

= The current message number is displayed.

Checking number of messages in your mailbox

At your mailbox prompt, you can use the **folder** subcommand in the way shown in the following example:

folder Lists information about your folder or mailbox. The system will respond similarly to the following:

```
"/u/lance/mbox": 29 messages.
```

Reading mail

You can read your mail in several ways. The following sections provide examples of each method. Choose the method you are most comfortable with, and use it to read your mail.

Prerequisites

1. The mail program must be installed on your system.
2. The mail program must be started.
3. There must be mail in your system mailbox.

Reading messages in your mailbox

At your mailbox prompt, you can use the (**t**)ype or (**p**)rint subcommands in the ways shown in the following examples:

`3`	If you use the number of the message, by default, the text of the message is displayed.
t	If you use the **t** subcommand, by default, the text of the current message is displayed.
t `3`	The text of message 3 is displayed.
t `2 4 9`	The text for messages 2, 4, and 9 is displayed.
t `2-4`	The text for the range of messages 2 through 4 is displayed.
p	If you use the **p** subcommand, by default, the text of the current message is displayed.
p `3`	The text of message 3 is displayed.
p `2 4 9`	The text for messages 2, 4, and 9 is displayed.
p `2-4`	The text for the range of messages 2 through 4 is displayed.

When you display more than one message at a time, be sure to include the **set crt** option in your **$HOME/.mailrc** file so you can scroll through your messages. You also can enter this subcommand at the mailbox prompt. If you do not use this subcommand and you have more than one screen of messages, the displayed messages scroll up and off the screen before you have time to read them.

Reading the next message in your mailbox

At your mailbox prompt, you can use the (**n**)ext or plus sign (**+**) subcommand in the way shown in the following example:

n or **+**	Displays the text of the next message, and this message becomes the current message.

You can also press the Enter key to display the text of the next message.

Reading the previous message

At your mailbox prompt, you can use the **-** subcommand in the way shown in the following example:

-	The text of the previous message is displayed.

Deleting mail

When deleting a message, you can delete the current message, delete a specific message, or delete a range of messages. You can also delete the current message and display the next message by combining subcommands.

Prerequisites

1. The mail program must be installed on your system.
2. There must be mail in your system mailbox.
3. The mail program must be started.

Deleting messages

At your mailbox prompt, you can use the (**d**)elete subcommand in the ways shown in the following examples:

d	The current message is deleted.
dp or **dt**	The current message is deleted and the next message is displayed. This also can be accomplished by including the **set autoprint** option in the **.mailrc** file, which will set the **d** subcommand to function like the **dp** or **dt** subcommand combination.
d 4	Deletes the specific message 4.
d 4-6	Deletes a range of messages 4 through 6.
d 2 6 8	Deletes messages 2, 6, and 8.

Undeleting messages

At your mailbox prompt, you can use the (**u**)ndelete subcommand in the ways shown in the following examples:

u	The current message is undeleted.
u 4	Undeletes the specific message 4.
u 4-6	Undeletes a range of messages 4 through 6.
u 2 6 8	Undeletes messages 2, 6, and 8.

Exiting mail

Prerequisites

1. The mail program must be installed on your system.
2. There must be mail in your system mailbox.
3. The mail program must be started.

Exiting mail and saving your changes to the original contents of the mailbox

At your mailbox prompt, you can use the (**q**)uit subcommand in the following ways:

If You Are Exiting the System Mailbox:

q The **q** subcommand leaves the system mailbox and returns to the operating system. When you leave the mailbox, all messages marked to be deleted are removed from the mailbox and cannot be recovered. The mail program saves the messages you read in your personal mailbox (**mbox**). If you did not read any of your mail, the messages remain in the system mailbox until acted upon.

If You Are Exiting Your Personal Mailbox or a Mail Folder:

q When using the **q** subcommand in your personal mailbox or a mail folder, messages read and not read will remain in your personal mailbox or in a mail folder until acted upon.

Exiting mail without changing the original contents of the mailbox

x or **ex** The **x** or **ex** subcommand allows you to leave the mailbox and return to the operating system without changing the original contents of the mailbox. The program ignores any requests you made prior to the **x** request; however, if you did save a message to another folder, the save will occur.

Organizing mail

Use folders to save messages in an organized fashion. You can create as many folders as you need. Give each folder a name that pertains to the subject matter of the messages it contains, similar to file folders in an office filing system. Each folder is a text file that is placed in the directory you specify in your **.mailrc**

file with the **set folder** option. You must create this directory before using folders to store messages. When the directory exists, the mail program creates the folders in that directory as needed. If you do not specify a directory with the **set folder** option in your **.mailrc** file, the folder is created in your current directory. Using the mail program, you can put a message into a folder from the system mailbox, a personal mailbox, or another folder.

You can add the contents of a message to a file or folder using the (**s**)ave or (**w**)rite subcommands. Both of these subcommands append information to an existing file or create a new file if it does not exist. Information currently in the file is not destroyed. If you save a message from your system mailbox to a file or folder, the message is deleted from your system mailbox and transferred to the file or folder specified. If you save a message from your personal mailbox or folder to another file or folder, the message is not deleted from your personal mailbox but is copied to the specified file or folder. When using the **s** subcommand, you can read the folder like a mailbox because the messages and the header information are appended at the end of the folder. When using the **w** subcommand, you can read the folder like a file because the message is appended without header information at the end of the file.

Prerequisites

1. The mail program must be installed on your system.
2. There must be mail in your system mailbox, personal mailbox, or a folder you have defined.
3. The mail program must be started.

Creating a letters mailbox directory to store messages in folders

1. To check if the **set folder** option has been enabled in the **.mailrc** file, enter the following subcommand at the mailbox prompt:

   ```
   set
   ```

   The **set** subcommand displays a list of the enabled mail options in your **.mailrc** file.

   If the **set folder** option has been enabled, the system responds with a message similar to the following:

   ```
   folder /home/george/letters
   ```

   In this example, **letters** is the directory in which mail folders will be stored.
2. If the **set folder** option has not been enabled, add a line similar to the following in the **.mailrc** file:

   ```
   set folder=/home/george/letters
   ```

   In this example, **/home/george** is George's home directory and **letters** is the directory in which mail folders will be stored. The **set folder** option enables you to use the plus sign (+) shorthand notation at your mailbox prompt to save messages in your **letters** directory.
3. You must create a **letters** directory in your home directory. In your home directory at the system command line prompt, type:

   ```
   mkdir letters
   ```

Saving messages with headers

Use the (**s**)ave subcommand in the following ways:

s `1-4 notes`
: Saves messages `1`, `2`, `3` and `4` with their header information to a folder called **notes** in the current directory.

  The mail program responds with the following message:

  ```
  "notes" [Appended] 62/1610
  ```

s +`admin`
: Saves the current message to an existing folder called **admin** in your folder directory.

  If the folder directory is defined as **/home/george/letters** in your **.mailrc** file, the system responds with:

  ```
  "/home/george/letters/admin" [Appended] 14/321
  ```

s 6 **+**admin

Saves message 6 to an existing folder called **admin** in your folder directory.

If the folder directory is defined as **/home/george/letters** in your **.mailrc** file, the system responds with:

```
"/home/george/letters/admin" [Appended] 14/321
```

Saving messages without headers

Use the (**w**)rite subcommand to save a message as a file instead of as a folder. To read or edit a file saved with the **w** subcommand, you must use vi or some other text editor. At your mailbox prompt, you can use the **w** subcommand in the following ways:

w 6 pass

Saves only the text of message 6 to a file called **pass** in the current directory.

If the **pass** file does not already exist, the system responds with the following message:

```
"pass" [New file] 12/30
```

If the **pass** file exists, the system responds with the following message:

```
"pass" [Appended] 12/30
```

w 1-3 safety

Saves only the text of the specific messages 1, 2, and 3 to a file called **safety** in the current directory.

The text of the messages in this example will be appended one after the other into one file. If the **safety** file does not already exist, the system responds with the following message:

```
"safety" [New file] 12/30
```

Finding the name of the current mailbox or folder

Although the **mail** command displays the name of the current mailbox when it starts, you might lose track of which mailbox you are in. At your mailbox prompt, you can use the **folder** subcommand in the way shown in the following example:

folder

Finds the name of your current mailbox or folder.

If the current mailbox is /home/lance/mbox**,** the following is displayed:

```
/home/lance/mbox: 2 messages 1 deleted
```

This message indicates that /home/lance/mbox is the current mailbox you are in, it contains two messages, and one of those messages will be deleted when you finish with this mailbox.

Changing to another mailbox

Changing to another mailbox is like quitting a mailbox or folder. Any messages that you marked to be deleted are deleted when you leave that mailbox. The deleted messages cannot be recovered. At your mailbox prompt, you can use the **file** or **folder** subcommand in the way shown in the following example:

folder **+**project

Once the mail program is started with one mailbox, use the **file** or **folder** subcommands to change to another mailbox.

If you change from the **mbox** file to the **project** folder and you have deleted all the messages in the **mbox** file, the mail program displays:

```
/home/dee/mbox removed
+project: 2 messages 2 new
```

followed by a list of the messages in the **project** folder.

Creating and sending mail

You can use the **mail** program to create, send, reply, and forward messages to other users or to send ASCII files to other users. An ASCII file might, for example, be a document you have written using a preferred editor or a source file for a program.

You can send messages and files to a user on your local system, on your network, or to a user on another connected network. The recipient does not need to be logged onto the system when you send the information. Mail is sent to a user's address.

For more information about creating and sending mail, see:

Addressing mail

Mail is sent to a user's address. The address, containing the login name and system name, directs the delivery of the mail message. Generally, to send a message to another user, you must enter the **mail** command and the address as follows:

```
mail User@Address
```

How you address mail depends on whether you are sending the mail:

- To users on your local system
- To users on your network
- To users on a different network
- Over a BNU or UUCP link

However, the format of the *Address* parameter depends upon the location of the recipient. The concept is similar to how you might address a note to a fellow worker in an office. To send a note to Ryan, who works in a small department of six to eight people, you might write the name on an envelope and put it in the office mail system. However, if Ryan is in another department, you might have to provide more information on the envelope:

```
Ryan
Payroll
```

If Ryan is in another geographic location, you might need even more information to ensure that the message reaches him:

```
Ryan
Payroll
Gaithersburg
```

mail `ryan`	To send mail electronically, use a similar addressing progression. To send mail to a user on your local system, the login name is the only part of the address required.
mail `ryan@tybalt`	To send mail to a user on your local network, enter the full system (node) address.
mail `ryan@mars.aus.dbm.com`	To send mail to a user on another connected network, enter the full system address and network address.
mail `dept71`	You can send mail to a specific group of people by using an alias or distribution list. To do so, you must create an alias or distribution list in your **.mailrc** file. If you need information on creating aliases, see “Creating an alias or distribution list” on page 32.

Addressing mail to more than one user

To address mail to more than one user at the same time, separate each user name with a space. For example:

```
ryan@tybalt suemc@julius dmorgan@ophelia
```

Prerequisites

The mail program must be installed on your system.

Addressing mail to users on your local system

To send a message to a user on your local system (to someone whose login name is listed in your **/etc/passwd** file), use the login name for the address. At your system command line prompt, you can use the **mail** command in the way shown in the following example:

mail *LoginName*

mail `ryan`	If Ryan is on your system and has the login name `ryan`, this command activates the mail program, enables you to create a message, and tries to send the message to a local login name of `ryan`. If the message is delivered successfully, you receive no notification. If Ryan is not on your system, the mail system immediately returns an error message and returns the unsent message to your system mailbox.

Addressing mail to users on your network

To send a message through a local network to a user on another system, at the command line, type:

mail *LoginName*@*SystemName*	For example, if Ryan is on system `zeus`, use the following command to create and send a message to him: `mail ryan@zeus` This command activates the mail program, enables you to create a message, and tries to send the message to login name `ryan` on system `zeus`. If the message is delivered successfully, you receive the system prompt with no notification. If the mail address is incorrect, you receive an error message.

Note: To send a message through a local network to a user on another system, you must know the login name and the name of the other system. For more information on this, see “Communications system commands” on page 6.

Addressing mail to users on a different network

If your network is connected to other networks, you can send mail to users on the other networks. The address parameters differ depending on how your network and the other networks address each other and how they are connected.

Using a Central Database of Names and Addresses: At your system command line prompt, use the **mail** command in the way shown in the following example:

mail *LoginName@SystemName*	If the networks use a central database of names, you do not need any additional information to send mail to users on the connected networks. Use the same addressing format as for users on your local network. This type of addressing works well when the nature of the network allows a central database of names to be maintained.

Using Domain Name Addressing: At your system command line prompt, use the **mail** command in the ways shown in the following examples:

mail *LoginName@SystemName.DomainName*	For networks that span large, unrelated networks in widespread locations, a central database of names is not possible. The *DomainName* parameter defines the remote network, relative to your local network, within the defined structure for the larger group of interconnected networks.
mail `kelly@merlin.odin.valryanl`	In this example, user `kelly` is on the system `merlin` that is on a local network, named `odin`, that is connected to a second network whose domain is called `valryanl`.

Addressing mail over a BNU or UUCP link

To send a message to a user on another system connected to your system by the Basic Networking Utilities (BNU) or another version of UNIX-to-UNIX Copy Program (UUCP), you must know:

- The login name
- The name of the other system
- The physical route to that other system

The person responsible for connecting your system to other systems should be able to provide routing information to address the other system.

When Your Computer Has a BNU or UUCP Link: At your system command line prompt, use the **mail** command in the ways shown in the following examples:

mail *UUCPRoute*`!`*LoginName*	If your local computer has a BNU or UUCP connection that can be used to reach the remote system, use the format in this example to address a message. The variable *LoginName* is the login name on the remote system for the message recipient. The variable *UUCPRoute* describes the physical route the message must follow along the UUCP network. If your system is connected to the remote system without any intermediate UUCP systems between, this variable is the name of the remote system.
mail`arthur!lancelot!merlin!ken`	If your message must travel through one or more intermediate UUCP systems before reaching the desired remote system, this variable is a list of each of the intermediate systems. The list starts with the nearest system and proceeds to the farthest system, separated by an exclamation mark (!). You can follow this example, if the message must travel through systems `arthur` and `lancelot` (in that order) before reaching `merlin`.
mail `merlin!ken`	If your local system has a UUCP link to a system called `merlin` and there are no other UUCP systems between your system and `merlin`, you can send a message to `ken` on that system.

When the BNU or UUCP Link Is on Another Computer: In a local or wide area network environment, one of the systems on the network may have a BNU or other type of UUCP connection to a remote

system. You can use that UUCP connection to send a message to a user on that remote UUCP system. At your system command line prompt, use the **mail** command in the way shown in the following example:

mail @arthur:merlin!ken
: Sends mail to `ken` on UUCP system `merlin` from the Internet system `arthur`. The delimiter @ is for the internet addressing, the delimiter ! is for the UUCP addressing, and the delimiter : connects the two addresses. Notice that in this format you are not sending mail to a user at any of the intermediate systems, so no login name precedes the @ in the domain address.

mail @arthur:odin!acct.dept!kelly
: Sends mail to `kelly` on UUCP system `acct.dept` through system `odin` from the Internet system `arthur`.

mail@odin.uucp:@dept1.UUCP:@dept2:bill@dept3
: Sends mail to `bill@dept3` over `odin` and `dept1` UUCP links and then over the local network link between systems `dept2` and `dept3`. The **/etc/sendmail.cf** file must be configured accordingly to use this type of UUCP address notation. See your system administrator for information.

If you often send mail to users on other networks, creating aliases that include the users' addresses can save you time. See "Creating an alias or distribution list" on page 32.

Starting the mail editor

The **mail** program provides a line-oriented editor for creating messages. This editor enables you to type each line of the message, press the Enter key to get a new line, and type more text. You cannot change a line after you have pressed the Enter key. However, before pressing the Enter key, you can change information on that one line by using the Backspace and Delete keys to erase. You can also use mail editor subcommands to enter a full-screen editor and change the message.

When you create mail with the mail editor, the **date:** and **from:** fields are automatically completed by the system. You have the option to complete the **subject:** and **cc:** fields. These fields are similar to the body of a standard business letter.

The mail editor includes many control subcommands that enable you to perform other operations on a message. Each of these subcommands must be entered on a new line and must begin with the special *escape* character. By default, the escape character is a tilde (~). You can change it to any other character by including the **set escape** option in your **.mailrc** file.

Prerequisites

1. The mail program must be installed on your system.
2. The mail program must be started.

Starting the mail editor from the command line or the mailbox prompt

At your system command line prompt or your mailbox prompt, you can use the **mail** command in the ways shown in the following examples:

mail *User@Address*	Issue this command from the command line prompt. The message is addressed to *User@Address*. The *Address* parameter depends upon the location of the recipient.
m *User@Address*	Issue this subcommand from the mailbox prompt. The message is addressed to *User@Address*. The *Address* parameter depends upon the location of the recipient.

The mail editor is also activated, if you use the (**R**)eply or (**r**)eply subcommands to reply to a message. For more information on how to reply to a message, see "Sending mail" on page 26 and "Replying to mail" on page 26.

Editing a message

While in your mailbox, you can add information to an existing message by typing the (**e**)dit or (**v**)isual subcommand at the mailbox prompt. While in the mail editor, you cannot change information on a line after you have pressed the Enter key and gone on to the next line. You can change the content of your message before sending it by editing the message with another editor.

Prerequisites

1. The mail program must be installed on your system.
2. The alternate editor must be defined in the **.mailrc** file with:

   `set EDITOR=`*`PathName`*

   This defines the editor you activate with the **~e** subcommand. The value of *PathName* must be the full path name to the editor program that you want to use. For example, the definition `set EDITOR=/usr/bin/vi` defines the vi editor for use with the **~e** subcommand.
3. To add information to a message in your mailbox, you must have started the **mail** command to read mail in the system mailbox, other mailbox, or folder.
4. To start an alternate editor while creating a message, you must be in the mail editor prompt.

Adding information to a specific message in your mailbox

To add information to a message in your mailbox, enter the (**e**)dit subcommand or the (**v**)isual subcommand, followed by the message number. At the mailbox prompt, you can use either the **e** subcommand or the **v** subcommand in the ways shown in the following examples:

e `13` To add a note to message `13` using the **e** editor (or whatever editor is defined in the **.mailrc** file).
v `15` To add a note to message `15` using the **vi** editor (or whatever editor is defined in the **.mailrc** file).

If you do not specify a message number, the **mail** command activates the editor using the current message. When you leave the editor, you return to the mailbox prompt to continue processing the messages in the mailbox.

Changing the current message while in the mail editor

At the beginning of a line while in the mail editor, you can use either the **~e** subcommand or the **~v** subcommand in the ways shown in the following examples:

~e Activates the **e** editor or other editor that you define in the **.mailrc** file.
~v Activates the **vi** editor or other editor that you define in the **.mailrc** file.

This enables you to edit the text of the current message. When you leave the different editor, you return to the mail editor.

Displaying a message while in the mail editor

Prerequisites

1. The mail program must be installed on your system.
2. To display a message while in the mail editor, you must have started the mail editor. If you need information on this, see “Starting the mail editor” on page 20.

Displaying the lines of a message

At the beginning of a line while in the mail editor, use the **~p** subcommand in the way shown in the following example:

~p The editor displays the contents of the message including the header information for the message. The text scrolls up from the bottom of the display. The end of the message is followed by the mail editor’s `(Continue)` prompt.

If the message is larger than one screen and you have not set the page size for your terminal by using the **stty** command, the text scrolls off the top of the screen to the end. To look at the content of large messages, use the mail editor subcommands to view the message with another editor. If you need information on this, see “Editing a message” on page 21.

Exiting the mail editor without sending the message

To quit the mail editor without sending the message, use the **~q** subcommand or the interrupt key sequence (usually the Alt-Pause or Ctrl-C key sequence). If you have entered any text, the **mail** command saves the message in the **dead.letter** file.

Prerequisites

1. The mail program must be installed on your system.
2. To display a message while in the mail editor, you must have started the mail editor. If you need information on this, see “Starting the mail editor” on page 20.

Quitting the mail editor

At the beginning of a line while in the mail editor, you can use the **~q** subcommand in the way shown in the following example:

~q Quits the mail editor and the message is not sent. The message is saved in the **dead.letter** file in your home directory, unless you have not entered any text. The system prompt is displayed.

Ctrl-C To quit the editor using an interrupt key sequence, press the break (Ctrl-C key sequence) or the interrupt (Alt-Pause key sequence). The following message is displayed:

```
(Interrupt -- one more to kill letter)
```

Press break or interrupt again.

```
(Last Interrupt -- letter saved in dead.letter)
```

The message is not sent. The message is saved in the **dead.letter** file in your home directory, unless you have not entered any text. The system prompt is displayed.

Note: When you exit the mail editor without sending the message, the previous content of the **dead.letter** file is replaced with the incomplete message. To retrieve the file, see “Adding a file and a specific message within a message.”

Adding a file and a specific message within a message

Prerequisites

1. The mail program must be installed on your system.
2. You must know the name and address of the mail recipient.
3. The mail editor must be started.

Including files in a message

At the beginning of a line while in the mail editor, you can use the **~r** subcommand in the way shown in the following example:

~r *schedule* Where *schedule* is the name of the file to include. In this example, the information in the file *schedule* is included at the current end of the message being written.

Including a specific message within a message

At the beginning of a new line while in the mail editor, you can use either the **~f** or **~m** subcommands in the ways shown in the following examples:

~f *MessageList* Appends the indicated message or messages to the end of the current message, but does *not* indent the appended message. Also, use this subcommand to append messages for reference whose margins are too wide to embed with the **~m** subcommand.

> **Note:** The parameter *MessageList* is a list of integers that refer to valid message numbers in the mailbox or folder being handled by mail. You can enter simple ranges of numbers also. For example:
>
> **~f 1-4** Appends messages 1, 2, 3, and 4 to the end of the message being written. These messages are aligned with the left margin (not indented).

~m 2 Appends the indicated message to the end of the current message. The included message is indented one tab character from the normal left margin of the message. In this example, message 2 is appended to the current message.

~m 1 3 Appends message 1 and then message 3 to the end of the message being written, indented one tab from the left margin.

Adding the contents of the dead.letter file into your current message

At the beginning of a new line while in the mail editor, you can use the **~d** subcommand in the way shown in the following example:

~d Retrieves and appends the contents of the **dead.letter** file to the end of the current message. At the (Continue) prompt, continue by adding to the message or by sending the message.

Changing or adding to the header fields of a message

The header of a message contains routing information and a short statement of the subject. You must specify at least one recipient of the message. The rest of the header information is not required. The information in the header can include the following:

To: Contains the address or addresses for sending the message.

Subject: Contains a short summary of the topic of the message.

Cc: Contains the address or addresses for sending copies of the message. The contents of this field are part of the message sent to all who receive the message.

Bcc: Contains the address or addresses for sending *blind* copies of the message. This field is *not* included as part of the message sent to all who receive the message.

You can customize the mail program to automatically ask for the information in these fields by putting entries in your **.mailrc** file. For more information, see “Customizing the mail program” on page 30.

Prerequisites

1. The mail program must be installed on your system.
2. Start the mail editor and begin editing a message.

Editing the header information

To add to or change information in more than one header field, use the **~h** subcommand. When you enter this subcommand on a new line while in the mail editor, the system displays each of the four header fields, one at a time. You can:

- View the content of each field.
- Delete information from that field (using the Backspace key).
- Add information to that field.

Pressing the Enter key saves any changes to that field and displays the next field and its contents. When you press the Enter key for the last field (**Bcc:**), you return to the editor.

1. Enter the following subcommand at the beginning of a new line while in the mail editor:
   ```
   ~h
   ```

   The system responds with the contents of the **To:** field and places the cursor at the end of that field:
   ```
   To: mark@austin_
   ```

   If necessary, edit or add to the field. Then press the Enter key.
2. The system responds with the contents of the **Subject:** field:
   ```
   Subject: Fishng Trip_
   ```

   Note: If you have changed this field, the cursor may not be at the end of the field.

   To correct the misspelling in the subject, use the Cursor Right or Cursor Left key to position the cursor under the `n` in `Fishng`. Re-type the rest of the subject to correct it to `Fishing Trip`. Press the Enter key.
3. The system responds with the contents of the **Cc:** field:
   ```
   Cc: mel@gtwn_
   ```

   To add another person to the copy list, ensure the cursor is at the end of the list, type a space, and type the new address:
   ```
   Cc: mel@gtwn geo@austin
   ```

   This expands the copy list to two persons. When you have completed the copy list, press the Enter key.
4. The system responds with the contents of the **Bcc:** field. Make any changes and press the Enter key.
5. The system responds with the `(Continue)` prompt and returns you to the mail editor at the current end of the message.

Setting or resetting the Subject: field

Use the **~s** subcommand to set the **Subject:** field to a particular phrase or sentence. Using this subcommand replaces the previous contents (if any) of the **Subject:** field. At the beginning of a new line while in the mail editor, you can use the **~s** subcommand in the way shown in the following example:

~s `Fishing Trip`

This changes the current **Subject:** field:
```
Subject: Vacation
```

To the following:
```
Subject: Fishing Trip
```

Note: You cannot append to the **Subject:** field with this subcommand. Use the **~h** subcommand, as described in “Editing the header information” on page 23.

Adding users to the To:, Cc:, and Bcc: fields

At the beginning of a new line while in the mail editor, you can use the **~t**, **~c** , or **~b** subcommands in the ways shown in the following examples:

~t `geo@austin mel@gtwn`

This changes the current **To:** field:
```
To: mark@austin
```

to the following:
```
To: mark@austin geo@austin mel@gtwn
```

~c `geo@austin mel@gtwn` This changes the current **Cc:** field:

```
Cc: mark@austin amy
```

to the following:

```
Cc: mark@austin amy geo@austin mel@gtwn
```

~b `geo@austin mel@gtwn` This changes the current **Bcc:** field:

```
Bcc: mark@austin
```

to the following:

```
Bcc: mark@austin geo@austin mel@gtwn
```

Note: You cannot use the **~t**, **~c**, or**~b** subcommands to change or delete the contents of the **To:**, **Cc:**, and **Bcc:** fields. Use the ~h subcommand, as described in "Editing the header information" on page 23.

Reformatting a message in the mail editor

Prerequisites

1. The mail program must be installed on your system.
2. The **fmt** command must be installed on your system.

Reformatting a message in the mail editor

After typing the message and before sending it, you can reformat the message to improve its appearance by using the **fmt** shell program. At the beginning of a new line while in the mail editor, you can use the **fmt** command in the way shown in the following example:

~| fmt Changes the appearance of the message by re-flowing the information for each paragraph within defined margins (a blank line must separate each paragraph). The pipe (**|**) subcommand pipes the message to standard input of the command and replaces the message with the standard output from that command.

Attention: Do not use the **fmt** command if the message contains embedded messages or pre-formatted information from external files. The **fmt** command reformats the header information in embedded messages and may change the format of pre-formatted information. Instead, use the **~e** or **~v** subcommand to enter a full-screen editor and reformat the message.

Checking for misspelling in the mail editor

Prerequisites

1. The mail program must be installed on your system.
2. The text-formatting programs must be installed on your system.

Spell-checking your message

Use the **spell** command to check your message for misspelled words, from the mail editor:

1. Write the message to a temporary file. For example, to write the message to the **checkit** file, type:

   ```
   ~w checkit
   ```

2. Run the **spell** command using the temporary file as input. Type:

   ```
   ~! spell checkit
   ```

   In this example, the exclamation point (!) is the subcommand that starts a shell, runs a command, and returns to the mailbox. The **spell** command responds with a list of words that are not in its list of known words, followed by an exclamation point (!) to indicate that you have returned to the mail program.
3. Examine the list of words. Determine if you need to use an editor to make corrections.

4. Type the following to delete the temporary file:

   ```
   ~! rm checkit
   ```

Sending mail

Prerequisites

- The mail program must be installed on your system.
- You must know the name and address of the mail recipient.

1. Enter the **mail** command on the command line, followed by the name and address of the recipient (or recipients) of the message. For example:

   ```
   >mail jan@brown
   ```

   The system responds with:

   ```
   Subject:
   ```

2. Type the subject of the message. For example:

   ```
   Subject: Dept Meeting
   ```

   and press Enter. You can now type the body of the text.

3. Type your message. For example:

   ```
   There will be a short department meeting this afternoon
   in my office. Please plan on attending.
   ```

4. To send a message you have typed with the mail editor, press the end-of-text character, which is usually the Ctrl-D key sequence or a period (**.**), while at the beginning of a new line in the message.

   The system displays the **Cc:** field:

   ```
   Cc:
   ```

5. Type the names and addresses of those users who should receive copies of the message. For example:

   ```
   Cc: karen@hobo cliff@cross
   ```

   Note: If you do not want to send copies, press Enter without typing.

When you press the Enter key, the message is delivered to the specified address.

Note: If you enter an address unknown to the system, or not defined in an alias or distribution list, the system responds with the login name followed by an error message: `[user ID]... User unknown`.

Replying to mail

Prerequisites

1. The mail program must be installed on your system.
2. There must be mail in your system mailbox.

At the mailbox prompt, you can use the (**r**)eply/respond and (**R**)eply/respond subcommands in the ways shown in the following examples:

r Creates a new message that is addressed to the sender of the selected message and copied to the people in the **Cc:** field (if any). The **Subject:** field of the new message refers to the selected message. The default value of the **r** subcommand is the current message. This default can be overridden by typing the message number after the **r**.

R Starts a reply only to the sender of the message. The default value of the **R** subcommand is the current message.

R 4 Starts a reply only to the sender of the message. You can override the current message by typing the message number after the **R**. This example starts a reply to message 4. The system responds with a message similar to the following:

```
To: karen@thor
Subject: Re: Department Meeting
```

You then can type your response:

```
I'll be there.
```

When you finish typing the text, press either the period (**.**) or the Ctrl-D key sequence to send the message. After the reply is sent, you are returned to the mailbox prompt.

Creating a new message while in the mailbox

At the mailbox prompt, you can use the **m** subcommand in the way shown in the following example:

m *Address* The *Address* parameter is any correct user address. This subcommand starts the mail editor and enables you to create a new message while in the mailbox. When you send the message, you will be returned to the mailbox prompt.

Forwarding mail

While reading your mail, you might want to forward a specific note to another user. This task can be accomplished by using the **~f** and **~m** subcommands.

If you are going to be away from your normal network address, you can have your mail sent to another network address by creating the **.forward** file. See “Creating a .forward file” on page 28.The new address can be any valid mail address on your network or a network connected to yours. It can be the address of a co-worker who will handle your messages while you are away. When you choose to forward your network mail, you do not receive a copy of any incoming mail in your mailbox. All mail is forwarded directly to the address or addresses that you have specified.

Prerequisites

1. The mail program must be installed on your system.
2. If forwarding a selected message, start the mail facility with the **mail** command. Make a note of the number of the mail message that you wish to forward.

Forwarding selected messages from within the mailbox

To forward specific mail messages:

1. Create a new message by using the **m** subcommand and specify a recipient by typing the following at the mailbox prompt:

   ```
   m User@Host
   ```

   where *User* refers to the login name of another user, and *Host* is the name of the user’s system. If the user is on your system, you can omit the *@Host* part of the address.
2. Type a subject name at the **Subject:** prompt.
3. To specify the number of the mail message to be forwarded, type:

   ```
   ~f MessageNumber
   ```

   OR

   ```
   ~m MessageNumber
   ```

   MessageNumber identifies the piece of mail to forward.

   The **mail** command displays a message similar to the following:

   ```
   Interpolating: 1
   (continue)
   ```

4. To exit mail, type a period (.) on a blank line. At the **Cc:** prompt, type any additional names to whom you wish to forward a mail message.

Forwarding all mail

To forward all your mail to another person:

1. Enter the **cd** command with no parameters to ensure you are in your home directory. For example, type the following for login name `mary`:

   ```
   cd
   pwd
   ```

   The system responds with:

   ```
   /home/mary
   ```

2. Create a **.forward** file in your home directory. See "Creating a .forward file."

   Note: You will not receive any mail until you delete the **.forward** file.

Creating a .forward file

The **.forward** file contains the network address or addresses that will receive your forwarded network mail. Addresses must take the form of *User*`@`*Host*. *User* refers to the login name of another user, and *Host* is the name of the user's system. If the user is on your system, you can omit the `@`*Host* part of the address. You can use the **cat** command to create a **.forward** file as follows:

```
cat > .forward
mark
joe@saturn
[END OF FILE]
```

`[END OF FILE]` represents the end of file character, which is the Ctrl-D key sequence on most terminals. This must be typed on a blank line.

The **.forward** file contains the addresses of the users you want your mail forwarded to. Your mail will be forwarded to `mark` on your local system, and to `joe` on system `saturn`.

This file must contain valid addresses. If it is a null file (zero length), your mail is not forwarded and is stored in your mailbox.

Note: You will not receive any mail until you delete the **.forward** file.

Canceling forwarded mail

To stop forwarding mail, delete the **.forward** file as follows:

Use the **rm** command to remove the **.forward** file from your home directory:

```
rm .forward
```

Sending a vacation message notice

Prerequisites

The mail program must be installed on your system.

1. Initialize the vacation message in your **$HOME** (login) directory by typing:

   ```
   vacation -I
   ```

   This creates a **.vacation.dir** file and a **.vacation.pag** file where names of the people who send messages are kept.

2. Modify the **.forward** file. For example, `carl` types the following statement in the **.forward** file:

   ```
   carl, |"/usr/bin/vacation carl"
   ```

The first `carl` entry is the user name to which mail is forwarded. The second `carl` entry is the user name of the sender of the vacation message. The sender of the mail message receives one vacation message from `carl` per week, regardless of how many messages are sent to `carl` from the sender. If you have your mail forwarded to someone else, the mail message from the sender is forwarded to the person defined in your **.forward** file.

Use the **-f** flag to change the frequency intervals at which the message is sent. For example, `carl` types the following statement in the **.forward** file:

```
carl, |"/usr/bin/vacation -f10d carl"
```

The sender of mail messages receives one vacation message from `carl` every ten days, regardless of how many messages are sent to `carl` from the sender.

3. To send a message to each person who sends you mail, create the file **$HOME/.vacation.msg** and add your message to this file. The following is an example of a vacation message:

```
From: carl@odin.austin (Carl Jones)
Subject: I am on vacation.
I am on vacation until October 1. If you have something urgent,
please contact Jim Terry <terry@zeus.valhalla>.
  --carl
```

The sender receives the message that is in the **$HOME/.vacation.msg** file, or if the file does not exist, the sender receives the default message found in the **/usr/share/lib/vacation.def** file. If neither of these files exists, no automatic replies are sent to the sender of the mail message, and no error message is generated.

Canceling vacation messages

To cancel the vacation message, remove the **.forward** file, **.vacation.dir** file, **.vacation.pag** file, and **.vacation.msg** file from your **$HOME** (login) directory as follows:

```
rm .forward .vacation.dir .vacation.pag .vacation.msg
```

Sending and receiving secret mail

Prerequisites

1. The mail program must be installed on your system.
2. A password must have been set up using the **enroll** command.

Sending secret mail

At the system command line prompt, use the **xsend** command in the way shown in the following example:

xsend `barbara`	In this example, secret mail is being addressed to the login name `barbara`. When you press Enter, a single line editor is used to type the text of the message. When you are finished typing your message, press the Ctrl-D key sequence or a period (.) to exit the mail editor and send the message. The **xsend** command encrypts the message before it is sent.

Receiving secret mail

1. At your system command line prompt, type:

```
mail
```

The system displays the list of messages in your system mailbox. The secret mail program sends you a notification that you have received secret mail. The message line will be similar to the following:

```
Mail [5.2 UCB] Type ? for help.
"/usr/spool/mail/linda": 4 messages 4 new
>N 1 robert Wed Apr 14 15:23 4/182 "secret mail from robert@Zeus"
```

The message text directs you to read your secret mail on your host using the **xget** command.

2. At the system command line prompt, type:

```
xget
```

You are prompted for the password that was previously set up using the **enroll** command. After you type your password, the **xget** command prompt is displayed, followed by a listing of any secret mail. The mail program is used to display any secret mail. You must enter the (**q**)uit subcommand if you want to leave read and unread messages in the secret mailbox and prevent the **xget** command from deleting the messages.

Viewing mail help information

You can obtain help information on using the mail program by using the **?**, **man**, or **info** commands.

To get help in the Mailbox	Enter `?` or `help` at the mailbox prompt. The **?** and **help** subcommands display a summary of common mailbox subcommands. You can display a list of all mailbox subcommands (with no summary) by entering the **(l)**ist subcommand.
To get help in the Mail Editor	Type `~?` at the mail editor prompt. The **~?** subcommand displays a summary of common mail editor subcommands.
To get help in secret mail	Type `?` at the mail editor prompt. The **?** subcommand displays a summary of common secret mail subcommands.
To get help using manual pages	Type `man mail` at the system command line prompt. In this example, `mail` is the command name being searched. The system will provide you with ASCII documentation about the **mail** command. At the continuation marker (:), press Enter to view the rest of the document. The **man** command provides information in ASCII form for reference articles on commands, subroutines, and files.

Customizing the mail program

Commands and options in the **.mailrc** and **/usr/share/lib/Mail.rc** files can be customized to fit your personal mailing needs. See “Enabling and disabling mail options” on page 31 for information about mail options.

Characteristics of a mail session that you can customize include:

- **Prompts for the subject of a message.** When you enter the **mail** command, the program asks you to complete a **Subject:** field. When this prompt is displayed, you can fill in a summary of the subject matter of the message. That summary is included at the start of the message when it is read by the recipient. See “Changing the prompt for the Subject: and Carbon Copy (Cc:) fields” on page 32.
- **Prompts for users to get a copy of a message.** You can customize the **.mailrc** file so that when you send a message, the mail program prompts you for the names of other users who should receive copies of the message. See “Changing the prompt for the Subject: and Carbon Copy (Cc:) fields” on page 32.
- **Aliases or distribution lists.** If you send mail on a large network or often send the same message to a large number of people, typing long addresses for each receiver can become tedious. To simplify this process, create an alias or a distribution list in your **.mailrc** file. An *alias* is a name you define that can be used in place of a single user address. A *distribution list* is a name you define that can be used in place of a group of user addresses. See “Creating an alias or distribution list” on page 32.

- **Number of lines displayed when reading messages.** You can change the number of lines of message headers or of message text that scroll across the screen. See “Changing the number of message headers or message text lines displayed in the mail program” on page 33.
- **Information listed in messages.** You can turn off message headers, such as the machine-set `message-id` field. See “Controlling what information is displayed in a message” on page 34.
- **Folder directory for storing messages.** You can create a special directory for storing messages. You can use the shorthand plus sign (+) subcommand to designate that directory when storing messages or looking at folders. See “Creating default folders to store messages” on page 36.
- **Log file for recording outgoing messages.** You can instruct the **mail** program to record all your outgoing messages in a file or a subdirectory in your home directory. See “Keeping a record of messages you send to others” on page 36.
- **Editors for typing messages**. In addition to the mail editor, you can choose two different editors to edit messages. See “Changing text editors for typing messages” on page 37.

For more information on customizing the mail program, see the following:

Enabling and disabling mail options

Options can be either binary or valued. Binary options are either **set** or **unset**, while valued options can be **set** to a specific value.

> **Note:** The form **unset** *option* is equivalent to **set no** *option*.

Use the **pg** command to view the **/usr/share/lib/Mail.rc** file. The contents of the **/usr/share/lib/Mail.rc** file define the configuration of the mail program. Alter the system configuration for your mail program by creating a **$HOME/.mailrc** file. When you run the **mail** command, subcommands in the **.mailrc** file override similar subcommands in the **/usr/share/lib/Mail.rc** file. The **.mailrc** options can be customized and are valid each time you use the mail program.

To execute mail commands that are stored in a file, use the **source** subcommand.

Prerequisites

The mail program must be installed on your system.

Enabling mail options

The mailbox subcommands most commonly used to alter the characteristics of a mail session are:

set Enables mail options.

source Enables mail options that are stored in a file. When reading mail, you can issue this subcommand at the mailbox prompt:

```
source PathName
```

where *PathName* is the path and file containing the mail commands. Commands in this file override the previous settings of any similar commands for the duration of the current session. You can also alter the characteristics of the current mail session by typing commands at the mailbox prompt.

You can set these options while in the mailbox or by making entries in the **.mailrc** file.

Viewing enabled mail options

When reading your mail, enter the **set** subcommand without any arguments to list all of the enabled **.mailrc** options. In this list, you can also see if a folder directory is selected and if a log file is set up to record outgoing messages.

At the mailbox prompt, type:

```
set
```

A message similar to the following is displayed:

```
ask
metoo
toplines 10
```

In this example, two binary options are enabled: **ask** and **metoo**. There is no **askcc** entry in the list. This indicates that the **askcc** option is not enabled. The **toplines** option has been assigned the value `10`. The **ask**, **metoo**, **askcc**, and **toplines** options are described in .mailrc File Format section of the *AIX 5L Version 5.3 Files Reference*.

Disabling mail options

The mailbox subcommands most commonly used to alter the characteristics of a mail session are:

unset	Disables mail options.
unalias	Deletes the specified alias names.
ignore	Suppresses message header fields.

You can set these options while in the mailbox or by making entries in the **.mailrc** file.

> **Note:** The form **unset** *option* is equivalent to **set no** *option*.

Changing the prompt for the Subject: and Carbon Copy (Cc:) fields

Prerequisites

The mail program must be installed on your system.

Enabling or disabling Subject: field prompting

You can enable or disable the **Subject:** field in the way shown in the following examples:

set ask	**Subject:** field prompting is enabled by editing the **.mailrc** file `ask` option.
unset ask	**Subject:** field prompting is disabled by editing the **.mailrc** file `ask` option.

Enabling or disabling Carbon Copy (Cc:) field prompting

You can enable or disable the **Cc:** field in the way shown in the following examples:

set askcc	Carbon copy (**Cc:**) field prompting is enabled by editing the **.mailrc** file `askcc` option.
unset askcc	Carbon copy (**Cc:**) field prompting is disabled by editing the **.mailrc** file `askcc` option.

Creating an alias or distribution list

Prerequisites

1. The mail program must be installed on your system.
2. You must know the names and addresses of users you want to include in your alias or distribution list.

You can create an alias or distribution list in the following ways:

alias	`kath kathleen@gtwn`
	
	In this example, the alias `kath` has been listed for user `kathleen` at address `gtwn`. After you have added this line to your **$HOME/.mailrc** file, to send a message to Kathleen, type the following at the command line prompt:
	`mail kath`
	
	You are now able to send mail to Kathleen using the `kath` alias.

alias	`dept dee@merlin anne@anchor jerry@zeus bill carl`
	
	After you have added this line to your **$HOME/.mailrc** file, to send a message to your department, type the following at the command line prompt:
	`mail dept`
	
	The message you now create and send will go to `dee` on system `merlin`, `anne` on system `anchor`, `jerry` on system `zeus`, and to `bill` and `carl` on the local system.

Listing the aliases and distribution lists

Type the following at the mailbox prompt:

```
alias
```

OR

```
a
```

A list of aliases and distribution lists is displayed.

Changing the number of message headers or message text lines displayed in the mail program

By changing the **.mailrc** file, you can customize the ability to scroll through mailbox lists or through actual messages.

Prerequisites

The mail program must be installed on your system.

Changing the number of displayed lines of the message list

Each message in your mailbox has a one-line heading in the message list. If you have more than 24 messages, the first headings from the message list scroll past the top of your screen. The **set screen** option controls how many lines of the list are displayed at a time.

To change the number of lines of the message list that are displayed at a time, in your **$HOME/.mailrc** file, type:

```
set screen=20
```

In this example, the system will display 20 message headers at a time. Use either the (**h**)eader or **z** subcommand to view additional groups of headers. You can also type this subcommand at the mailbox prompt.

Changing the number of displayed lines in a long message

If you display a message with more than 24 lines, the first lines of the message scroll past the top of the screen. You can use the **pg** command from within mail to browse through long messages if you have included the **set crt** option in the **.mailrc** file. The **set crt** option controls how many lines a message must contain before the **pg** command is started.

For example, if you use the **t** subcommand to read a long message, only one screen (or page) is displayed. The page is followed by a colon prompt to let you know there are more pages. Press the Enter key to display the next page of the message. After the last page of the message is displayed, there is a prompt similar to the following:

```
EOF:
```

At the prompt, you can enter any valid **pg** subcommand. You can display previous pages, search the message for character strings, or quit reading the message and return to the mailbox prompt.

The **set crt** option is entered in the **.mailrc** file as:

set crt=*Lines*

For example:

```
set crt=20
```

specifies that a message must be 20 lines before the **pg** command is started. The **pg** command is started when you read messages with more than 20 lines.

Changing the number of displayed lines at the top of a message

The **top** subcommand enables you to scan through a message without reading the entire message. You control how many lines of a message are displayed by setting the **toplines** option as follows:

set toplines=*Lines*

In this subcommand, the *Lines* variable is the number of lines, starting from the top and including all header fields, that are displayed with the **top** subcommand.

For example, if user `Amy` has the following line in her **.mailrc** file:

```
set toplines=10
```

when Amy runs the **mail** command to read her new messages, the following text is displayed:

```
Mail Type ? for help.
"/usr/mail/amy": 2 messages 2 new>
N 1 george  Wed Jan 6 9:47  11/257 "Dept Meeting"
N 2 mark    Wed Jan 6 12:59 17/445 "Project Planner"
```

When Amy uses the **top** subcommand to browse through her messages, the following partial message is displayed:

```
top 1
Message 1:
From george Wed Jan 6 9:47 CST 1988
Received: by zeus
    id AA00549; Wed, 6 Jan 88 9:47:46 CST
Date: Wed, 6 Jan 88 9:47:46 CST
From: george@zeus
Message-Id: <8709111757.AA00178>
To: amy@zeus
Subject: Dept Meeting
Please plan to attend the department meeting on Friday
at 1:30 in the planning conference room. We will be
```

The message is partially displayed because **toplines** is set to 10. Only lines 1 (the **Received:** field) through 10 (the second line of the message body) are displayed. The first line, `From george Wed Jan 6 9:47 CST 1988`, is always present and does not count in the **toplines** option.

Controlling what information is displayed in a message

By changing the **.mailrc** file, you can control what header information is displayed in a message. Some header information might be already turned off. Examine your **/usr/share/lib/Mail.rc** file for ignored header fields.

Prerequisites

The mail program must be installed on your system.

Preventing the Date, From, and To headers from being displayed

Every message has several header fields at the top. These header fields are displayed when you read a message. You can use the **ignore** subcommand to suppress the display of header fields when a message is read. The format for the **ignore** subcommand is:

ignore [*FieldList*]

The *FieldList* can consist of one or more field names that you want to ignore when you display a message. For example, if user Amy includes the following line in her **.mailrc** file:

```
ignore date from to
```

and the file **/usr/share/lib/Mail.rc** has the line:

```
ignore received message-id
```

the result of using the (**t**)ype subcommand is:

```
t 1
Message 1:
From george Wed Jan 6 9:47 CST 1988
Subject: Dept Meeting
Please plan to attend the department meeting on Friday
at 1:30 in the planning conference room. We will be
discussing the new procedures for using the project
planning program developed by our department.
```

The **Received:**, **Date:**, **From:**, **Message-Id:**, and **To:** fields are not displayed. To display these fields, use a (**T**)ype or (**P**)rint subcommand or the **top** subcommand.

> **Note:** In the example, the **From** line is displayed. This is not the same as the **From:** field that has been listed in the *FieldList* for the **ignore** subcommand.

Listing the ignored header fields

To get a list of the currently ignored header fields, at the mailbox prompt, type:

```
ignore
```

A list of all currently ignored headers is displayed. For example:

```
mail-from
message-id
return-path
```

Resetting the header fields

To reset the header fields, use the **retain** subcommand. For example:

```
retain date
```

Listing the retained header fields

To see which header fields are currently retained, enter the **retain** subcommand without a header field parameter.

Preventing the banner from displaying

The mail banner is the line at the top of the list of messages that shows the name of the mail program when you issue the **mail** command. It is similar to the following line:

```
Mail [5.2 UCB] [Workstation 3.1] Type ? for help.
```

To prevent the banner from displaying when you start the mail program, add the following line to your **$HOME/.mailrc** file:

```
set quiet
```

Another option that suppresses the **mail** banner is:

```
set noheader
```

With this option in the **.mailrc** file, the list of messages in your mailbox is not displayed. When you start the **mail** program, the only response is the mailbox prompt. You can get a list of messages by typing the (**h**)eader subcommand.

Combining the delete and print commands

After you read a message, you can delete it with the **d** subcommand. You can display the next message with the **p** subcommand. Combine these subcommands by typing the following line in your **.mailrc** file:

```
set autoprint
```

With the **set autoprint** option in the **.mailrc** file, the **d** subcommand deletes the current message and displays the next.

Creating default folders to store messages

Prerequisites

The mail program must be installed on your system.

Creating a letters mailbox directory to store messages in folders

1. To check if the **set folder** option has been enabled in the **.mailrc** file, at the mailbox prompt, type:

   ```
   set
   ```

   If the **set folder** option has been enabled, the system responds with the following:

   ```
   folder /home/george/letters
   ```

   In this example, **letters** is the directory in which mail folders will be stored.
2. If the **set folder** option has not been enabled, make a **set folder** entry in the **.mailrc** file:

   ```
   set folder=/home/george/letters
   ```

   In this example, **/home/george** is George's home directory and **letters** is the directory in which mail folders will be stored. The **set folder** option will allow you to use the plus sign (+) shorthand notation to save messages in your **letters** directory.
3. If a **letters** directory does not exist, you must create a **letters** directory in your home directory. From your home directory, at your system command line, type:

   ```
   mkdir letters
   ```

Keeping a record of messages you send to others

1. Type the following statement in your **.mailrc** file:

   ```
   set record=letters/mailout
   ```
2. If a **letters** directory does not exist, you must create a **letters** directory in your home directory. From your home directory, at your system command line, type:

   ```
   mkdir letters
   ```
3. To read copies of the messages you have sent to others, type:

   ```
   mail -f +mailout
   ```

   In this example, the file **mailout** contains copies of the messages you have sent to others.

Changing text editors for typing messages

Prerequisites

The mail program must be installed on your system.

set EDITOR=*PathName*

This option in your **.mailrc** file defines the editor that you activate with the **~e** key sequence. The value of *PathName* must be the full path name to the editor program you want to use.

To change to the e editor, while in the mail program, type:

~e

This sequence activates the e editor or other editor that you have defined in the **.mailrc** file. Edit your mail message using this editor.

set VISUAL=*PathName*

This option in your **.mailrc** file defines the editor that you activate with the **~v** key sequence. The value of *PathName* must be the full path name to the editor program that you want to use. The default is **/usr/bin/vi**.

To change to the vi editor while in the mail program, type:

~v

This sequence activates the vi editor or other editor that you have defined in the **.mailrc** file. Edit your mail message using this editor.

Mail command and subcommand summary

The following gives you a summary of the mail command and subcommands available for your use:

- "Summary of system commands to execute mail"
- "Summary of mailbox subcommands in the mail program"
- "Summary of mail editor subcommands" on page 38
- "Summary of secret mail subcommands" on page 39

Summary of system commands to execute mail

The following information summarizes the system commands to execute mail:

mail	Displays the system mailbox.
mail -f	Displays your personal mailbox (**mbox**).
mail -f +*folder*	Displays a mail folder.
mail *user@address*	Addresses a message to the specified user.

Summary of mailbox subcommands in the mail program

When the mail program is processing a mailbox, it displays the mailbox prompt to indicate that it is waiting for input. The mailbox prompt is an ampersand (&) displayed at the beginning of a new line. At the prompt, you can enter any of the mailbox subcommands.

Control subcommands

q	Quits and applies the mailbox subcommands entered this session.
x	Quits and restores the mailbox to original state.
!	Starts a shell, runs a command, and returns to the mailbox.
cd *dir*	Changes directory to **dir** or **$HOME**.

Display subcommands

t	Displays the messages in *msg_list* or the current message.
n	Displays the next message.
f *msg_list*	Displays the headings of messages in *msg_list* or of the current message if *msg_list* is not provided.
h *num*	Displays the headings of groups containing message *num*.
top *num*	Displays a partial message.
set	Displays a list of all enabled **.mailrc** options.
ignore	Displays a list of all the ignored header fields.
folder	Displays the number of messages in the current folder along with the path name of the folder.

Message handling

e *num*	Edits the message *num* (default editor is e).
d *msg_list*	Deletes the messages in *msg_list* or the current message.
u *msg_list*	Recalls deleted messages in *msg_list*.
s *msg_list* **+***file*	Appends messages (with headings) to *file*.
w *msg_list* **+***file*	Appends messages (text only) to *file*.
pre *msg_list*	Keeps messages in the system mailbox.

Creating new mail

m *addrlist*	Creates and sends a new message to the addresses in *addrlist*.
r *msg_list*	Sends a reply to senders and recipients of messages.
R *msg_list*	Sends a reply only to senders of messages.
a	Displays a list of aliases and their addresses.

Summary of mail editor subcommands

When the mail editor is processed, it displays the mail editor prompt to indicate that it is waiting for input. At the prompt, you can enter any of the mail editor subcommands.

Control subcommands

~q	Quits the editor without saving or sending the current message.
~p	Displays the contents of the message buffer.
~: *mcmd*	Runs a mailbox subcommand (*mcmd*).
EOT	Sends message (Ctrl-D on many terminals).
.	Sends the current message.

Add to heading subcommands

~h	Adds to the **To:**, **Subject:**, **Cc:**, and **Bcc:** fields.
~t *addrlist*	Adds the user addresses in *addrlist* to the **To:** field.
~s *subject*	Sets the **Subject:** field to the string specified by *subject*.
~c *addrlist*	Adds the user addresses in *addrlist* to the **Cc:** (carbon copy) field.
~b *addrlist*	Adds the user addresses in *addrlist* to the `Bcc:` (blind carbon copy) field.

Add to message subcommands

~d	Appends the contents of **dead.letter** to the message.
~r *filename*	Appends the contents of *filename* to the message.
~f *numlist*	Appends the contents of message numbers *numlist*.
~m *numlist*	Appends and indents the contents of message numbers *numlist*.

Change message subcommands

~e	Edits the message using the e editor (default is e).
~v	Edits the message using the vi editor (default is vi).
~w *filename*	Writes the message to *filename*.
~! *command*	Starts a shell, runs *command*, and returns to the editor.
~| *command*	Pipes the message to standard input of *command* and replaces the message with the standard output from that command.

Summary of secret mail subcommands

When the secret mail program processes a secret mailbox, it displays the secret mailbox prompt to indicate that it is waiting for input. The secret mailbox prompt is a question mark (?) displayed at the beginning of a new line. At the prompt, you can enter any of the secret mailbox subcommands.

System commands to send secret mail

xsend *barbara*	Addresses a message to the specified user.
xget	Displays the secret mailbox.

Secret mailbox subcommands

q	Quits, leaving unread messages.
n	Deletes the current message and displays the next message.
d	Deletes the current message and displays the next message.
Return key	Deletes the current message and displays the next message.
!	Executes a shell command.
s	Saves the message in the named file or **mbox**.
w	Saves the message in the named file or **mbox**.

Chapter 3. Transmission Control Protocol/Internet Protocol (TCP/IP) Overview

When computers communicate with one another, certain rules, or *protocols*, allow them to transmit and receive data in an orderly fashion. Throughout the world, one of the most routinely used sets of protocols is the Transmission Control Protocol/Internet Protocol (TCP/IP). (Much of Europe, however, uses the X.25 protocol.) Some common functions for using TCP/IP are electronic mail, computer-to-computer file transfer, and remote login.

The **mail** user command, the Message Handling (MH) user commands, and the **sendmail** server command can use TCP/IP for sending and receiving mail between systems, and the Basic Networking Utilities (BNU) can use TCP/IP for sending and receiving files and commands between systems.

TCP/IP is a suite of protocols that specify communications standards between computers and detail conventions for routing and interconnecting networks. It is used extensively on the Internet and consequently allows research institutions, colleges and universities, government, and industry to communicate with each other.

TCP/IP allows communication between a number of computers (called hosts) connected on a network. Each network can be connected to another network to communicate with hosts on that network. Although there are many types of network technologies, many of which operate with packet-switching and stream transport, TCP/IP offers one major advantage: hardware independence.

Because Internet protocols define the unit of transmission and specify how to send it, TCP/IP can hide the details of network hardware, allowing many types of network technologies to connect and exchange information. Internet addresses allow any machine on the network to communicate with any other machine on the network. TCP/IP also provides standards for many of the communications services that users need.

TCP/IP provides facilities that make the computer system an Internet host, which can attach to a network and communicate with other Internet hosts. TCP/IP includes commands and facilities that allow you to:

- Transfer files between systems
- Log in to remote systems
- Run commands on remote systems
- Print files on remote systems
- Send electronic mail to remote users
- Converse interactively with remote users
- Manage a network

> **Note:** TCP/IP provides basic network management capability. The Simple Network Management Protocol (SNMP) provides more network management commands and functions.

Topics discussed in this chapter are:

Understanding TCP/IP terminology

Before continuing, you may find it useful to become familiar with the following Internet terms as they are used in this book:

client A computer or process that accesses the data, services, or resources of another computer or process on the network.

host A computer that is attached to an Internet network and can communicate with other Internet hosts. The *local host* for a particular user is the computer at which that user is working. A *foreign host* is any other host name on the network. From the point of view of the communications network, hosts are both the source and destination of packets. Any host can be a client, a server, or both. On an Internet network, a host is identified by its Internet name and address.

network The combination of two or more hosts and the connecting links between them. A *physical network* is the hardware that makes up the network. A *logical network* is the abstract organization overlaid on all or part of one or more physical networks. The Internet network is an example of a logical network. The interface program handles translation of logical network operations into physical network operations.

packet The block of control information and data for one transaction between a host and its network. Packets are the exchange medium used by processes to send and receive data through Internet networks. A packet is sent from a *source* to a *destination*.

port A logical connecting point for a process. Data is transmitted between processes through ports (or *sockets*). Each port provides queues for sending and receiving data. In an interface program network, each port has an Internet *port number* based on how it is being used. A particular port is identified with an Internet *socket address*, which is the combination of an Internet host address and a port number.

process A program that is running. A process is the active element in a computer. Terminals, files, and other I/O devices communicate with each other through processes. Thus, network communications is *interprocess communication*s (that is, communication between processes).

protocol A set of rules for handling communications at the physical or logical level. Protocols often use other protocols to provide services. For example, a *connection-level protocol* uses a *transport-level protocol* to transport packets that maintain a connection between two hosts.

server A computer or process that provides data, services, or resources that can be accessed by other computers or processes on the network.

Communicating with other systems and users

There are several methods of communicating with other systems and users. Two methods are discussed here. The first method is by connecting a local host to a remote host. The second method is by conversing with a remote user.

- “Connecting a local host to a remote host”
- “Conversing with a remote user” on page 45

Connecting a local host to a remote host

There are several reasons why you might need to access a computer other than your own. For example, your system administrator might need to reassign permissions to a sensitive file you have been working on, or you might need to access a personal file from someone else’s workstation. You can even connect to your own computer from someone else’s computer station. Remote login functions, such as the **rlogin**, **rexec**, and **telnet** commands, enable the local host to perform as an input/output terminal host. Key strokes are sent to the remote host, and the results are displayed on the local monitor. When you end the remote login session, all functions return to your local host.

TCP/IP contains the following commands for remote login and command execution:

rexec

The **rexec** command makes it possible to execute commands interactively on different foreign hosts when you log in to a remote host with the **rlogin** command. This command is disabled by the system manager if extra security is needed for your network. When you issue the **rexec** command, your local host searches the **$HOME/.netrc** file of the remote host for your user name and a password from your local host. If these are found, the command you requested to be run on the local host will then be run. Otherwise, you will be required to supply a login name and password before the request can be honored.

rlogin

The **rlogin** command makes it possible to log in to similar foreign hosts. Unlike **telnet**, which can be used with different remote hosts, the **rlogin** command can be used on UNIX hosts only. This command is disabled by the system manager if extra security is needed for your network.

The **rlogin** command is similar to the **telnet** command in that both allow a local host to connect to a remote host. The only difference is that the **rlogin** command is not a trusted command and can be disabled if your system needs extra security.

The **rlogin** command is not a trusted command because both the **$HOME/.rhosts** file, which is owned by the local user, and the **/etc/hosts.equiv** file, which is owned by your system manager, keep a listing of remote hosts that have access to the local host. Therefore, if you leave your terminal on while unattended, an unauthorized user could examine the names and passwords contained in these files, or worse, could damage a remote host in some way. Ideally, remote users should be required to type a password after issuing the **rlogin** command, but it is quite possible to bypass this recommended feature.

If neither the **$HOME/.rhosts** file nor the **/etc/hosts.equiv** file contains the name of a remote host that is trying to log in, the local host prompts for a password. The remote password file is first checked to verify the password entered; the login prompt is again displayed if the password is not correct. Pressing tilde and period (~.) at the login prompt ends the remote login attempt.

The **rlogin** command can also be configured to use Kerberos V.5 to authenticate the user. This option allows the user to be identified without the use of a **$HOME/.rhosts** file or passing the password across the network. For more information about this use of the **rlogin** command, see “Understanding the secure rcmds” on page 53.

rsh and **remsh**

The **rsh** and **remsh** commands make it possible to execute commands on similar foreign hosts. All required input must be performed by the remote host. The **rsh** and **remsh** commands are disabled by the system manager if extra security is needed for your network.

The **rsh** command can be used in two ways:

- To execute a single command on a remote host when a command name is specified
- To execute the **rlogin** command when no command name is specified

When the **rsh** command is issued, your local host searches the **/etc/hosts.equiv** file on the remote host for permission to log in. If that is unsuccessful, the **$HOME/.rhosts** file is searched. Both of these files are lists of remote hosts having login permission. Remote users should be required to type a password after issuing the **rsh** command.

It is also possible to eliminate the need to issue the **rlogin** command. The **rsh** command permits the execution of commands on a remote host, but does not provide a means of bypassing the password requirement. If a password is needed to access a remote host, then a password is needed to use the **rsh** command as well because both commands access the **$HOME/.rhosts** and **/etc/hosts.equiv** files.

The **rsh** command can also be configured to use Kerberos V.5 to authenticate the user. This option allows the user to be identified without either the use of a **$HOME/.rhosts** file or passing the password across the network. For more information about this use of the **rsh** command, see “Understanding the secure rcmds” on page 53.

telnet, **tn**, and **tn3270**

The **telnet** command is a terminal emulation program that implements the TELNET protocol and allows you to log in on a similar or dissimilar foreign host. It uses TCP/IP to communicate with other hosts in the network.

Note: For convenience, **telnet** hereafter refers to the **telnet**, **tn**, and **tn3270** commands.

The **telnet** command is one way a user can log in to a remote host. The most important feature of the **telnet** command is that it is a *trusted* command. In contrast, the **rlogin** command, which also allows for remote login, is not considered a trusted command.

A system may need extra security to prevent unauthorized users from gaining access to its files, stealing sensitive data, deleting files, or placing viruses or worms on the system. The security features of TCP/IP are designed to help prevent these occurrences.

A user who wishes to log in to a remote host with the **telnet** command must provide the user name and password of an approved user for that computer. This is similar to the procedure used for logging in to a local host. When successfully logged in to a remote host, the user's terminal runs as if directly connected to the host.

The **telnet** command supports an option called *terminal negotiation*. If the remote host supports terminal negotiation, the **telnet** command sends the local terminal type to the remote host. If the remote host does not accept the local terminal type, the **telnet** command attempts to emulate a 3270 terminal and a DEC VT100 terminal. If you specify a terminal to emulate, the **telnet** command does not negotiate for terminal type. If the local and remote hosts cannot agree on a terminal type, the local host defaults to **none**.

The **telnet** command supports these 3270 terminal types: 3277-1, 3278-1, 3278-2, 3278-3, 3278-4, and 3278-5. If you are using the **telnet** command in 3270 mode on a color display, the colors and fields are displayed just as those on a 3279 display, by default. You can select other colors by editing one of the keyboard mapping files in the preceding list of terminal types. When the **telnet** session has ended, the display is reset to the colors in use before the session began.

The **telnet** command can also be configured to use Kerberos V.5 to authenticate the user. This option allows the user to be identified without either the use of a **$HOME/.rhosts** file or passing the password across the network. For more infomation about this use of the **telnet** command, see "Understanding the secure rcmds" on page 53.

Note: The **rsh** and **rexec** commands can be used to execute commands on a remote host, but neither is a trusted command, so they may not meet all security levels configured into your computer. As a result, these commands can be disabled if your system requires extra security.

Logging in to a remote host

You can log in to a remote host by using the **telnet** command. In order to do this, you must have a valid user ID and password for the remote host.

To log in to a remote host (`host1` in this example), type:

```
telnet host1
```

Information similar to the following displays on your screen:

```
Trying . . .
Connected to host1
Escape character is '^T'.

AIX telnet (host1)

AIX Operating System
Version 4.1
(/dev/pts0)
login:_
```

After you have logged in, you can issue commands. To log out of the system and close the connection, press the Ctrl-D key sequence.

If you cannot log in, cancel the connection by pressing the Ctrl-T key sequence.

Conversing with a remote user

Use the **talk** command to have a real-time conversation with another user on a remote host. The **talk** command requires a valid address on which to bind. The host name of the remote terminal must be bound to a working network interface that is usable by other network commands, such as the **ping** command. If a machine has no network interface that is a standalone terminal, it must bind its host name to the loopback address (127.0.0.1) in order for the **talk** command to work.

Using electronic mail, you can send text messages to other users on a local network and receive mail from them as well. If a computer system is configured appropriately and you know the appropriate electronic address, you can send electronic mail messages across the world to someone on a remote system.

TCP/IP contains the following commands for remote communications:

mail Sends and receives electronic memos and letters
talk Lets you have an interactive conversation with a user on a remote host

Prerequisites

1. The **talkd** daemon must be active on both the local and remote host.
2. The user on the remote host must be logged in.

Procedure

1. To talk to the remote user `dale@host2` logged in on a remote host, `jane@host1` types:

   ```
   talk dale@host2
   ```

   A message similar to the following displays on the screen of `dale@host2`:

   ```
   Message from TalkDaemon@host1 at 15:16...
   talk: connection requested by jane@host1.
   talk: respond with: talk jane@host1
   ```

   This message informs `dale@host2` that `jane@host1` is trying to converse with her.
2. To accept the invitation, `dale@host2` types:

   ```
   talk jane@host1
   ```

   Users `dale@host2` and `jane@host1` are now able to have an interactive conversation.
3. To end a conversation at any time, either user can press the Ctrl-C key sequence. This returns them to the command line prompt.

Transferring files

Although it is possible to send relatively short files using electronic mail, there are more efficient ways of transferring large files. Electronic mail programs are usually designed to transmit relatively small amounts of text; therefore, other means are needed to transfer large files effectively. The **ftp**, **rcp**, and **tftp** commands rely on TCP/IP to establish direct connections from your local host to a remote host. Basic Network Utilities (BNU) can also use TCP/IP to provide direct connections with foreign hosts. The following sections tell you how to copy files from one host to another.

- “Copying files using the ftp and rcp commands” on page 46
- “Copying files using the tftp and utftp command” on page 48

Copying files using the ftp and rcp commands

Use the **ftp** command to copy a file from a remote host. The **ftp** command does not preserve file attributes or copy subdirectories. If either of these conditions are necessary, use the **rcp** command.

ftp Uses the File Transfer Protocol (FTP) to transfer files between hosts that use different file systems or character representations, such as EBCDIC and ASCII. It provides for security by sending passwords to the remote host and also permits automatic login, file transfers, and logoff.

rcp Copies one or more files between a local host and a remote host, between two separate remote hosts, or between files at the same remote host. This command is similar to the **cp** command except that it works only for remote file operations. If extra security is needed for your network, this command is disabled by the system manager.

Prerequisites

1. You must have remote login permission specified in the remote host's **$HOME/.netrc** file if the automatic login feature is to be used. Otherwise, you must know a login name and password for the remote host. For more information about the **.netrc** file, see "Creating the .netrc file" on page 52.

   Alternatively, the system can be configured to use Kerberos V.5 authentication. This is used instead of the **.netrc** or **$HOME/.rhosts** files. See "Understanding the secure rcmds" on page 53.
2. If you wish to copy a file from a remote host, you must have read permission for that file.

   Note: The read and write permissions for files and directories on a remote host are determined by the login name used.
3. If you wish to copy a file from your local host to a remote host, you must have write permission for the directory that is to contain the copied file. Also, if the directory on the remote host contains a file with the same name as the file that you wish to place in it, you must have write permission to append the file on the remote host.

Logging in to a remote host directly

1. Use the **cd** command to change to the directory that contains the file you want to send (sending a file) or to the directory where you want the transferred file to reside (receiving a file).
2. Log in to the remote host directly by typing:

   ftp *HostName*

   If you have automatic login permission, information similar to the following is displayed on your local host:

   ```
   Connected to canopus.austin.century.com.
   220 canopus.austin.century.com FTP server
   (Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
   331 Password required for dee.
   230 User dee logged in.
   ftp>
   ```

   Otherwise, information similar to the following is displayed on your local host:

   ```
   Connected to canopus.austin.century.com.
   220 canopus.austin.century.com FTP server
   (Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
   Name (canopus:eric): dee
   331 Password required for dee.
   Password:
   230 User dee logged in.
   ftp>
   ```
3. Enter your login name and password when prompted by the system.

   You are now ready to copy a file between two hosts.

Logging in to a remote host indirectly

1. Use the **cd** command to change to the directory that contains the file you want to send (sending a file) or to the directory where you want the transferred file to reside (receiving a file).

2. Log in to the remote host indirectly by typing:

   ```
   ftp
   ```

3. When the ftp> prompt is displayed, type:

   open *HostName*

   If you have automatic login permission, information similar to the following is displayed on your local host:

   ```
   Connected to canopus.austin.century.com.
   220 canopus.austin.century.com FTP server
   (Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
   331 Password required for dee.
   230 User dee logged in.
   ftp>
   ```

   Otherwise, information similar to the following is displayed on your local host:

   ```
   Connected to canopus.austin.century.com.
   220 canopus.austin.century.com FTP server
   (Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
   Name (canopus:eric): dee
   331 Password required for dee.
   Password:
   230 User dee logged in.
   ftp>
   ```

4. Enter your name and password when prompted by the system.

Copying a file from a remote host to a local host

To copy a file from a remote host to a local host using the **ftp** command, you must first log in to the remote system either directly or indirectly. See “Logging in to a remote host directly” on page 46 or “Logging in to a remote host indirectly” on page 46 for instructions.

Note: The **ftp** command uses the ASCII default transfer type to copy files.

To copy a file from a remote host to a local host:

1. Determine if the file that you want to copy is in the current directory by running the **dir** subcommand. (The **dir** subcommand for the **ftp** command works in the same way as the **ls -l** command.) If the file is not there, use the **cd** subcommand to move to the proper directory.
2. To copy your local file using binary image, type:

   ```
   binary
   ```

3. To copy a file to your host, type:

   ```
   get FileName
   ```

   The file is placed in the directory from which you issued the **ftp** command.
4. To end the session, press the Ctrl-D key sequence, or type `quit`.

Copying a file from a local host to a remote host

To copy a file from a local host to a remote host using the **ftp** command, you must first log in to the remote system either directly or indirectly. See “Logging in to a remote host directly” on page 46 or “Logging in to a remote host indirectly” on page 46 for instructions.

Note: The **ftp** command uses the ASCII default transfer type to copy files.

To copy a file from a local host to a remote host:

1. If you want to place the file in a directory other than the **$HOME** directory, use the **cd** subcommand to move to the desired directory.
2. To copy your local file using binary image, type:

   ```
   binary
   ```

3. To copy a file to the remote host, type:

```
put FileName
```

4. To end the session, press the Ctrl-D key sequence, or type **quit**.

Copying files using the tftp and utftp command

Use the **tftp** and **utftp** commands for the Trivial File Transfer Protocol (**TFTP**) to transfer files to and from hosts. Because **TFTP** is a single-file transfer protocol, the **tftp** and **utftp** commands do not provide all the features of the **ftp** command. If extra security is required for your network, the system manager can disable this command.

Note: The **tftp** command is not available when your host is operating at a high level of security.

Prerequisites

1. If you wish to copy a file *from* a remote host, you must have *read* permission for the directory that contains the desired file.
2. If you wish to copy a file *to* a remote host, you must have *write* permission for the directory in which the file is to be placed.

Copying a file from a remote host

1. To establish a connection to a remote host, type:

   ```
   tftp host1
   ```

   In this example, `host1` is the name of the host to which you wish to connect.

   The `tftp>` prompt is displayed.
2. To determine that a connection has been established, type:

   ```
   status
   ```

   A message similar to the following is displayed:

   ```
   Connected to host1
   Mode: netascii Verbose: off Tracing: off
   Remxt-interval: 5 seconds, Max-timeout: 25 seconds
   tftp>
   ```
3. Enter the **get** subcommand, the name of the file to be transferred, and the name to be assigned to the file on the remote system:

   ```
   get /home/alice/update update
   ```

   The **/home/alice** directory on the remote host must have *read* permission set for other users. In this example, the **/home/alice/update** file is transferred from `host1` to the **update** file in the current directory on the local system.
4. To end the session, type:

   quit

   or press the Ctrl-D key sequence.

Copying a file to a remote host

1. To establish a connection to a remote host, type:

   ```
   tftp host1
   ```

   In this example, `host1` is the name of the host to which you wish to connect.

   The `tftp>` prompt is displayed.
2. To determine that a connection has been established, type:

   ```
   status
   ```

   A message similar to the following is displayed:

   ```
   Connected to host1
   Mode: netascii Verbose: off Tracing: off
   Remxt-interval: 5 seconds, Max-timeout: 25 seconds
   tftp>
   ```

3. Enter the **put** subcommand, the name of the file to be transferred from the local host, and the path and file name for the file on the remote host:

   ```
   put myfile /home/alice/yourfile
   ```

   The **/home/alice** directory on the remote host must have *write* permission set for others. The **myfile** file, located in the user's current working directory, is transferred to `host1`. The path name must be specified unless a default has been established. The **myfile** file appears on the remote host as **yourfile**.
4. To end the session, type:

   ```
   quit
   ```

   or use the Ctrl-D key sequence.

Printing files

This section provides the following information:

- "Printing to a remote system"
- "Printing from a remote system" on page 50

Printing to a remote system

If you have a local printer attached to your host, then this section refers to printing to a remote printer. If you have no local printer, then this section refers to printing to a nondefault remote printer. You can use either the **enq** command or the System Management Interface Tool (SMIT) to complete this task.

> **Note:** This section explains how to print to a remote host at the simplest level possible. For more information and ideas about remote printing, see **enq** command.

Prerequisites

1. Your host name must appear in the **/etc/hosts.lpd** file of the remote host.

   > **Note:** The queuing system does not support multibyte host names.

   To implement changes to the **/etc/hosts.lpd** file without restarting the system, use the System Resource Controller (SRC) **refresh** command.
2. You must be able to determine the queue name and the remote printer name in your local **/usr/lib/lpd/qconfig** file.

Placing a print job in a remote print queue

In order for you to place a job in a remote printing queue, your host name must appear in the **/etc/hosts.lpd** file of the remote host (the queuing system does not support multibyte host names). To implement changes to the **/etc/hosts.lpd** file without restarting the system, use the System Resource Controller (SRC) **refresh** command. Also, you must be able to determine the queue name and the remote printer name in your local **/usr/lib/lpd/qconfig** file.

1. Find the appropriate queue name and remote device name. The queue name usually begins with the letters **rp** followed by a numeral or set of numerals. The remote printer name usually begins with the letters **drp** followed by a numeral or set of numerals.
2. Enter the following command:

   enq -P *QueueName*:*PrinterName FileName*

   where *QueueName* is the name of the queue (such as **rp1**), and *PrinterName* is the name of the printer (such as **drp1**) as found in the **/usr/lib/lpd/qconfig** file. Do not omit the colon (**:**) between the *QueueName* and the *PrinterName*. *FileName* is the name of the file that you wish to print.

The following are examples of how the **enq** command can be used:

- To print the file **memo** on the default printer, type:

  ```
  enq memo
  ```

- To print the file **prog.c** with page numbers, type:

  ```
  pr prog.c | enq
  ```

  The **pr** command puts a heading at the top of each page that includes the date the file was last modified, the name of the file, and the page number. The **enq** command then prints the file.
- To print the file `report` on the next available printer configured for the `fred` queue, type:

  ```
  enq -P fred report
  ```
- To print several files beginning with the prefix `sam` on the next available printer configured for the `fred` queue, type:

  ```
  enq -P fred sam*
  ```

  All files beginning with the prefix `sam` are included in one print job. Normal status commands show only the title of the print job, which in this case is the name of the first file in the queue unless a different value was specified with the **-T** flag. To list the names of all the files in the print job, use the long status command **enq -A -L**.

Enqueuing a job using SMIT

1. To enqueue a job using SMIT, type the following command:

   ```
   smit
   ```
2. Select the **Spooler**, and start a print job menu.
3. Select the **File to Print** option, and type the name of the file you wish to print.
4. Select the **Print Queue** option, and select the name of the remote printer to which you wish to print. You are now ready to print to a remote printer.

Printing from a remote system

Occasionally, you might need to print a file that is located on a remote host. The location of the printout depends upon which remote printers are available to the remote host.

> **Note:** This section explains how to print to a remote host at the simplest level possible. For more information and ideas about remote printing, read about the **enq** command.

Prerequisites

1. You must be able to log in to the remote system using the **rlogin** or **telnet** command.
2. You must have read permission for the remote file that you wish to print on your local printer.

Procedure

To print from a remote system:

1. Log in to the remote system using the **rlogin** or **telnet** command.
2. Find the appropriate queue name and remote device name. The queue name usually begins with the letters **rp** followed by a numeral or set of numerals. The remote printer name usually begins with the letters **drp** followed by a numeral or set of numerals.
3. Enter the following command:

   enq -P *QueueName*:*PrinterName FileName*

   where *QueueName* is the name of the queue (such as **rp1**), and *PrinterName* is the name of the printer (such as **drp1**) as found in the **/usr/lib/lpd/qconfig** file. Do not omit the : (colon) between the *QueueName* and the *PrinterName*. *FileName* is the name of the file that you want to print.
4. End the connection to the remote host by pressing the Ctrl-D sequence or by typing **quit**.

Finding information about hosts and users

You can use TCP/IP commands to determine the status of a network, display information about a user, and resolve host information needed to communicate to another host or user.

Status commands

TCP/IP contains the following commands to determine the status of local and remote hosts and their networks:

finger or **f**	Displays information about the current users on a specified host. This information can include the user's login name, full name, and terminal name, as well as the date and time of login.
host	Resolves a host name into an Internet address or an Internet address into a host name.
ping	Helps determine the status of a network or host. It is most commonly used to verify that a network or host is currently running.
rwho	Shows which users are logged in to hosts on a local network. This command displays the user name, host name, and date and time of login for everyone on the local network.
whois	Identifies to whom a user ID or nickname belongs. This command can only be used if your local network is connected to the Internet.

Displaying information about all users logged in to a host

To display information about all users logged in to a remote host:

1. Log in to the remote host with which you want to communicate.
2. To display information about all users logged in to host `alcatraz`, type:

   ```
   finger @alcatraz
   ```

   Information similar to the following is displayed:

   ```
   brown    console  Mar 15 13:19
   smith    pts0     Mar 15 13:01
   jones    tty0    Mar 15 13:01
   ```

User `brown` is logged in at the console, user `smith` is logged in from a pseudo teletype line pts0, and user `jones` is logged in from a tty0. Your system administrator can set up your system so that the **finger** command works differently. If you encounter any problems using the **finger** command, contact your system administrator.

Displaying information about a user logged in to a host

To display information about a user logged in to a remote host:

1. Log in to the remote host with which you want to communicate.
2. To display information about user `brown` on host `alcatraz`, type:

   ```
   finger brown@alcatraz
   ```

   Information similar to the following is displayed:

   ```
   Login name:  brown
   Directory:  /home/brown     Shell:  /home/bin/xinit -L -n Startup
   On since May 8 07:13:49 on console
   No Plan.
   ```

Your system administrator can set up your system so that the **finger** command works differently. If you encounter any problems using the **finger** command, contact your system administrator.

Customizing TCP/IP features

This section provides information about the following:

Creating the .netrc file

The **.netrc** file specifies automatic login information for the **ftp** and **rexec** commands. The following steps describe how to create and edit the **$HOME/.netrc** file:

Prerequisites

1. You must have a copy of the **/usr/lpp/tcpip/samples/netrc** file.
2. The **securetcpip** command must not be running on your system.

Procedure

To create the .netrc file:

1. Copy the **/usr/lpp/tcpip/samples/netrc** file to your **$HOME** directory by typing the following command:

   ```
   cp /usr/lpp/tcpip/samples/netrc $HOME
   ```

2. Edit the **$HOME/netrc** file to supply the appropriate *HostName*, *LoginName*, and *Password* variables. For example:

   ```
   machine host1.austin.century.com login fred password bluebonnet
   ```

3. Set the permissions on the **$HOME/netrc** file to 600 by using the **chmod** command. At the command line prompt ($), type:

   ```
   chmod 600 $HOME/.netrc
   ```

4. Rename the **$HOME/netrc** file to **$HOME/.netrc** file. The initial period (.) causes the file to be hidden.

   ```
   mv $HOME/netrc $HOME/.netrc
   ```

The **$HOME/.netrc** file can contain multiple login definitions and up to 16 macros per login definition.

Writing ftp macros

ftp macros are defined in the **$HOME/.netrc** file. The following steps describe how to create an **ftp** macro.

Prerequisites

You must have created the **$HOME/.netrc** file.

Procedure

To write an ftp macro:

1. Edit the **$HOME/.netrc** file to include the following instructions:

   ```
   macdef init
   put schedule
   ```

   Be sure to insert a blank line at the end of your **ftp** macro. The blank line terminates the **ftp** macro. In the above example, the **macdef** subcommand defines the subcommand macro `init`. The line following is the command the macro specifies, in this case `put schedule`, where **`schedule`** is the name of a file.

2. After you create the **ftp** macro, at the command line prompt, type:

   ```
   ftp hostname
   ```

   Where *hostname* is the name of the host to which you are connecting. **ftp** scans the **$HOME/.netrc** file for a login definition matching your host name and uses that login definition to log you in.

3. After you log in, at the command line prompt, type:

   ```
   ftp init
   ```

   In this example, **ftp** scans for the macro named **`init`** and executes the command or commands the macro specifies.

   An **ftp** macro is associated with the login entry immediately preceding it. **ftp** macros are not global to the **$HOME/.netrc** file. The macro `init` is executed automatically upon login. Other macros can be executed from the **ftp** prompt (`ftp>`) by typing the following:

   ```
   $getit
   ```

   In this example, the **$** executes the **ftp** macro **getit.**

Changing the assignment of a key set

The following steps describe how to create and edit the **$HOME/.3270keys** file to customize key functions or sequences:

Prerequisites

1. You must have working knowledge of the vi editor.
2. The vi editor must be on your system.

Procedure

1. Copy the **/etc/3270.keys** file to the **$HOME** directory and rename it **.3270keys** using the following command:

   ```
   cp /etc/3270.keys $HOME/.3270keys
   ```

2. Change the bind statements in the **$HOME/.3270keys** file to change the assignment of a key set using the following steps:
   a. Start the vi editor on a new file and enter insert mode.
   b. Press the Ctrl-V key sequence and then the key that you want to map. This displays a value for the pressed key.
   c. Place the displayed value on the appropriate line in the `Sequence` column of the **$HOME/.3270keys** file.

   For example, after you have invoked the vi editor and entered insert mode, press Ctrl-V and then Alt-Insert. This displays `[[141q`. The first `[` is replaced with `\e` in the `Sequence` column so that the configured line looks like the following:

   ```
   3270 Function Sequence  Key
   bind  pa1      "\e[141q" #a_insert
   ```

Using a .k5login file

The **.k5login** file is used when Kerberos V.5 authentication is used for the secure rcmds. This file specifies which DCE principals on which cells are allowed access to the user's account. The file is located at **$HOME/.k5login**. It should be owned by the local user and the owner should have *read* permission on this file. The minimum permission setting for this file is 400.

The **.k5login** file contains a list of the DCE principal/cell pairs allowed to access the account. The principal/cell pairs are kept in Kerberos format (as opposed to DCE format). For example, if the file contains

```
UserA@Cell1
```

then the DCE principal `UserA` on the DCE cell `Cell1` can access the account.

If the DCE principal is the same as the user's account name, and if there is no **$HOME/.k5login** file for the user's account, the DCE principal gains access to the account (provided Kerberos V.5 authentication is configured).

For more information about Kerberos V.5 authentication, see "Understanding the secure rcmds."

Understanding the secure rcmds

The secure rcmds are **rlogin**, **rcp**, **rsh**, **telnet**, and **ftp**. These commands have been enhanced to provide additional authentication methods to those used today. They are known collectively by the term, *Standard AIX* method. (This method refers to the authentication method used by AIX 4.3 and prior releases.) The two additional methods are Kerberos V.5 and Kerberos V.4.

When using the Kerberos V.5 authentication method, the client gets a Kerberos V.5 ticket from the DCE security server or Native Kerberos server. The ticket is a portion of the user's current DCE or Native credentials encrypted for the TCP/IP server with which they wish to connect. The daemon on the TCP/IP

server decrypts the ticket. This allows the TCP/IP server to absolutely identify the user. If the DCE or Native principal described in the ticket is allowed access to the operating system user's account, the connection proceeds.

> **Note:** Beginning with DCE version 2.2, the DCE security server can return Kerberos V.5 tickets. Beginning with AIX version 5.2, the secure rcmds will use the Kerberos V.5 library and the GSSAPI library provided by NAS (Network Authentication Service) version 1.3.

In addition to authenticating the client, Kerberos V.5 forwards the current user's credentials to the TCP/IP server. If the credentials are marked forwardable, the client sends them to the server as a Kerberos TGT (Ticket Granting Ticket). On the TCP/IP server side, if one is communicating with a DCE security server, the daemon upgrades the TGT into full DCE credentials using the **k5dcecreds** command.

The **ftp** command uses a different authentication method than the other commands. It uses the GSSAPI security mechanism to pass the authentication between the **ftp** command and the **ftpd** daemon. Using the **clear**/**safe**/**private** subcommands, the **ftp** client supports data encryption.

Between operating system clients and servers, **ftp** has been enhanced to allow multiple-byte transfers for encrypted data connections. The standards define only single-byte transfers for encrypted data connections. When connected to third-party machines and using data encryption, **ftp** follows the single-byte transfer limit.

System configuration

For all of the secure rcmds, there is a system-level configuration mechanism to determine which authentication methods are allowed for that system. The configuration controls both outgoing and incoming connections.

The authentication configuration consists of a library, **libauthm.a**, and two commands, **lsauthent** and **chauthent**, that provide command line access to the library's two routines: **get_auth_methods** and **set_auth_methods**.

The system supports three different authentication methods: Kerberos V.5, Kerberos V.4, and *Standard AIX*. The authentication method defines which method is used to authenticate a user across a network.

- Kerberos V.5 is the most common method, as it is the basis for the Distributed Computing Environment (DCE). The operating system either upgrades incoming Kerberos V.5 tickets to full DCE credentials or uses incoming Native Kerberos V.5 tickets.
- Kerberos V.4 is used by only two of the secure rcmds: **rsh** and **rcp**. It is provided to support backward compatibility on SP™ systems and will only be functional on one. A Kerberos V.4 ticket is not upgraded to DCE credentials.
- The term, *Standard AIX* authentication method as previously mentioned, refers to the authentication method that is used by AIX 4.3 and prior releases.

There is a fallback implementation when more than one authentication method is configured. If the first method fails to connect, the client attempts to authenticate using the next authentication method configured.

Authentication methods can be configured in any order. The only exception is that *Standard AIX* must be the final authentication method configured, because there is no fallback option from it. If *Standard AIX* is not a configured authentication method, password authentication is not attempted, and any connection attempt using this method is rejected.

It is possible to configure the system without any authentication methods. In this case, the system refuses all connections from and to any terminal using secure rcmds. Also, because Kerberos V.4 is supported only with the **rsh** and **rcp** commands, a system configured to use only Kerberos V.4 will not allow connections using **telnet**, **ftp**, or **rlogin**.

For more information, see the **get_auth_method** and **set_auth_method** subroutines in *AIX 5L Version 5.3 Technical Reference: Communications Volume 2*, the **lsauthent** command in *AIX 5L Version 5.3 Commands Reference, Volume 3*, and the **chauthent** command in *AIX 5L Version 5.3 Commands Reference, Volume 1*.

Kerberos V.5 user validation

When using the Kerberos V.5 authentication method, the TCP/IP client gets a service ticket encrypted for the TCP/IP server. When the server decrypts the ticket, it has a secure method of identifying the user (by DCE or Native principal). However, it still needs to determine if this DCE or Native principal is allowed access to the local account. Mapping the DCE or Native principal to the local operating system account is handled by a shared library, **libvaliduser.a**, which has a single subroutine **kvalid_user**. If a different method of mapping is preferred, the system administrator must provide an alternative for the **libvaliduser.a** library.

DCE configuration

To use the secure rcmds, two DCE principals must exist for every network interface to which they can be connected. They are:

```
host/FullInterfaceName
ftp/FullInterfaceName
```

where *FullInterfaceName* is the interface name and domain name for the primary *HostName.DomainName*.

Native configuration

To use the secure rcmds, two principles must exist for every network interface to which they can be connected. They are:

```
host/FullInterfaceName@Realmname
ftp/FullInterfaceName@Realmname
```

where *FullInterfaceName* is the interface name and the domain name for the primary HostName.DomainName. RealmName is the name of the Native Kerberos V realm.

Summary of TCP/IP commands

TCP/IP is part of the underlying structure of your system. It allows you to communicate with another terminal or system merely by executing a command or program. Your system takes care of the rest.

The TCP/IP commands for users can be grouped into the following categories:

- "File transfer commands"
- "Remote login commands"
- "Status commands" on page 56
- "Remote communication commands" on page 56
- "Print commands" on page 56

File transfer commands

ftp *hostname*	Transfers files between a local and a remote host.
rcp *file host:file*	Transfers files between local and remote host or between two remote hosts.
tftp	Transfers files between hosts.

Remote login commands

rexec *host command*	Executes commands one at a time on a remote host.

rlogin *remotehost*	Connects a local host with a remote host.
rsh and **remsh** *remotehost command*	Executes specified command at remote host or logs into the remote host.
telnet, **tn** and **tn3270** *hostname*	Connects the local host with a remote host, using the **TELNET** interface.

Status commands

finger or **f** *user@host*	Shows user information.
host *hostname*	Resolves a host name into an Internet address or an Internet address into a host name.
ping *hostname*	Sends an echo request to a network host.
rwho	Shows which users are logged in to hosts on the local network.
whois *name*	Identifies a user by user ID or alias.

Remote communication commands

talk *User@Host*	Converse with another user.

Print commands

enq *file*	Enqueues a file.
refresh	Requests a refresh of a subsystem or group of subsystems.
smit	Performs system management.

Chapter 4. Basic Networking Utilities (BNU) Overview

The Basic Networking Utilities (BNU) establish communications between computer systems on local and remote networks. BNU is one of the Extended Services® programs that can be installed with the Base Operating System.

BNU contains a group of commands related to the UNIX-to-UNIX Copy Program (UUCP) developed by AT&T and modified as part of the Berkeley Software Distribution (BSD).

BNU provides commands, processes, and a supporting database for connections to local and remote systems. Communication networks such as token-ring and Ethernet are used to connect systems on local networks. A local network can be connected to a remote system by hardwire or telephone modem. Commands and files can then be exchanged between the local network and the remote system.

Topics discussed in this overview are:

BNU path names

Path names used with Basic Networking Utilities (BNU) commands can be entered in a number of different ways. Path names consist of either the root directory or a shortcut path to the target, which is the name of a remote system or systems. Each path variation follows specific guidelines.

Full path name

A full path name starts at the root and traces all the directories down to the target directory and file. For example, **/etc/uucp/Devices** refers to the **Devices** file in the **uucp** directory in the root directory **etc**.

Always type a preceding forward slash (/) to signify a root directory. Always separate elements in paths with a forward slash (/).

Relative path name

The relative path name lists only those directories that are relative to the current directory.

For example, if your current directory is **/usr/bin** and your target directory is **/usr/bin/reports**, type the relative path name **reports** (without the leading slash).

Relative path names can be used with the **cu**, **uucp**, and **uux** commands, and with the name of the source file in the **uuto** command.

> **Note:** Relative path names might not work with all BNU commands. If you have trouble using a relative path name, type the command again with the full path name.

~ [option] path name

The **~ [option]** path name represents the home directory of the specified user. The tilde (~) can be used as a shortcut to certain directories.

For example, **~jane** refers to the home directory of the user `jane`. The entry, **~uucp** or **~** (tilde alone) refers to the BNU public directory on the remote system. The full path name for the BNU public directory is **/var/spool/uucppublic**.

> **Note:** This use of the tilde should not be confused with the other use of the tilde in BNU. The tilde is also used to preface commands for execution on a local system when logged in to a remote system when using the **cu** command.

system_name! path name

The *system_name***!** path name identifies the path to a file on another system. For example, **distant!/account/march** refers to the **march** file in the **account** directory on the remote system **distant**.

system_name!system_name! path name

The *system_name***!***system_name***!** path name identifies a path through multiple systems. For example, if the system named **distant** can only be reached through another system called **near**, then the path name is **near!distant!/account/march**.

Separate system names with an exclamation mark (!). In the case of multiple-system path names, the rule of separating elements with a forward slash (/) does not apply to the system names. However, the rule does hold for the termination system, where directories and files are stipulated.

> **Note:** When you use a bourne shell, separate system names with an exclamation mark (!). When you use BNU in either a C or korn shell, precede the exclamation mark with a backward slash (\). The backward slash is an escape character necessary to interpret the next character literally rather than as the special character.

Communicating between local and remote systems

For Basic Networking Utilities (BNU) to establish communication between a local system and a remote system, the remote system must have:

- Hardwire or modem link to the local system
- UNIX-based operating system installed
- Either BNU or another version of UNIX-to-UNIX Copy Program (UUCP) running

> **Note:** You can use BNU to communicate to a non-UNIX system, but such connections might require additional hardware or software.

BNU has two commands that enable you to communicate with remote systems. The **cu** command connects systems over either hardwire or telephone lines. The **ct** command connects systems over telephone lines only, using a modem.

Use the **cu** command to establish communication between networks when you know either the phone number or the name of the target system. To use the **ct** command, you *must* have the phone number of the target system.

> **Note:** A third command, **tip**, functions very much like the **cu** command. However, the **tip** command is a component of the Berkeley Software Distribution (BSD) version of the UUCP program. Its installation with BNU requires special configuration.

Communicating with another system by hardwire or modem

Use the **cu** command from your local system to:

- Establish a connection to a specified remote system
- Log in to the remote system
- Perform tasks on the remote system
- Switch back and forth, working concurrently on both systems

If the remote system is running under the same operating system, you can issue regular commands from your local system. For example, you can issue commands to change directories, list directory contents, view files, or send files to the print queue on the remote system. To issue commands for use on your local system, or to initiate remote command and file exchanges, use special **cu** local commands, prefaced with a tilde (**~**).

Communicating with another system by modem

Issue the **ct** command to communicate by modem with another system. Enter the **ct** command, followed by a phone number, to call the remote modem. When the connection is made, the remote login prompt is displayed on your screen.

The **ct** command can be useful under certain conditions. For details on using the BNU **ct** command, see:

- "Dialing a number until a connection is made"
- "Dialing multiple numbers until a connection is made"

Dialing a number until a connection is made

Prerequisites

The system to be called must be running Basic Networking Utilities (BNU) or some version of the UNIX-to-UNIX Copy Program (UUCP).

Procedure

This procedure describes how to use the **ct** command to continue dialing a remote modem number until a connection is made or until a specified time has passed.

At the command line on the local system, type:

```
ct -w3 5550990
```

This dials the remote modem at phone numbers `555-0990`. The **-w3** flag and number instructs the **ct** command to dial the remote modem at one minute intervals until a connection is made or until three minutes have passed.

> **Note:** Type the phone number of the remote modem on the **ct** command line, either before or after the flag.

Dialing multiple numbers until a connection is made

Prerequisites

The system to be called must be running Basic Networking Utilities (BNU) or some version of the UNIX-to-UNIX Copy Program (UUCP).

Procedure

This procedure describes how to use the **ct** command to continue dialing multiple remote modem numbers until a connection is made or until a specified time has passed.

At the command line on the local system, type:

```
ct -w6 5550990 5550991 5550992 5550993
```

This dials the remote modems at phone numbers `555-0990`, `555-0991`, `555-0992`, and `555-0993`. The **-w6** flag and number instructs the **ct** command to dial the remote modems at one minute intervals until a connection is made or until six minutes have passed.

> **Note:** Type the phone numbers of the remote modems on the **ct** command line, either before or after the flag.

Exchanging files between local and remote systems

The transfer of files among systems is the most common application of the Basic Networking Utilities (BNU). BNU uses four commands, **uucp**, **uusend**, **uuto**, and **uupick**, to exchange files between local and remote systems.

The **uucp** command is the primary BNU-data transfer utility. The **uusend** command is the Berkeley Software Distribution (BSD) transfer command incorporated into BNU. The **uuto** and **uupick** commands are specialized send and receive commands working with the **uucp** command.

The BNU commands, **uuencode** and **uudecode**, assist file transferring. These commands encode and decode binary files transmitted through the BNU mail facility.

Sending and receiving files with uucp and uusend commands

Use the **uucp** command and options to exchange files within your local system, between your local and a remote system, and between remote systems. The **uucp** options can, for example, create directories to hold files on the receiving end or mail messages on the success or failure of file transfers.

Use the **uusend** command to send files to a remote system that is not directly linked to the sending system but is accessible through a chain of BNU connections. Though equipped with fewer options than the **uucp** command, **uusend** is included among the BNU utilities to satisfy the preferences of BSD UNIX-to-UNIX Copy Program (UUCP) users.

Sending files to a specific user

Prerequisites

The sending and receiving systems must be running either Basic Networking Utilities (BNU) or some version of the UNIX-to-UNIX Copy Program (UUCP).

Use the **uuto** command to send file from one system to another. It is part of the **uucp** command and simplifies the file exchange process for senders and receivers. The **uuto** command sends files to a specific user and deposits them directly into the user's personal directory under that system's BNU public directory. It notifies the recipient that a file has arrived. The recipient uses the **uupick** command to handle the new file.

Sending a file with the uuto command

When you use the **uuto** command to send a file, include the file to be sent, the remote system destination, and the user at the destination. For example:

```
uuto /home/bin/file1 distant!joe
```

This sends **file1** from the local **/home/bin** directory to user `joe` on the remote system **distant**.

The **uuto** command runs under the **uucp** command. The file is transferred to the remote system, in **/var/spool/uucppublic**. The file is deposited in the **/var/spool/uucppublic/receive/user/System** directory on the remote system. If the target directory does not exist, it is created during the file exchange.

The BNU **rmail** command notifies the receiver that a file has arrived.

> **Note:** To send a file to a user on a *local* system, enter the **uuto** command, and include the file to be sent, the local system destination, and the user at the local destination. For example:
>
> ```
> uuto /home/bin/file2 near!nick
> ```
>
> This sends **file2** from the local **/home/bin** directory to user `nick` on the local system **near**.

Receiving and handling files

Prerequisites

The sending and receiving systems must be running either Basic Networking Utilities (BNU) or some version of the UNIX-to-UNIX Copy Program (UUCP).

Use the **uupick** command to receive and manipulate files sent with the **uuto** command. It has file-handling options that allow the recipient to find sent files, move files to a specified directory, execute commands, or delete files.

Receiving a file with the uupick command

Use the **uupick** command to receive a file. For example:

```
uupick
```

The **uupick** command searches the public directory for files that include the remote user ID in the path names. The **uupick** command then displays on the remote screen a message similar to:

```
from system base: file file1?
```

The ? (question mark) on the second line of the notification display prompts the receiver to use any **uupick** options for handling files in BNU's public directory.

uupick options: For a list of all available options, type an asterisk (*) on the line below the question mark (?) prompt. The display, save, and quit options are:

p	Displays the contents of the file.
m *[Directory]*	Saves the file to the directory specified by the *[Directory]* variable. If no destination is given with the **m** option, the file is moved to the current working directory.
q	Quits (exits) from the **uupick** file-handling process.

Encoding and decoding files for transfer

Use the **uuencode** and **uudecode** commands to prepare files for transmission by modem. The commands work in tandem. The **uuencode** command transforms binary files into ASCII files. These files can be sent by the mail facility to a remote system.

With the **uudecode** command, the receiving user converts ASCII-encoded files back to binary format.

Exchanging commands between local and remote systems

The Basic Networking Utilities (BNU) enable users to exchange commands between local and remote systems. The **uux** command runs commands on a remote system. The **uupoll** command controls the timing for command execution.

Requesting execution of command on remote system

Use the **uux** command to request execution of a command on a remote system. The **uux** command does not execute the commands on the remote system. Instead, it prepares the necessary control and data files

in **/var/spool/uucp**. The **uucico** daemon is invoked to do the transfer. After the transfer is complete, the remote system's **uucico** creates an execute file in its spool directory.

When the two **uucico** daemons agree to hang up, the **uuxt** daemon scans the spool directory for outstanding execution requests, checks permissions, and checks to see if additional information is needed. It then forks a command to do what was requested.

> **Note:** You can use the **uux** command on any system configured to run a specified command. However, policies at some sites might restrict the use of certain commands for security reasons. Some sites, for example, might only permit execution of the **mail** command.

After the files are received on the remote system, the **uuxqt** daemon runs the specified command on that system. The **uuxqt** daemon periodically scans the remote system's public spool directory for files received in **uux** transmissions. The **uuxqt** daemon checks that data to be accessed by the sent files is present on the remote system. It also verifies that the sending system has permission to access the data. The **uuxqt** daemon then either executes the command or notifies the sending system that the command did not run.

Transferring a file to a remote system for printing

Prerequisites

- A Basic Networking Utilities (BNU) connection must be established to the remote system
- You must have permission to execute operations on the remote system

Procedure

Use the **uux** command to transfer a file to a remote system for printing.

On the command line of the local system, type:

```
uux remote!/usr/bin/lpr local!filename
```

This prints the local file *filename* on the remote system.

Starting the transmission of spooled jobs

Use the **uupoll** command to start the transmission of jobs stored in the local system's public spooling directory.

The **uupoll** command creates a null job in the public directory for the remote system and starts the **uucico** daemon. This forces the **uucico** daemon to immediately contact the remote system and transfer any queued jobs.

Identifying compatible systems

Use the **uuname** command to display a list of all systems accessible to the local system. For example, when you type:

```
uuname
```

at the command line, the system displays a list such as:

```
arthur
hera
merlin
zeus
```

This information is used to determine the name of an accessible system before copying a file to it. The **uuname** command is also used to establish the identity of the local system. The **uuname** command acquires its information by reading the **/etc/uucp/systems** file.

Reporting the status of command and file exchanges

The Basic Networking Utilities (BNU) have three commands available for learning the status of command and file exchanges:

- **uusnap** command displays the status of BNU connections with remote systems
- **uuq** command shows the BNU job queue
- **uustat** command reports information about the status of several BNU operations

Displaying the status of systems connected by BNU

The **uusnap** command displays a table of information about all the systems connected by BNU. The table shows a line for each system, reporting the names and the numbers of command files, data files, and remote command executions holding in the systems' queues. The last item on each line is a status message. This message indicates either a successful BNU connection or an explanation of why BNU did not establish a link.

Displaying the BNU job queue

The **uuq** command lists any entries in the BNU job queue. The format of the list is similar to the format displayed by the **ls** command. The display for each entry includes the job number, followed on the same line by a summary, including the system name, the number of jobs for the system, and the total number of bytes to send. Users with root authority can use a **uuq** command to identify specific queued jobs by their job numbers.

Displaying the status of BNU operations

The **uustat** command provides the status of a particular command or file exchange in the BNU system. Entered without flag options, the **uustat** command displays a single line for each job requested by the current user, including:

- Job ID number
- Date and time
- Status (send or receive)
- System name
- User ID of the person who issued the command
- Size and name of the job file

Equipped with several flags, the **uustat** command can report on all jobs, by all users, in the queue, or on jobs requested by other systems on the network.

The **uustat** command gives users limited control of jobs queued to run on a remote computer. You can examine the status of BNU connections to other systems and track file and command exchanges. Then, for example, you can cancel copy requests started by the **uucp** command.

Canceling remote jobs

Prerequisites

- A Basic Networking Utilities (BNU) connection must be established with the target remote system
- A remote job must have been issued from the local system

Procedure

Use the **uustat** command to cancel a BNU process issued to a remote system:

1. Determine the job ID number of the process, listed in the remote queue. On the command line of the local system, type:

   ```
   uustat -a
   ```

The *-a* option displays all jobs in the remote system's holding queue and the job requests of any other BNU users on the system.

BNU responds with a message similar to:

```
heraC3113   11/06-17:47 S hera   you  289 D.venus471afd8
merlinC3119 11/06-17:49 S merlin jane 338 D.venus471bc0a
```

2. Then type:

   ```
   uustat -k heraC3113
   ```

   The *-k* option cancels the `heraC3113` job request.

BNU commands

This list includes commands for using Basic Networking Utilities (BNU).

See "BNU files, file formats, and directories" for more reference information.

ct	Connects with another system using a telephone line.
cu	Connects with another system using a telephone line or a direct hardwire connection.
tip	Connects with another system. This variation of the **cu** command is an additional connection command installed with BNU. It requires special configuration.
uucp	Copies files from one system to another system running either BNU or another version of UNIX-to-UNIX Copy Program (UUCP).
uudecode	Decodes a binary file encoded by the **uuencode** command.
uuencode	Encodes a binary file for transmission by mail.
uuname	Lists accessible systems.
uupick	Completes the transfer of files sent by the **uuto** command.
uupoll	Forces a call to a remote system so that queued jobs can be transferred.
uuq	Displays the BNU job queue.
uusend	Uses BNU to send a file to a remote host running either BNU or another version of UUCP.
uusnap	Displays a brief summary of BNU's status.
uustat	Reports the status of, and provides limited control over, BNU operations.
uuto	Copies files to another system using either BNU or another version of UUCP.
uux	Requests execution of a command on another system using either BNU or another version of UUCP.

BNU files, file formats, and directories

The list includes:

These files, file formats, and directories are for use in BNU security, administration, configuration, and emulation.

Lock and public directories

- **/var/locks** directory
- **/var/spool/uucppublic/*** directories

Administrative directories and files

- **/var/spool/uucppublic/*** directories
- **/var/spool/uucp** directory

- **/var/spool/uucp/.Admin** directory
- **audit** file
- **errors** file
- **Foreign** file
- **xferstats** file
- **/var/spool/uucp/.Corrupt** directory
- **/var/spool/uucp/.Log** directories
- **/var/spool/uucp/.Old** directory
- **/var/spool/uucp/.Status** directory
- **/var/spool/uucp/.Workspace** directory
- **/var/spool/uucp/.Xqtdir** directory
- **/var/spool/uucp/***SystemName* directory
- Command (**C.***) files
- Data (**D.***) files
- Execute (**X.***) files
- Temporary (**TM.***) files

Configuration file formats

The BNU configuration file formats that reside in the **/etc/uucp** directory are:

- **Devices**
- **Dialcodes**
- **Dialers**
- **Maxuuscheds**
- **Maxuuxqts**
- **Permissions**
- **Poll**
- **Sysfiles**
- **Systems**

The BNU configuration file format that resides in the **/usr/sbin/uucp** directory is:

- **remote.unknown**

Emulator file formats

- **tip phones**
- **tip remote**
- **tip .tiprc**

Chapter 5. Asynchronous Terminal Emulation (ATE)

Asynchronous Terminal Emulation (ATE) enables a terminal to emulate a remote system. With ATE, you can connect to most systems that support asynchronous terminals. ATE makes the remote system see a terminal either as a system's display or as a DEC VT100 terminal. The VT100 option allows you to log in to systems that do not support your terminal, but do support VT100 terminals.

ATE uses both direct (cabled) and modem connections to communicate between your system and a remote system. You can use ATE to connect either to a system in the next room or to a system across the country. For a direct connection, you must know the port to use on your system. For a modem connection, you must know the port to use on your system and the telephone number of the remote system. You must also have a login ID and password on the remote system.

Topics discussed in this chapter are:

ATE concepts

Asynchronous Terminal Emulation (ATE) enables a user to run commands on the remote system, send and receive files, and use the **xmodem** protocol to check the data integrity in the files transferred between systems. The user can also capture and file incoming data from the remote system.

> **Note:** You must be a member of the UNIX-to-UNIX Copy Program (UUCP) group in order to use ATE. A user with root authority uses System Management Interface Tool (SMIT) to install individual users in groups.

This section provides the following conceptual information:

Starting ATE

ATE uses menus and subcommands. Starting ATE with the **ate** command displays the Unconnected Main Menu, which lets you:

- Temporarily change characteristics of ATE (**modify**, **alter**)
- Connect to another system (**directory**, **connect**)
- Get help (**help**)
- Execute workstation operating system commands on the system (**perform**)
- Leave ATE (**quit**)

Depending on the subcommand issued from the Unconnected Main Menu, ATE displays various submenus:

When you use	ATE displays
modify subcommand	Modify Menu
alter subcommand	Alter Menu
connect or **directory** subcommand to connect to a remote system	Connected Main Menu
directory subcommand	dialing directory (a list of phone numbers)

From the Connected Main Menu, you can issue subcommands to:

- Send files to and receive files from the remote system (**send**, **receive**)
- Send a break signal to the remote system (**break**)
- End the connection to the remote system (**terminate**)

In addition, the **modify**, **alter**, **help**, **perform**, and **quit** subcommands perform the same functions as those provided from the Unconnected Main Menu.

You can control certain actions of ATE with control key sequences. These key sequences are known as the CAPTURE_KEY, the MAINMENU_KEY, and the PREVIOUS_KEY. The key sequences are discussed in “Control key sequences” on page 69. ATE is installed with default key combinations for these keys, but you can change the key combinations by modifying the ATE default file, **ate.def**.

Displaying the ATE Unconnected Main Menu

Use the **ate** command to display the ATE Unconnected Main Menu. After a connection is established, use the ATE **connect** subcommand to display the Connected Main Menu.

You can issue the following subcommands from the ATE Unconnected Main Menu. To issue the subcommand, type the first letter of the subcommand at the command prompt on the menu. For example, type **d** to issue the **directory** subcommand.

alter Temporarily changes data transmission characteristics, such as the speed of transmission.
connect Makes a connection.
directory Displays a dialing directory.
help Displays help information.
modify Temporarily modifies local settings, such as the capture file, for incoming data.
perform Allows you to perform workstation operating system commands within ATE.
quit Quits the ATE program.

Note: Of the control key sequences CAPTURE_KEY, MAINMENU_KEY, and PREVIOUS_KEY, only REVIOUS_KEY may be used from the ATE Unconnected Main Menu.

Displaying the ATE Connected Main Menu

Use the **connect** subcommand from the ATE Unconnected Main Menu to display the Connected Main Menu. Or, press the MAINMENU_KEY while connected to a remote system.

You can issue the following subcommands from the ATE Connected Main Menu. To issue the subcommand, type the first letter of the subcommand at the command prompt on the menu. For example, type **a** to issue the **alter** subcommand.

alter Temporarily changes data transmission characteristics, such as the speed of transmission.
break Sends a break signal to the remote system.
help Displays help information.

modify	Temporarily modifies local settings used by the emulator, such as the capture file for incoming data.
perform	Allows you to perform workstation operating system commands within ATE.
quit	Quits the ATE program.
receive	Receives files from a remote system.
send	Sends files to a remote system.
terminate	Terminates the ATE connection.

All three ATE control key sequences can be used from the ATE Connected Main Menu.

Control key sequences

Use the following control keys with ATE. Change the key sequence for each function by editing the **ate.def** file.

CAPTURE_KEY	Starts or stops saving data displayed on the screen during a connection. The default key sequence for the CAPTURE_KEY is Ctrl-B. The CAPTURE_KEY has a switch or toggle effect. Pressing this control key starts saving data. Pressing this control key a second time stops saving data. Data is saved in the capture file defined in the **ate.def** file. The default capture file name is the **$HOME/kapture** file. Use the **modify** subcommand to temporarily change the capture file name. Edit the ATE default file to permanently change the name of the capture file. See “Editing the ATE default file” on page 70. The CAPTURE_KEY key sequence does not function while the terminal is performing a file transfer operation, and it is valid only when a connection is established. If you press the CAPTURE_KEY key sequence before a connection is established, the next command entered is unsuccessful, and an error message is displayed.
PREVIOUS_KEY	Returns to the previously displayed screen. The PREVIOUS_KEY is also used to stop a file transfer operation. The default key sequence for the PREVIOUS_KEY is Ctrl-R. The PREVIOUS_KEY can be used from either ATE Main Menu.
MAINMENU_KEY	Displays the Connected Main Menu so you can issue an ATE subcommand. The default key sequence for the MAINMENU_KEY is Ctrl-V. Use this control key to display the Connected Main Menu after a connection to a remote system is established. If you press the MAINMENU_KEY key sequence before a connection is established, the next command entered is unsuccessful, and an error message is displayed. By customizing the ATE default file, you can permanently change the control key settings and the capture file name. See “Editing the ATE default file” on page 70.

Customizing ATE

ATE creates the **ate.def** default file in the current directory the first time you run ATE. Edit the **ate.def** file to customize various aspects of ATE. For example, you can change the name of the dialing directory file, the type of transfer protocols used to send and receive files from the remote system, and the baud rate ATE expects the modem to use. See “Editing the ATE default file” on page 70 for a complete discussion of ATE customization.

In addition to editing the default file to customize ATE, you can temporarily change certain aspects of ATE with the **modify** and **alter** subcommands. These subcommands cannot change the control key sequences (which can only be changed by editing the default file) and the name of the dialing directory (which can be changed with the **directory** subcommand or by editing the default file). Any changes made with the

modify, **alter**, or **directory** subcommands are effective only for that session of ATE. The next time the you run ATE, the settings used are those defined in the default file.

When using a modem with ATE, you can create a dialing directory of up to 20 phone numbers. The **directory** subcommand displays the telephone numbers and lets you select one to connect to the system being called.

By using a dialing directory, you avoid having to look up the telephone number when calling a particular system. You can also specify certain data transmission characteristics in the dialing directory file. This is useful if some connections use characteristics that differ from the defaults.

You can create a personalized dialing directory, and the system administrator can create a system-wide dialing directory. Specify which dialing directory to use in the ATE default file. See “Setting up an ATE dialing directory” for more information.

Setting up an ATE dialing directory

Prerequisites

- The Asynchronous Terminal Emulation (ATE) program must be set up on the system
- To set up a system-wide dialing directory, the user must have write access to the **/usr/lib/dir** file

Procedure

1. Create the dialing directory file:
   a. Change to the directory where the dialing directory file will reside.
   b. Copy the **/usr/lib/dir** file to use as a template. Rename the file to any valid file name.
   c. Create telephone number entries using the format given in the dialing directory file format.
   d. Save the file.

      Note: If the new dialing directory file is to be the system-wide default file, save the file with the name **/usr/lib/dir**.
2. If the dialing directory file name is not the default (**/usr/lib/dir**), edit the **ate.def** file in the directory from which the ATE program is run. Change the *DIRECTORY* parameter in the **ate.def** file to the new dialing directory file. See “Editing the ATE default file.”
3. Start ATE, and view the dialing directory with the **directory** subcommand.

Editing the ATE default file

Prerequisites

The Asynchronous Terminal Emulation (ATE) program must be set up on the system.

Procedure

To change the settings in the **ate.def** file:

1. Open the **ate.def** file with an ASCII text editor.
2. Enter new values for the parameters to be changed. Other values can be deleted or left alone. The system uses its defaults for any parameters that are deleted.
3. Save the modified **ate.def** file.

The changes made to the **ate.def** file take effect the next time ATE is run from the directory containing the customized **ate.def** file.

You can keep a copy of the **ate.def** file in any directory to which you have read and write permissions. For example, if you need to run the ATE program with different defaults at different times, keep multiple copies of the **ate.def** file, with the appropriate settings, in different subdirectories of the **$HOME** directory. However, multiple copies of the **ate.def** file use system storage. As an alternative, temporarily change most settings with the ATE **alter** and **modify** subcommands. Use a dialing directory entry to change settings for an individual modem connection. See “Setting up an ATE dialing directory” on page 70.

ATE commands

This Asynchronous Terminal Emulation (ATE) list includes:

- “ate command and subcommands”
- “xmodem command.”

See “ATE file formats” for additional reference information.

ate command and subcommands

ate Starts the ATE program. Subcommands are:

break Inputs current activity on a remote system.

connect
Connects to a remote computer.

directory
Displays the ATE dialing directory and lets you choose an entry from the directory to connect to a remote system.

help Provides help for the ATE subcommands.

perform
Lets you issue workstation operating system commands while using ATE.

quit Exits the ATE program.

receive
Receives a file from a remote system.

send Sends a file to a remote file system.

terminate
Terminates an ATE connection to a remote system.

xmodem command

xmodem Transfers files with the xmodem protocol, which detects data transmission errors during asynchronous transmission.

ATE file formats

This section lists Asynchronous Terminal Emulation (ATE) file formats.

ate.def	Sets default settings for connections.
Dialing directory	Defines telephone numbers and settings for specific modem connections.

See “ATE commands” for additional reference information.

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull AIX 5L System User's Guide Communications and Networks

Nº Reférence / Reference Nº : 86 A2 45EM 01

Daté / Dated : October 2005

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

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

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

NOM / NAME : _____ Date : _____

SOCIETE / COMPANY : _____

ADRESSE / ADDRESS : _____

_____

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

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

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

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

BULL CEDOC
ATTN / Mr. L. CHERUBIN
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone / Téléphone : +33 (0) 2 41 73 63 96
FAX / Télécopie +33 (0) 2 41 73 60 19
E–Mail / Courrier Electronique : srv.Cedoc@franp.bull.fr

Or visit our web sites at: / Ou visitez nos sites web à:
http://www.logistics.bull.net/cedoc
`http://www-frec.bull.com` `http://www.bull.com`

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

NOM / NAME : ______________________________ Date : __________

SOCIETE / COMPANY : ______________________________

ADRESSE / ADDRESS : ______________________________

____

PHONE / TELEPHONE : ____________________ FAX : ____________________

E–MAIL : ______________________________

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

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

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

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

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 45EM 01