IBM RT PC Advanced Interactive Executive Operating System

AIX Operating System Communications Guide

Programming Family

IBM

Personal Computer Software

59X7668

IBM RT PC Advanced Interactive Executive Operating System

AIX Operating System Communications Guide

Programming Family

IBM

Personal
Computer
Software

First Edition (NOVEMBER 1985)

About This Book

About This Book

This guide explains how to use the local and remote AIX communications facilities of IBM RT PC and how to customize the system for your own needs. With this information, you can send and receive messages and files as well as carry out certain commands remotely.

Who Should Read This Book

This guide is intended mainly for system users already familiar with IBM RT PC and AIX. Further, to install or reconfigure the system, you should obtain superuser authority to gain access to files, commands, and directories you may need for these tasks. AIX is based on UNIX[1] System V.2.

If you are not familiar with IBM RT PC and AIX, but you have been given login and password information, you can begin to use the commands in Chapter 3 right away.

[1] Trademark of AT&T Bell Laboratories.

Before You Begin

To use the communications facilities described in this guide, you will need the following hardware items:

- Asynchronous serial adapter capable of handling speeds up to 19.2 bps
- Modem, modem eliminator, or cable.
- A keyboard from the following list:
  - IBM RT PC Keyboard
  - IBM PC 1 or PC AT keyboard
  - IBM 3161 keyboard
  - DEC VT100[2] keyboard
  - DEC VT220[3] keyboard.

See Chapter 5 of this guide and *User Setup Guide* for more on hardware.

The software you need is the IBM RT PC AIX Operating System Licensed Program Product (LPP).

2 Trademark of Digital Equipment Corporation.

3 Trademark of Digital Equipment Corporation.

How To Use This Guide

This guide has five chapters arranged in the order of increasing complexity of subject matter. A brief description of the contents follows:

- Chapter 1, "Introduction," provides an overview of AIX-based communications. It also describes how to communicate with other people in a local, multi-user environment and how to communicate with other systems using either Asynchronous Terminal Emulation or UUCP.
- Chapter 2, "Communicating with Other People," gives an overview of the multi-user environment and explains five commands for communicating with other system users on your local system.
- Chapter 3, "Communicating: Asynchronous Terminal Emulation," shows how to create a dialing directory file and how to use menus for interactive communications between your work station and a remote system.
- Chapter 4, "Communicating: UUCP," shows how to use this batch process that runs in the background to communicate with other system users on local or remote systems and how to protect a system against unauthorized users.
- Chapter 5, "Configuring for Communications," contains information on ports and cables.
- The "Glossary" and "Index" give definitions and references to information.
- A Reader's Comment Form and Book Evaluation Form are provided at the back of this book. Use the Reader's Comment Form at any time to give IBM information that may improve the book. After you have become familiar with the mamual, use the Book Evaluation Form to give IBM specific feedback about it.

Certain typographical conventions used in this guide are as follows:

- The name of a program or command appears in lowercase bold:

  uux program or **uux command**

- A general name or field that must have a real name substituted for it appears in italics:

  file-name

  Book titles and emphasized words or phrases also appear in italics.

- Words that you are supposed to key in or words that the system causes to appear on your display appear in monospace type:

  ```
  Select an option.
  ```

- Optional flags for commands are enclosed in { } (braces). For example, the following line means you can use the command either alone or with the flag:

  ```
  who {-u}
  ```

Prerequisite Information

You can get started in the multi-user environment with relatively little experience in computer communications. The first system-to-system feature, Asynchronous Terminal Emulation, requires some computer experience and an understanding of how computers communicate with each other. To use UUCP (UNIX-to-UNIX copy facility), you need more knowledge and experience with communications.

You should be familiar with the following publications before starting to use this guide:

- *IBM RT PC AIX Operating System Commands Reference* lists and describes the AIX Operating System commands.
- *IBM RT PC Using and Managing the AIX Operating System* describes using AIX Operating System commands, working with

file system, and developing shell procedures. This book also provides instructions for performing such system management tasks as adding and deleting user IDs, creating and mounting file systems, backing up the system, and repairing file system damage.

Related Information

Other publications that you may find helpful include the following:

- *IBM RT PC Messages Reference* lists messages displayed by the IBM NEW PRODUCT and explains how to respond to the messages.
- *IBM RT PC Guide to Operations* describes the IBM NEW PRODUCT Model Y, Model X, and Model XX system units, the displays, keyboard, and other devices that can be attached. This guide also includes procedures for operating the hardware and moving the IBM NEW PRODUCT Model Y, Model X, and Model XX system units.
- *IBM RT PC Installing and Customizing the AIX Operating System* provides step-by-step instructions for installing and customizing the Operating System, including how to add or delete devices from the system and how to define device characteristics. This book also explains how to create, delete, or change NP and non-NP minidisks.

Ordering Additional Copies of This Book

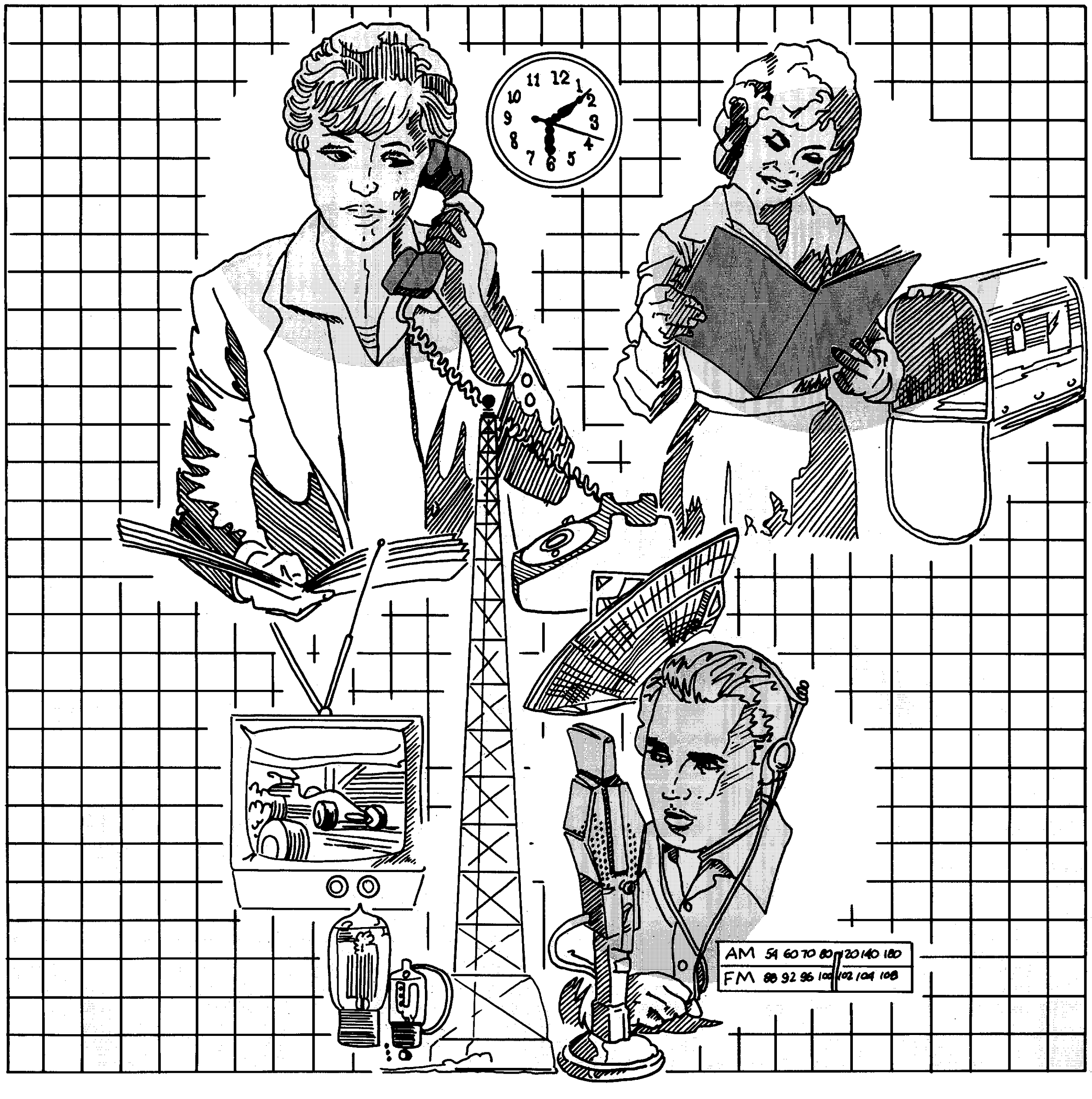
To order additional copies of this publication use either of the following sources:

- To order from your IBM representative, use Order Number SV21-8007
- To order from your IBM dealer, use Part Number 55X8894.

A binder is included in the order.

Contents

Chapter 1. Introduction



CONTENTS

About This Chapter

This chapter provides some general information on computer communications and refers you to other sources of information. It focuses on AIX-based communications, explains the similarities and differences among the various ways to communicate on this system, and tells why you might choose a certain method to accomplish a specific task.

Overview of Communications

Communications is the process of transferring information electronically from one location to another. Two terminals or work stations can communicate with each other, two computer systems can communicate, and a terminal or work station and a computer system can communicate. The components of a communications system include terminals or work stations, computers, programs, and transmission equipment. For detailed information, see the IBM publication *Data Communications Concepts* (GC21-5169) or books on computer networks and communications systems.

This guide concentrates on AIX-based communications to take advantage of the operating system that this computer uses. There are many UNIX installations, especially in the scientific, academic, and research communities.

Communications Methods Described in this Guide

The methods of communicating explained in this guide include the following:

- Between two or more people in a local, multi-user environment (People are frequently called *system users*)
- Between two different computer systems that may be miles apart.

Communicating with Other System Users

Communicating with other people in a multi-user environment requires little or no advance preparation. For example, the system automatically sets up a mailbox for you the first time another user sends you mail.

There are five tasks that you can perform with this communication method:

- Use the **who** command to find out the name and location of each person currently logged into the system as well as the date and hour of the current login session.
- Use the **write** command to send a message to another person who is currently logged in.
- Use the **mesg** command either to accept or reject messages from other system users while you are logged in.
- Use the **confer** command to conduct an on-line conference with several other system users.
- Use the **mail** command to send a mail message to another system user, whether that person is currently logged in or not.

It is possible to send fairly long messages or even files with **mail**, but the command was designed for short messages since the amount of space assigned to mailboxes is typically quite small.

Tasks you should perform with one of the alternative methods of communicating, explained later in this chapter, include the following:

- Sending or receiving large files
- Taking measures for maximum security in file transfers
- Communicating with a remote system.

Communicating with Other Systems

This guide describes two ways to communicate with other systems. The first, using Asynchronous Terminal Emulation, shows how to use this computer interactively just as if you were directly connected to the remote system. The second, using UUCP, shows you how to perform file transfer tasks and remote execution in the background so that you can use your work station for other work at the same time.

Communicating with Other Systems: Asynchronous Terminal Emulation

This method of communicating with other systems was designed to let you connect to a public or private system and use the facilities of that system just as if your work station were directly connected to it. Asynchronous Terminal Emulation displays menus from which you can select commands to carry out the tasks you want to perform. These tasks include the following:

- Store a list of frequently dialed numbers in a dialing directory file
- Make a connection to a remote system
- Send and receive files on a connection
- Modify certain characteristics of your terminal
- Alter various data transmission characteristics and connection settings

- Set up a file containing the transmission and connection information that you want the system to employ each time you use Asynchronous Terminal Emulation.

Asynchronous Terminal Emulation was designed to be a versatile, easy-to-use communications program, and you may find it adequate for all or most of your file transfer needs. But there are several tasks that are better suited for UUCP:

- Perform file transfers as a background process so that your terminal is free for other work
- Send files to specific system users on a local or remote system
- Notify yourself and another person that a file transfer completed successfully
- Run commands on a remote system.

Communicating with Other Systems: UUCP

UUCP (UNIX-to-UNIX Copy facility) is a set of directories, files, programs, and commands that you can use to communicate with another system, either directly over a dedicated line or indirectly over a telephone line. You can configure a system as a call-in site, a call-out site, or both.

The tasks you can perform with UUCP include the following:

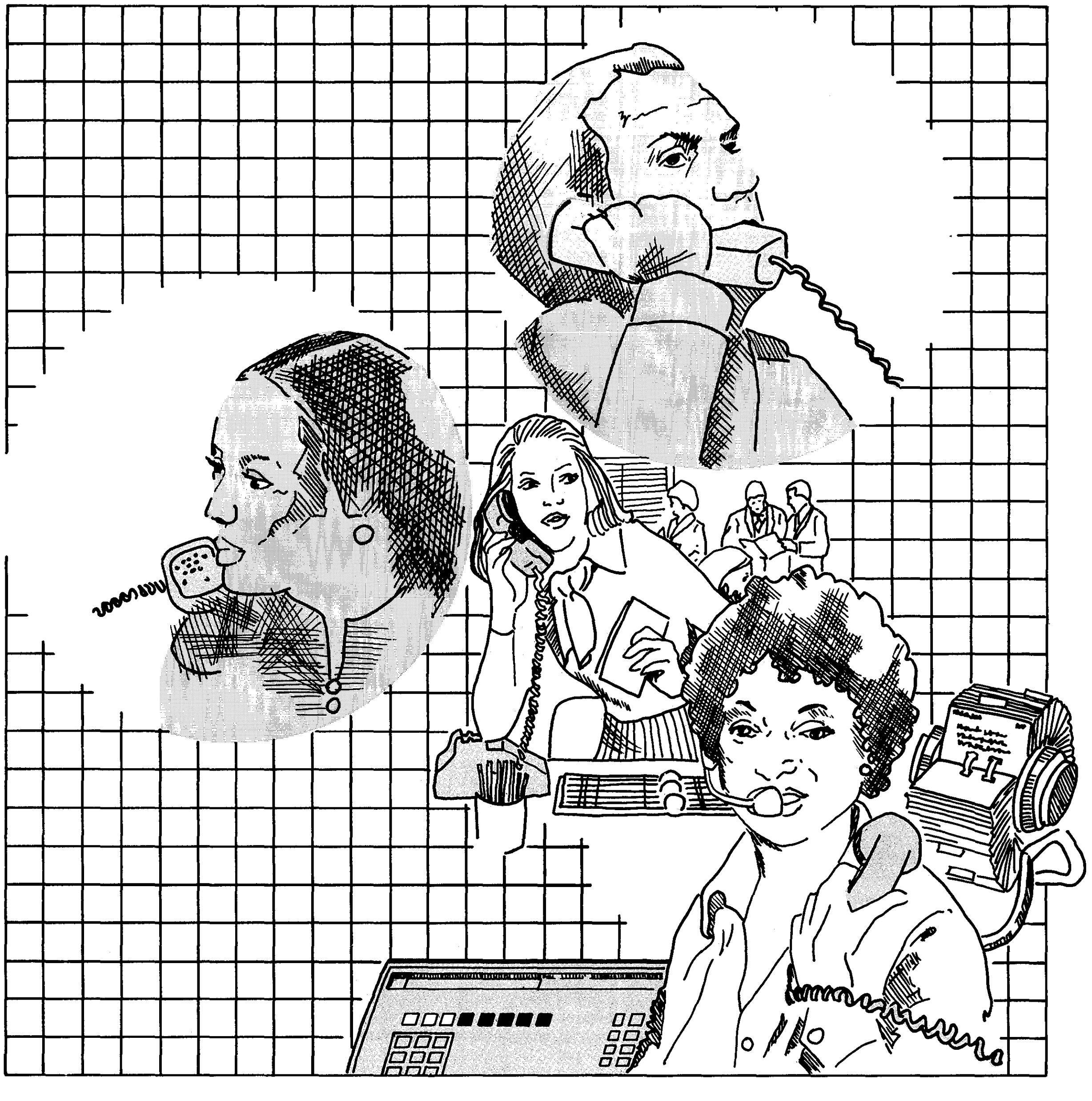
- Gather and send one or more source files from one system to a destination file on another system or on the local system
- Notify yourself and another system user about the successful completion of a file transfer
- Run commands on another system without logging into it
- Route file transfers through several intermediate systems
- Send or receive files to or from specific people on other systems or on the local system

- Perform various record-keeping and clean-up functions.

Since UUCP is the most complex of the communication methods described in this book, you should make sure you are thoroughly familiar with this computer and UNIX before you start.

Also, you should be aware that UUCP provides no way for you to monitor a file transfer while it is in progress. If the transfer fails, you may not find out about the failure for a period of time.

Chapter 2. Communicating with Other People



CONTENTS

About This Chapter

This chapter shows you how to use five commands to communicate with other people on the local system:

- **who**
- **write**
- **mesg**
- **confer**
- **mail**

Introduction

The multiple-user environment supports several work stations connected to the computer. You can communicate with people at other work stations by using one or more of these five commands:

- The **who** command gives you information about each person currently logged into the system so that you can direct your communication to a specific name at a specific terminal.
- The **write** command lets you send messages to another logged-in system user.
- The **mesg** command allows you to determine whether or not you will receive messages from other people while you are logged in at your work station.
- The **confer** command shows you how to set up and carry out on-line conferences among several logged-in system users
- The **mail** command lets you send a message to someone who customarily uses the system but who is not logged in at the time that you send it. The command also informs you that someone has sent you a message.

  Additional information on **mail** is in the *AIX Operating System Commands Reference.*

The who Command

Keying in the **who** command gives you the following information:

- The name by which a person is known on the system
- The work station where the person is now logged in
- The date and hour when the person logged in for the current work session.

For example, if your login name is pat and you are logged into tty1, and one other person is logged in, you would see a display similar to the following one after you entered **who** on the system prompt line right after the $ (dollar sign):

```
bjh  tty0  Feb 8  07:43
pat  tty1  Feb 8  09:27
```

This shows that system user bjh has been logged in since 7:43 at a work station known to the system as tty0.

You can obtain more information by keying in **who -u**:

- The number of hours and minutes that have elapsed since there was any activity at each work station. From this, you may be able to tell how busy someone is.
- The process ID of each logged-in system user.

For example, the **who -u** command, with bjh and pat logged in, would display information similar to the following:

```
bjh  tty00  Feb 8  07:43  01:02  17
pat  tty01  Feb 8  09:27    .    28
```

From this you can see that bjh has been inactive for an hour and two minutes and had a process ID number of 17. The **.** (dot) in your own elapsed time column means that you have been inactive for less than a minute.

The write Command

After the **who** command gives you the name someone uses on the system, you can send a logged-in user a message with a **write** command. To transmit a message to *bjh*, for example, you would take the following steps:

1. On the command line right after the $ (system prompt), type

   ```
   write bjh
   ```

   On the display `bjh` is using, there should appear a notice like the following:

   ```
   message from pat tty1 Feb 8 10:32:45
   ```

2. You do not need to wait for a reply before sending your message. However, `bjh` must use the **write** command to send anything to you. If so, you will see a message similar to this one:

   ```
   message from bjh tty0 Feb 8 10:33:03
   ```

   See the **mesg** command later in this chapter if `bjh` does not respond when you expect an answer or if you get an error notice like this one:

   ```
   cannot write to bjh on tty0
   ```

   If neither event happens, type your message and then, on a line by itself, press **Ctrl-d**. (Hold down **Ctrl** and press **d** at the same time. The **Ctrl** key is at the left of your keyboard). This displays an end-of-transmission symbol (EOT) at your recipient's work station.

   Note: If you are not using one of the keyboards listed in "Before You Begin" on page vi, this key sequence may give different results. See "Remapping Keys" on page 3-65 for information on resetting key values.

3. If you expect several exchanges of messages with someone, you should end each of your messages, as well as the entire conversation, with distinctive signals. It is common to use o (for over) at the end of a message and oo (for over and out) at the end of the whole communication period. This convention allows you and the person you are communicating with to take turns sending messages without having to retype `write bjh` and **Ctrl-d** before and after each item you send.

Additional Information

When you key in the **write** command, it sends an attention-getting sound (the ASCII BEL character) and the notice to a person, `bjh`, for example. When the connection is ready, you receive two ASCII BEL characters to let you know that whatever you type next will appear on the display where `bjh` is logged in. You will see it on your own screen, too. If you press **Ctrl-d**, the end-of-transmission character (EOT) also goes to your correspondent's display, showing that you do not expect a response.

If your intended receiver is not logged into the system when you try to send a message, you will get a notice like this:

```
user not logged in
```

If you wish, you can use the **mail** command explained later in this chapter to store a message for delivery at the person's next login.

Two Other Ways To Use the write Command

For a long message, you may prefer to use an editor to place your text in a file where you can edit it. Then you can redirect the file to `bjh` with the **write** command. If you name your file `bjh.letter`, this is how you would send the file:

```
write bjh <bjh.letter
```

If `bjh` is logged in and receiving messages (See the **mesg** command later in this chapter), your letter and the EOT character will go to bjh and there will be no further connection between the two work stations.

The other way to use **write** leaves the two work stations connected when your transmission is complete so that you or bjh can immediately send more material. For this method, you send the output of a command to the other user.

First, you set up a file for your text, like the bjh.letter in the previous example.

Next, you establish a connection by keying in the **write** command and receiving the beeping sounds (ASCII BEL characters).

Then, you use the ! (exclamation mark), the command name, and the file name. In this use of the **write** command, you are calling the shell to run the command and write the results to bjh. Since the connection between you and the other person is still in effect, both of you can continue the conversation if you wish.

The mesg Command

When you are logged into the system and working, you may not want other people to interrupt you by sending messages to your display. A message from someone else will not destroy anything you are working on, but you can prevent others from communicating with you if you wish. Three forms of the **mesg** command (`mesg`, `mesg n`, `mesg y`) give you control over incoming messages.

Three Ways To Use the mesg Command

First, you can key in **mesg** right after the $ (system prompt) on the command line to find out whether your work station is set up to receive messages or not. A **y** means that your work station will receive incoming messages; an **n** means it will not.

Then to prevent **write** messages from appearing on your display, key in the following:

```
mesg n
```

To allow messages, type:

```
mesg y
```

There is no harm to using **mesg n** or **mesg y** without first checking the status with **mesg**.

Additional Information

When you first login to a work station, the shell start-up procedure automatically permits you to receive messages. You can override this default by editing your **$HOME/.profile** file. To this file, add this line:

```
mesg n
```

There are two cases where using **mesg n** will not prevent incoming messages from appearing:

- A person with **superuser** privileges can send **write** messages to a work station, regardless of its permission status.
- Someone using the **mail** command, explained later in this chapter, can get a message through because **mesg n** works only with the **write** command. The **mail** command does not deliver the message immediately unless the recipient is also using the **mail** command, but that person will get the mail at the next login or at the next use of the **mail** command.

The confer Command

The **confer** command lets several people participate in an on-line conference. You can conduct the conference *on the record* and mail a transcript to each participant or have it *off the record* and discard the proceedings. As in a face-to-face conference, there are rules to promote an orderly exchange of information. For instance, only one person at a time should *take the floor*. Also, each participant who leaves the conference should be *excused*.

It is essential to use o (over) and oo (over and out) as with **write** to avoid the chaos of having messages appear as if from nowhere. Once a conference is established, whatever a participant keys in goes as is to the other participants' displays. Therefore, one conferee at the time should take the floor, send a message ended by o (over), and then yield the floor by pressing Enter.

In the example that follows, you are the conference leader with the login name of `pat`. The users you want to set up a conference with are `bjh`, `chris`, and `smiley`. If you want to get a written record of the conference, key in this:

```
confer bjh chris smiley
```

If you do not want a transcript, use ~ (tilde character) right after **confer** like this:

```
confer ~ bjh chris smiley
```

All of the other users, if logged in, receive a notice at their work stations that you want a conference:

```
please joinconf with pat
```

To participate, each of the other people should key in this line:

```
joinconf pat
```

Each user then receives notice that others have entered the conference:

```
bjh has joined
chris has joined
smiley has joined
```

Conducting a Conference

To take the floor, press the Enter key once. You should see your name in square brackets. If not, someone else may have tried to take the floor at the same time. Press Enter again. When you get the floor, your name displays in brackets at the work stations of the other participants so that they know you have the floor. Then type your contribution to the conference, ending with o (for over).

To yield the floor so another user can communicate, enter a blank line by pressing the Enter key once.

If more than one conferee claims the floor at the same time, the last person to do so (the one whose name appears in brackets last) takes the floor. The others should immediately yield by entering a blank line. It is possible, if pointless, to contend for the floor at length, so all participants should agree in advance to wait until the person who has the floor finishes and keys in o (for over).

To withdraw from the conference while it is still going on, first take the floor and then press **Ctrl-d**. If `smiley` wants to leave, for example, the other three users get this message on their displays:

```
[smiley] BYE.
```

When they see this, the other users should key in the following so that `smiley` can do other work and will not continue to receive conference output on the screen:

```
!excuse smiley
```

You then receive a message that `smiley` has successfully left:

```
smiley excused
```

Normally, the conference leader stays till the end, but it is all right for another user to close. The last participant only has to key in **Ctrl-d** to end the conference because **!excuse** ends the connections with other users.

Getting a Transcript

You can get a transcript of a conference you initiated on the record. If you used **confer** without the ~, you can now request a print out because a prompt like the following appears on your display:

```
Transcript? (n)
```

Type **y** for yes or press Enter for the default (no transcript).

It is not necessary to give a unique name to a conference, even though you may have several conferences in a short period of time. The system assigns a new name to each one, such as **pat** for the first, **pat-1** for the second, **pat-2** for the third, and so on. The name of the conference appears on the transcript, so you can keep the papers in order.

For more information on **confer**, see *AIX Operating System Commands Reference.*

The mail Command

The **mail** command lets you send messages to other people whether they are logged in or not. The **mail** command also enables you to read messages other users sent during your absence from the system.

Sending Mail

To send a message to someone, key in the command and a system user ID like this example:

```
mail bjh
```

Press Enter, and when the system prompt does not appear, you will know that whatever you type will go to `bjh's` mailbox.

To end your transmission, be sure to key in a **.** (period) on a line by itself at the end of your message. Wait for the $ (system prompt) to return to the screen before doing other work.

You can send the same message to as many mailboxes as you wish by listing each name, separating them with blanks like this example:

```
mail bjh chris smiley
```

The mail each person receives contains no notice that other users are getting the same message.

If you want your message recipients to know who else is getting the same mail, you can have each person's name prefixed to the message by keying in **-t** like this example:

```
mail -t bjh chris smiley
```

Reading Your Mail

If you have received messages while you were not logged in, you will get a mail notice when you login. A typical notice is this one:

```
[you have new mail]
```

Note: You will also get a new mail notice when a message arrives while you are using the **mail** command, either to send or to read messages.

To read all of your messages, just key in the command by itself:

```
mail
```

A message has a *postmark* first. It contains `From:`, the sender's name, the date, and the time the message was sent to you.

If there are several messages, the last one to arrive is normally displayed first. Following each message is a prompt, the **?** (question mark), that requires you to respond with an action that tells the system how to dispose of your mail. See "Options for Reading and Disposing of Mail" on page 2-16 for the choices you can make.

Before turning to the list, you should understand three files that the **mail** command uses.

- **/usr/mail/pat**

  If your login name is `pat`, this file is where messages from other system users are temporarily stored for you to read later.

- **/usr/pat/mbox**

  This is the name of the mail file that the system automatically sets up for you.

- **/usr/pat/dead.letter**

  This file is where undeliverable messages are stored.

Options for Reading and Disposing of Mail

Action	*Result*
Press Enter	Display the next message.
Press plus (+)	Same action as *Enter.*
Press hyphen (-)	Display the previous message.
Press d	Delete this message; display next one.
Press p	Display current message again.
Press s{*file*}	Save message and postmark in *mbox* or in the specified file.
Press w{*file*}	Write (save) message but no postmark in *mbox* or in the specified file.
Type m*bjh*	Forward current message to *bjh.*
Press q	Put undeleted mail back into */usr/mail/pat* and exit from *mail.* Press *Ctrl-d* to do the same thing.
Press x	Put mail *unchanged* into */usr/mail/pat* and exit.
Key in !*command*	Run any shell *command* from *mail.*
Press *	Display this list of options.

You should be aware that **d**, as well as several other symbols you use after **?** (question mark), delete the current mail message. They are **s**, **w**, and **m**. If you delete something and immediately realize you should not have, you can use **x** to return the entire mail file to its original state before you began reading it.

When you work your way through the file of messages, you find yourself back at the $ (system prompt).

See *AIX Operating System Commands Reference* for more on **mail**.

Subcommand Choices

You can change the way your messages are displayed on your screen by using one of the following five options with the command:

- **mail -e** prevents your messages from being displayed. Instead, you get a **0** (zero) if you have mail or a **1** (one) if you do not.

- **mail -p** displays your messages without the prompt **?** (question mark) that asks you how to dispose of the mail. You just look at your mail without deleting, copying, or forwarding it.

- **mail -q** lets you quit reading mail whenever you want to by pressing the **Ctrl-Backspace** key sequence.

  Note: If you are not using a keyboard listed in "Before You Begin" on page vi, the results of this key sequence may be unpredictable. See "Remapping Keys" on page 3-66 for information on remapping keys.

- **mail -r** displays your messages in first-in, first-out order instead of last-in, first-out order.

- **mail -fnewfile** saves your mail in **newfile**, for example, instead of in **/usr/pat/mbox**.

Other Modifications

There are two final ways to modify the results of the **mail** command by changing the file **/usr/mail/pat**, using the example of `pat` as your login name.

- If the file is empty, it has no memory space. To reserve space for it, change the permission assignment from the default of *read only* to *read/write* or to *all permissions denied* for other system users.

- Add the following as the first line of the file:

  ```
  forward to pat
  ```

  This causes all messages sent to your mailfile to be forwarded to you at the machine you are logged into instead of requiring you to login to one particular machine.

Remote Mail

The **mail** command permits you to send messages to people on a remote system if you know the name of that system and if there is a connection between it and your own system. Use the **!** (exclamation point) like this example sending to user `rumpole` on system `mack4`:

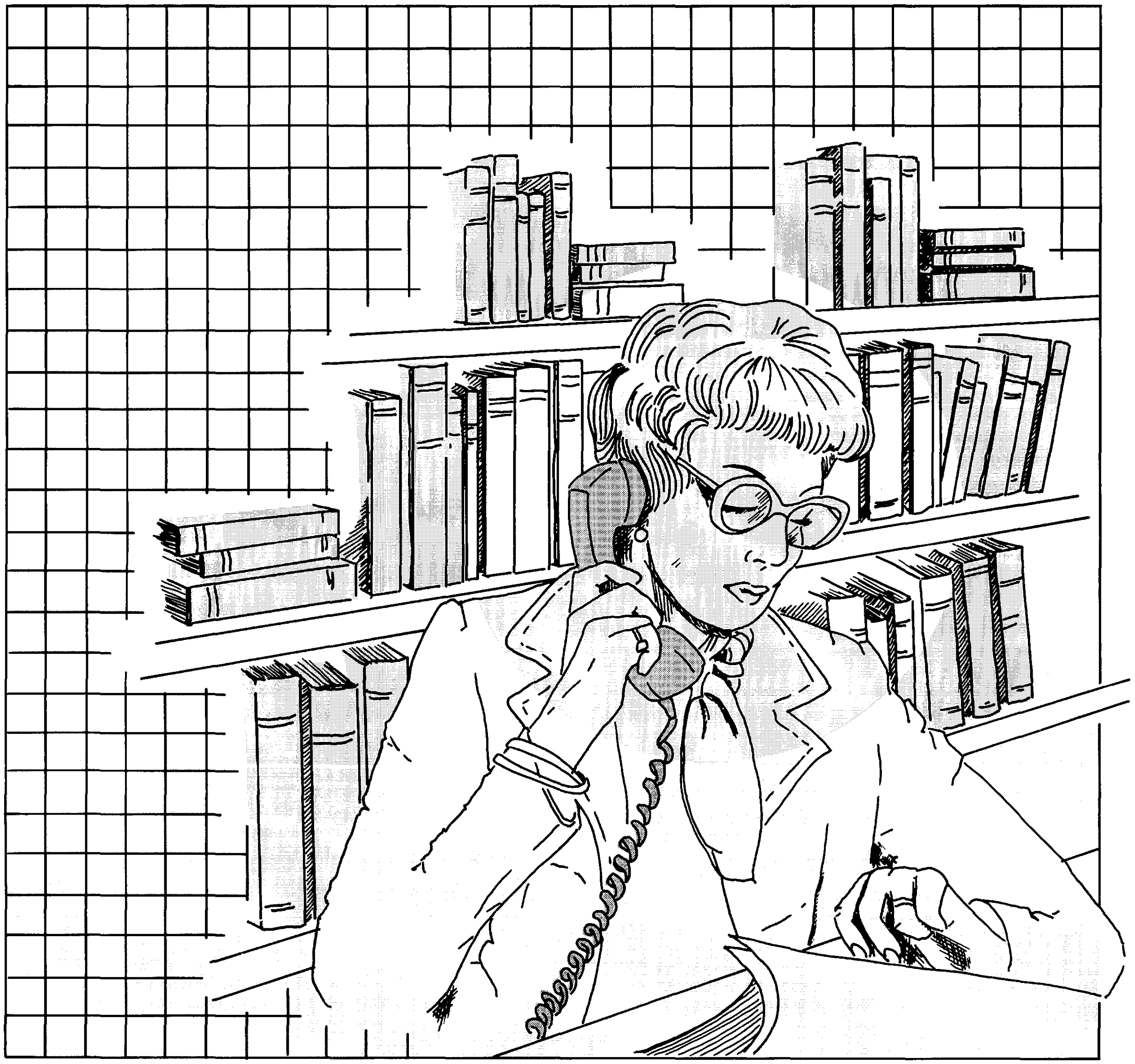
```
mail mack4!rumpole
```

You can go through more than one system like this:

```
mail mack4!aztec!rumpole
```

Even if your own system can not access system `aztec`, you can get a message to it if you can reach `mack4` and `mack4` can get to `aztec`. For more on remote communications, see "Remote Transfers" on page 4-10.

Chapter 3. Communicating: Asynchronous Terminal Emulation



CONTENTS

About This Chapter

This chapter shows you how to communicate with another computer by using Asynchronous Terminal Emulation. You can login to a data base service, run programs, issue commands, use files on the remote system, and record and control the session.

For information on how to install Asynchronous Terminal Emulation, see *Installing and Customizing the AIX Operating System.*

For information on how to set up a port for Asynchronous Terminal Emulation, see *User Setup Guide* and “Getting Started with Ports” on page 5-4.

Creating a Dialing Directory File

The *dialing directory file* is an AIX file that Asynchronous Terminal Emulation uses to store and retrieve telephone numbers of remote systems you can call. When you invoke the **directory** command, Asynchronous Terminal Emulation retrieves and displays the requested **directory** file at your terminal in the form of a Directory Menu.

You may set up several dialing directory files, each with up to 20 entries. You can create or edit a dialing directory file with any AIX file editor. A sample dialing directory file, **dir** located in **/usr/lib**, comes with Asynchronous Terminal Emulation. You can use this sample dialing directory file, containing fictitious numbers, as a model for creating a dialing directory file of your own. Each dialing directory file should have only 20 telephone entries, numbered from 0 to 19. The program ignores entries beyond this amount. If you are using INed, refer to *INed* for help on creating files.

This section contains the following:

- The general format of a dialing directory file
- A representation of your sample dialing directory file
- A representation of your sample dialing directory file in Directory Menu form
- Entry definitions and commands for the dialing directory file.

Dialing Directory File Format

This file uses a special format with eight fields in an entry, each with one or more spaces between them. Each line is called an *entry*, and you can place a maximum of 20 entries in a single dialing directory file. (The program ignores entries beyond this amount). Every dialing directory file must follow this order for the eight fields in a line:

```
Field  1: NAME
Field  2: NUMBER (telephone number)
Field  3: RATE
Field  4: LENGTH
Field  5: STOP (stop bits)
Field  6: PARITY
Field  7: ECHO
Field  8: LF's (line feed)
```

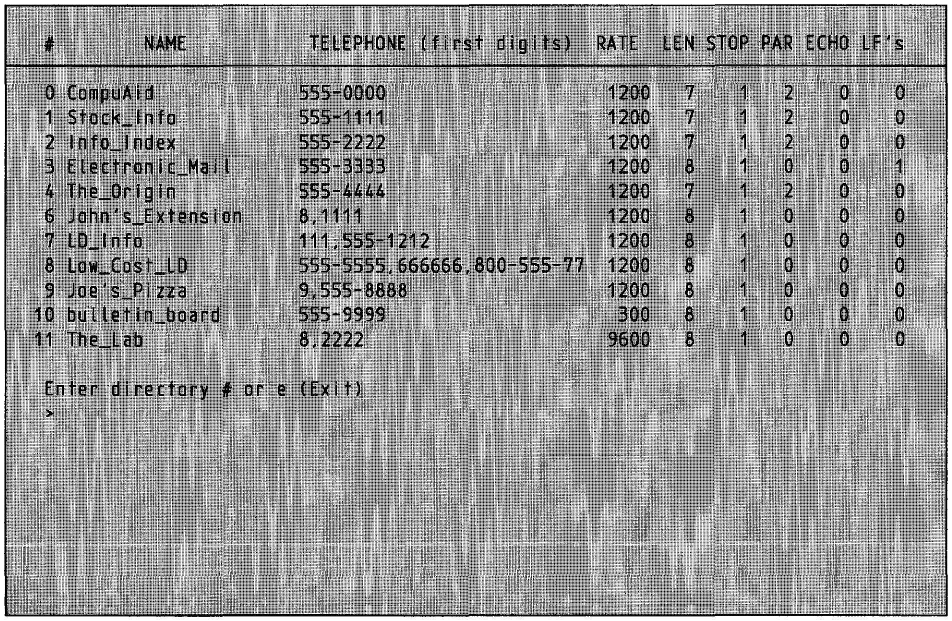
The following is a representation of what your sample directory file (**/usr/lib/dir**) looks like:

```
CompuAid          555-0000                                  1200 7 1 2 0 0
Stock_Info        555-1111                                  1200 7 1 2 0 0
Info_Index        555-2222                                  1200 7 1 2 0 0
Electronic_Mail   555-3333                                  1200 8 1 0 0 1
The_Origin        555-4444                                  1200 7 1 2 0 0
John's_extension  8,1111                                    1200 8 1 0 0 0
LD_Info           111,555-1212                              1200 8 1 0 0 0
Low_Cost_LD       555-5555,666666,800-555-7777              1200 8 1 0 0 0
Joe's_Pizza       9,555-8888                                1200 8 1 0 0 0
bulletin_board    555-9999                                   300 8 1 0 0 0
The_Lab           8,2222                                    9600 8 1 0 0 0
```

The following is a representation of what your sample dialing directory file looks like when displayed at your terminal in Directory Menu form:

Note: When you first start using Asynchronous Terminal Emulation, the sample dialing directory file is available to you. To use it, select the **directory** command from the Asynchronous Terminal Emulation Unconnected Main Menu, and follow the prompts. For more information, see the section on the Unconnected Main Menu later in this chapter.

First, you need to become familiar with the fields in each entry. The fields are also called *commands*, because you can change them.



```
 #        NAME           TELEPHONE (first digits)  RATE  LEN STOP PAR ECHO LF's

 0 CompuAid            555-0000                    1200   7    1   2    0    0
 1 Stock_Info          555-1111                    1200   7    1   2    0    0
 2 Info_Index          555-2222                    1200   7    1   2    0    0
 3 Electronic_Mail     555-3333                    1200   8    1   0    0    1
 4 The_Origin          555-4444                    1200   7    1   2    0    0
 6 John's_Extension    8,1111                      1200   8    1   0    0    0
 7 LD_Info             111,555-1212                1200   8    1   0    0    0
 8 Low_Cost_LD         555-5555,666666,800-555-77  1200   8    1   0    0    0
 9 Joe's_Pizza         9,555-8888                  1200   8    1   0    0    0
10 bulletin_board      555-9999                     300   8    1   0    0    0
11 The_Lab             8,2222                      9600   8    1   0    0    0

 Enter directory # or e (Exit)
 >
```

Directory Entry Commands

name

Definition: The name you assign to a telephone number to help you recognize it.

Options: Any combination of characters not longer than 20 characters. Note that blanks are not valid characters. Use the _ (underscore mark) instead of a blank between names of two or more words, such as `data_bank` rather than `data bank`.

number

Definition: The actual telephone number to be dialed.

Options: Any telephone number that does not exceed 40 digits. Consult your modem user's guide for a list of acceptable digits and characters. For example, if your telephone system is designed such that you need to dial a specified number in order to access an outside line, you may need to put a comma before the telephone number entry, like this: `9,5551111`. Only the first 26 characters (including digits, commas, and dashes) appear in the Directory Menu.

rate

Definition: The bit rate (bits per second) determines the number of characters that can be transmitted per second. You need to select the bit rate according to the capability of the communication line you are using.

Options: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19200. You should use Xon/Xoff signals when you use the higher speeds to insure that no characters are lost. See the section on Xon/Xoff later in this chapter.

length

Definition: The number of bits that make up a character.

Options: 7 or 8.

stop

Definition: Stop bits signal the end of character.

Options: 1 or 2.

parity

Definiton: Parity is a simple means of finding out whether the character you sent was received at the remote system, and vice versa. For example, if parity is *even*, when the number of 1 bits in the character is odd, the parity bit is turned on to make an even number of 1 bits.

Options: 0 = none, 1 = odd, 2 = even.

echo

Definition: Echo is a feature that determines whether or not the characters you type at the keyboard display on your screen.

Options: 0 = off (no echo), 1 = on (echo).

linefeed

Linefeed adds a linefeed character at the end of each line of incoming data being received from the remote system. (The linefeed character is similar in function to the carriage return and newline characters).

Options: 0 = off (no linefeed), 1 = on.

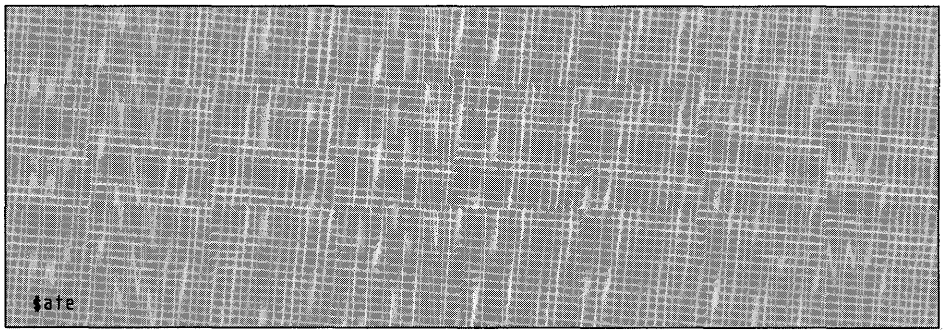
Using Asynchronous Terminal Emulation Menus

This section tells you how to use the Asynchronous Terminal Emulation program menus. It contains a short description of the following:

- How to start and stop using Asynchronous Terminal Emulation
- How to use the Unconnected Main Menu
- How to use the Connected Main Menu
- How to use the Modify Menu
- How to use the Alter Menu.

Getting Started with Asynchronous Terminal Emulation

To use Asynchronous Terminal Emulation you must type the command `ate` on the command line right after the $ (system prompt):



```
$ate
```

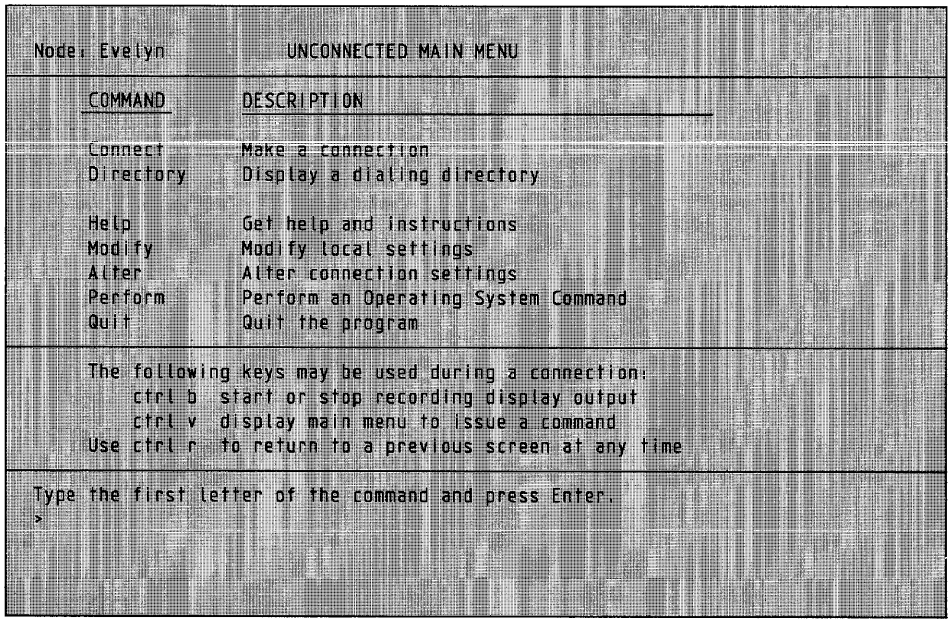
When you finish, the Asynchronous Terminal Emulation Unconnected Main Menu appears on your display screen. The picture on the following page shows you what it looks like.

The Asynchronous Terminal Emulation Unconnected Main Menu

The Asynchronous Terminal Emulation Unconnected Main Menu is the first menu that appears on your screen after you type `ate`. You can access this menu whenever you are using Asynchronous Terminal Emulation and do *not* have a connection to a remote computer established.

This menu lets you establish a connection by using either the **connect** command or the **directory** command. This is what the menu looks like:

Note: To exit from the program, type **q** right after the > (prompt sign).



```
Node: Evelyn              UNCONNECTED MAIN MENU

     COMMAND        DESCRIPTION

     Connect        Make a connection
     Directory      Display a dialing directory

     Help           Get help and instructions
     Modify         Modify local settings
     Alter          Alter connection settings
     Perform        Perform an Operating System Command
     Quit           Quit the program

     The following keys may be used during a connection:
         ctrl b  start or stop recording display output
         ctrl v  display main menu to issue a command
     Use ctrl r  to return to a previous screen at any time

Type the first letter of the command and press Enter.
>
```

The Unconnected Main Menu lets you do the following tasks:

- Initiate a connection to another system by an asynchronous communication link
- Request a phone directory menu that you use to make a connection
- Request help information
- Request the Modify Menu to change the variables that pertain to your local terminal
- Request the Alter Menu to change the variables that affect data transmission
- Run an operating system command
- Stop using Asynchronous Terminal Emulation.

You need to key in commands to the Asynchronous Terminal Emulation program right after the > (prompt character) at the bottom of the menu. You should *not* try to enter values directly into the menu by using the cursor movement keys. Message 062-003 appears if cursor movement controls are part of your input, and the menu does not display correctly. Press the Enter key alone to reprint the menu if you experience this problem.

Command Definitions and Formats

The following information explains how to use the commands associated with the Unconnected Main Menu and shows you what to expect from each command.

connect

The **connect** command allows you to make a connection to a remote computer using the values specified in the Asynchronous Terminal Emulation Alter Menu and Modify Menu. To achieve this you may use a manually-dialed connection, an autodialed connection, or a direct connection.

1. If you are using a switched (telephone) line and want to dial the telphone number yourself, use the following format:

   - Type **c**

   - Press Enter

   - The following prompt displays:

     ```
     Type the phone number of the connection for auto
     dialing, or the name of the port for direct
     connect, and press Enter.  To manually dial a number,
     just press Enter.  To redial the last number (0),
     type 'r' and press Enter.
     >
     ```

     Press Enter

   - The following message displays:

     ```
     062-007  Please dial a telephone number to make the
              requested connection.
     ```

     Manually dial the telephone number of the remote system you want to establish a connection to.

2. If you are using a switched line and want the modem to dial the telephone number for you, do one of the following:

a. To type in all the attributes yourself at the Unconnected Main Menu:

   - Type **c** *telephone_number {port_name}*
   - Press Enter

     Note: The { } (braces) indicate that the port name is optional. For information on port names, see "Getting Started with Ports" on page 5-4.

b. To use the prompts:

   - Type **c**
   - Press Enter
   - The following prompt displays:

     ```
     Type the phone number of the connection for auto
     dialing, or the name of the port for direct
     connect, and press Enter.  To manually dial a number,
     just press Enter.  To redial the last number (0),
      type 'r' and press Enter.
     >
     ```

     Type the telephone number after the > (prompt sign) and press Enter. Use your modem instruction book for information on how to type the number (for example, whether to use spaces or dashes).

   - If the line is busy and you want to redial, type **r** and press Enter. See Example 2 at the end of this section on the **connect** command.

3. If you have a direct connection established to another system, do one of the following:

   a. To type in all the attributes yourself from the Unconnected Main Menu:

      - Type **c** *port_name*

- Press Enter.

b. To use the prompts:

- Type **c**
- Press Enter
- The following prompt displays:

  ```
  Type the phone number of the connection for auto
  dialing, or the name of the port for direct
  connect, and press Enter.  To manually dial a
  number, just press Enter.  To redial the last
  number (0), type 'r' and press Enter.
  >
  ```

  Type the port name after the prompt sign and press Enter.

Examples:

1. SWITCHED LINE WITH MANUAL DIALING

   ```
   >c
   ```

   Press Enter.

   This prompt appears:

   ```
   Type the phone number of the connection for auto
   dialing, or the name of the port for direct
   connect, and press Enter.  To manually dial a
   number, just press Enter.  To redial the last
   number (0), type 'r' and press Enter.
   >
   ```

   Press Enter

   This message appears:

   ```
   062-007  Please dial a telephone number to make the
            requested connection.
   ```

Dial the number, and follow the modem manufacturer's instructions on making the connection.

2. SWITCHED LINE WITH MODEM DIALING

   - WITHOUT PROMPTS

     ```
     >c 5551111
     ```

     Press Enter.

   - WITH PROMPTS

     ```
     >c
     ```

     Press Enter

     This prompt appears:

     ```
     Type the phone number of the connection for auto
     dialing, or the name of the port for direct connect,
     and press Enter.  To manually dial a number, just
     press Enter.  To redial the last number (0), type 'r'
     and press Enter.
     ```

     ```
     >5551111
     ```

     Press Enter

     If the line is busy and you can hear the busy signal, you may want to use control sequence **Ctrl-r** to disconnect. See the section on control key sequences later in this chapter.

     If a connection is not established (busy, no answer, number dialed is not a computer or modem number), you get a message after 45 seconds. See message 062-009 in *Messages Reference*. The program then returns you to the Unconnected Main Menu, and you can select **r** for **redial** to try the number again.

3. DIRECT CONNECTION

   WITHOUT PROMPTS

```
>c tty6
```

   Press Enter.

- WITH PROMPTS

```
>c
```

   Press Enter

   This prompt appears:

```
Type the phone number of the connection for auto
dialing, or the name of the port for direct
connect, and press Enter.  To manually dial
a  number, just press Enter.  To redial
the last number (0), type 'r' and press Enter.
>tty6
```

   Press Enter.

directory

The **directory** command displays a dialing directory file and allows you to establish a connection to a remote computer by selecting one of the entries in it. Asynchronous Terminal Emulation comes with a sample dialing directory file called **/usr/lib/dir**. The first time you use Asynchronous Terminal Emulation, the sample dialing directory file appears when you select the **directory** command from the Unconnected Main Menu. Every time after this, the current dialing directory file is the last directory you used. You may select a directory entry in one of two ways, with or without prompts.

1. To use the **directory** command with the aid of prompts:

   - Type **d**
   - Press Enter
   - The following message appears:

     ```
     Type the file name of the directory you want to
     display and press Enter.  To use the current
     directory (/usr/lib/dir), just press Enter.
     >
     ```

     Do one of the following:

     – Type the file name of the directory you want and press Enter, or

     – Press Enter to use the current directory.

2. To use the **directory** command without the aid of prompts:

   - Type **d** *directory_name*
   - Press Enter.

Examples:

1. WITH PROMPTS

- `>d`
- Press Enter

  This message appears:

  ```
  Type the file name of the directory you want to
  display and press Enter.  To use the current
  directory (/usr/lib/dir), just press Enter.
  >
  ```

- Either press Enter or type the name of the dialing directory file you want to use, such as `bulletin_bds`, and then press Enter.

2. WITHOUT PROMPTS

- `>d bulletin_bds`
- Press Enter.

help

The **help** command displays a description and instructions for each command on the Unconnected Main Menu and on the Connected Main Menu. When you use the **help** command you may enter several command initials at the same time. Help descriptions appear in the order you request them. You get each command description and instruction displayed individually on your screen. Just press Enter to view the next command description. Valid command initials are:

c	connect
d	directory
s	send
r	receive
b	break
t	terminate
h	help
m	modify
a	alter
p	perform
q	quit

To use the **help** command select one of the following formats:

1. To access the general help screen:

   - Type **h**
   - Press Enter.

2. To view a help message for a particular command (or commands) while you are in the Unconnected Main Menu or in the Connected Main Menu:

   - Type **h** *command_initial* {*command_initial*}

     The braces, { }, indicate that you may key in as many command initials as you wish.

   - Press Enter.

Example:

```
>h c s q
```

This gives you help on **connect**, **send**, and **quit**.

modify

The **modify** command is used to change characteristics that pertain to your terminal. When you use the **modify** command, you may enter several command initials at the same time. (See the section on the Modify Menu in this chapter for more information). Press the Enter key once to leave the Modify Menu when you are finished.

1. To display the Asynchronous Terminal Emulation Modify Menu:

   - Type **m**

   - Press Enter

   - Type *command_initial (value)* {*command_initial*}

     Note: Only one command, **name**, takes a value. All other commands are *switches*, or on-off settings. The system produces an invalid command message if you use a value with them. See message 062-003 in *Messages Reference.*

     Note: The ( ) (parentheses) indicate that a value following **name** is optional. The { } (braces) indicate that you may repeat the contents as many times as you want to.

- Press Enter again to leave the Modify Menu when you are finished.

2. To change one or more of the values from the Asynchronous Terminal Emulation Modify Menu without actually viewing the menu:

   - Type **m** *command_initial (value)* {*command_initial*}

     Note: The ( ) (parentheses) indicate that a value following **name** is optional. The { } (braces) indicate that you may repeat the contents as many times as you want to.

   - Press Enter.

Example:

Change the **name** and **write** values.

- `>m n goodfile w`
- Press Enter.

alter

The **alter** command lets you change data transmission and autodialing characteristics. When you use this command you may list several command initials at the same time. (See the Alter Menu later in this section for more information). Press Enter once to leave the Alter Menu when you are finished.

1. To display the Asynchronous Terminal Emulation Alter Menu:

   - Type **a**
   - Press Enter
   - Type *command_initial value* {*command_initial value*}
   - Press Enter again to leave the Alter Menu.

2. To change one or more of the values from the Asynchronous Terminal Emulation Alter Menu without actually viewing the menu:

   - Type **a** *command_initial value* {*command_initial value*}
   - Press Enter.

Example:

Change **length**, **parity**, and **transfer** values.

- `>a l 7 p 1 t x`
- Press Enter.

perform

The **perform** command allows you to run an AIX shell command from Asynchronous Terminal Emulation. Select one of the following formats:

1. To use the **perform** command with the aid of prompts:

   - Type **p**
   - The following prompt displays:

     ```
     Type an operating system command and press Enter.
     >
     ```

     Type the command you wish to run after the prompt and press Enter.

2. To run a shell command directly from the Unconnected Main Menu:

   - Type **p** *shell_command*
   - Press Enter.

Example:

Display a file named **cheerfile** on your screen.

- `>p cat cheerfile`
- Press Enter.

Note: For information on the **cat** command, see *AIX Operating System Commands Reference.*

quit

The **quit** command exits the Asynchronous Terminal Emulation program, and returns you to the operating system. To use this command, do the following:

- Type **q**
- Press Enter.

Example:

```
>q
```

$ (system prompt)

Control Key Sequences

Note: To perform a control key sequence, first hold down the **Ctrl** key and then press the key named after the - (hyphen or minus sign). If you are not using a keyboard listed in "Before You Begin" on page vi, results may be unpredictable. See "Remapping Keys" on page 3-65 for information on remapping keys.

Ctrl-b

This sequence allows you to start or stop saving the data displayed on your screen in the file named in the Modify Menu while you are in an active session. You *initiate* the save action the first time you perform this control key sequence. You *reverse* the action the second time you press the two keys. This back-and-forth feature is called *switching on* and *switching off*. You can switch **Ctrl-b** on or off only when Asynchronous Terminal Emulation is *connected* and you are *not* transferring files.

Pressing **Ctrl-b** and Enter from the Connected Main Menu sends the **Ctrl-b** sequence to a remotely connected system that is running Asynchronous Terminal Emulation.

You receive message 062-003 if you press **Ctrl-b** from the Unconnected Main Menu. Since control characters do not normally display, the message may indicate that you entered " ".

Ctrl-v

The **Ctrl-v** sequence lets you access the Connected Main Menu from the connection screen when you use it in the connected state. Pressing **Ctrl-v** and Enter from the Connected Main Menu sends the **Ctrl-v** sequence to a remotely connected system that is running Asynchronous Terminal Emulation.

You receive message 062-003 if you press **Ctrl-v** from the Unconnected Main Menu. Since control characters do not normally display, the message may indicate that you entered " ".

Ctrl-r

This sequence returns you to a previously displayed screen.

Examples:

FROM THE UNCONNECTED STATE

Where You Are	***What Happens when You Use Ctrl-r***
Unconnected Main Menu	Ignored
Directory Menu	Returns to Unconnected Main Menu
Help Menu	Returns to Unconnected Main Menu
Modify Menu	Returns to Unconnected Main Menu
Alter Menu	Returns to Unconnected Main Menu
During a perform	Displays the message: Press any key
During a quit	Ignored
Making a connection	Terminates the connection and returns to the Unconnected Main Menu or the Directory Menu, depending on which one you were using to make the connection.

FROM THE CONNECTED STATE

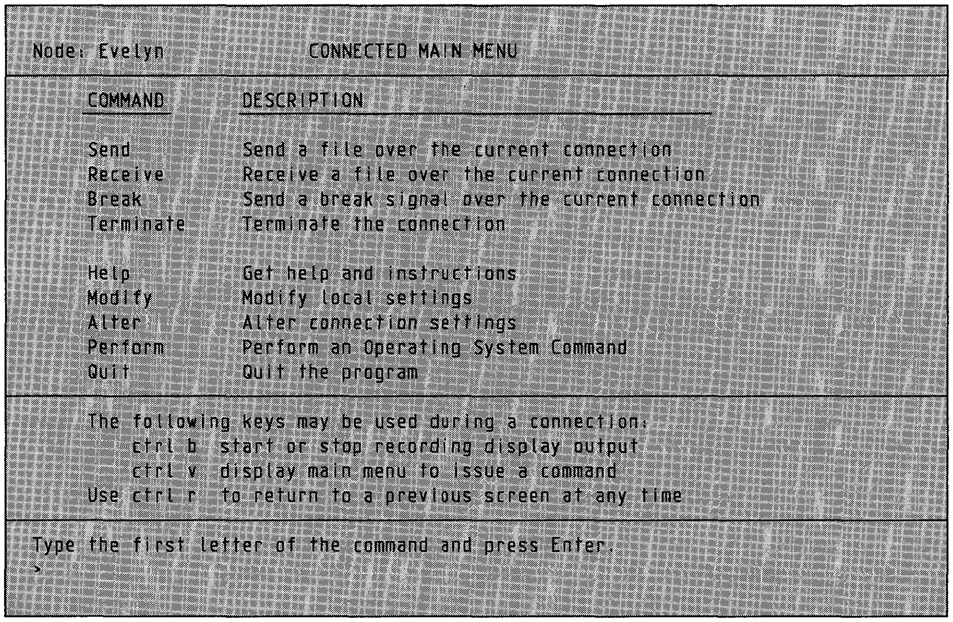
Where You Are	*What Happens When You Use Ctrl-r*
Connection Screen	Displays the message: Disconnect (y/n)?
Connected Main Menu	Returns to connection screen
Help Menu	Returns to connection screen
Modify Menu	Returns to connection screen
Alter Menu	Returns to connection screen
During a perform	Displays the message: Press any key
During a quit	Ignored
During a terminate	Ignored
During a break	Ignored
During a send	Returns to connection screen
During a receive	Returns to connection screen

See the section on the Connected Main Menu for further information on this key.

The Asynchronous Terminal Emulation Connected Main Menu

You can invoke the Connected Main Menu by typing **Ctrl-v** any time after you have established a connection to the remote system. There are additional functions on this menu that are not on the Unconnected Menu. The Connected Main Menu looks like this:

Note: Select **q** to exit from the Asynchronous Terminal Emulation program.



```
Node: Evelyn                CONNECTED MAIN MENU

     COMMAND       DESCRIPTION

     Send          Send a file over the current connection
     Receive       Receive a file over the current connection
     Break         Send a break signal over the current connection
     Terminate     Terminate the connection

     Help          Get help and instructions
     Modify        Modify local settings
     Alter         Alter connection settings
     Perform       Perform an Operating System Command
     Quit          Quit the program

     The following keys may be used during a connection:
         ctrl b  start or stop recording display output
         ctrl v  display main menu to issue a command
     Use ctrl r  to return to a previous screen at any time

Type the first letter of the command and press Enter.
>
```

The Connected Main Menu lets you to do the following tasks:

- Send a file to the system connected to your terminal
- Receive a file from the system connected to your terminal
- Interrupt an established connection by sending a break sequence
- Terminate an established connection
- Request help information
- Request the Modify Menu and change the variables that pertain to your local terminal
- Request the Alter Menu and change the variables that affect data transmission
- Run an operating system command
- Stop using Asynchronous Terminal Emulation.

Command Definitions and Formats

send

The **send** command lets you send a file from your local system to the remote system you are connected to. Select one of the following formats:

1. To use the **send** command with the aid of prompts:

   - Type s
   - Press Enter
   - The following prompt appears:

     ```
     Type the name of the file you wish to send and
     press Enter.  To use the last file name (   ),
     just press Enter.
     >
     ```

     Respond by entering the name of the file you want to send.

2. To use the **send** command without the aid of prompts:

   - Type s *filename*, using the name of an existing file in the *filename* field
   - Press Enter.

Example:

WITHOUT PROMPTS

- `>s myfile`
- Press Enter.

receive

The **receive** command allows you to obtain a file from the remote system you are connected to. Select one of the following formats:

1. To use the **receive** command with the aid of prompts:

   - Type **r**
   - Press Enter
   - The following prompt appears:

     ```
     Type the name of the file you wish to store the received
     data in and press Enter.  To use the last file name (   ),
     just press Enter.
     >
     ```

     Press Enter or type the name of the file where you want the incoming data to go.

2. To use the **receive** command without the aid of prompts:

   - Type **r** *filename*
   - Press Enter.

Example:

WITHOUT PROMPTS

- `>r incomingfile`
- Press Enter.

break

The **break** command allows you to send a break signal (0 bits for .25 seconds) to the remote computer.

Warning: This may either interrupt the current activity on the remote computer or IT MAY DISCONNECT THE CURRENT SESSION. You may lose data.

To use the **break** command, do the following:

- Type **b**
- Press Enter.

terminate

The **terminate** command disconnects an active session and returns you to the Unconnected Main Menu.

- Type **t**
- Press Enter.

help

The **help** command displays a description and instructions for each command on the Unconnected Main Menu and the Connected Main Menu. When you enter the **help** command you are allowed to enter several command initials at the same time. Help descriptions appear on the screen in the order you request them. Each command description and instruction comes into view individually. Press Enter to view the next command description.

Valid command initials are:

c	connect
d	directory
s	send

r	receive
b	break
t	terminate
h	help
m	modify
a	alter
p	perform
q	quit

To use the **help** command select one of the following formats:

1. To access the general help screen:
   - Type **h**
   - Press Enter.
2. To view a help message for a particular command or commands while you are in the Unconnected Main Menu or in the Connected Main Menu:
   - Type **h** *command_initial* {*command_initial*}

     The braces, { }, indicate that you may key in as many command initials as you wish.
   - Press Enter

Example:

```
>h r t
```

This gives you help on the **receive** and **terminate** commands.

modify

The **modify** command changes characteristics that pertain to your terminal. When you use the **modify** command you may list several command initials at the same time. (See the section on the Modify Menu in this chapter for more information).

1. To display the Asynchronous Terminal Emulation Modify Menu:

   - Type **m**
   - Press Enter
   - Type *command_initial (value)* {*command_initial*}
   - Press Enter again to leave the Modify Menu when you are finished.

2. To change one of the values from the Asynchronous Terminal Emulation Modify Menu without actually viewing the menu:

   - Type **m** *command_initial* (*value*) {*command_initial*}
   - Press Enter.

     Note: The parentheses, ( ), indicate that the value is optional. But only the **name** command can take a value. All others are switches. The braces, { }, indicate that you may enter as many command initials as you wish.

Example:

WITHOUT VIEWING THE MODIFY MENU

- `>m n nicefile l`
- Press Enter.

  This lets you use `nicefile` as the capture file and turns the **linefeeds** value on.

alter

The **alter** command lets you change data transmission and autodialing characteristics. When you use the **alter** command you may list several command initials at the same time. (See the Alter Menu in this chapter for more information).

1. To display the Asynchronous Terminal Emulation Alter Menu and change the values:
   - Type **a**
   - Press Enter
   - Type *command_initial value* {*command_initial value*}
   - Press Enter again to leave the Alter Menu.
2. To change the values from the Asynchronous Terminal Emulation Alter Menu without actually viewing the menu:
   - Type **a** *command_initial value* {*command_initial value*}
   - Press Enter.

Example:

Change the **rate**, **wait**, and **attempt** values.

- `>a r 9600 w 5 a 1`
- Press Enter.

perform

The **perform** command lets you run an operating system shell command from Asynchronous Terminal Emulation. Select one of the following formats:

1. To use the **perform** command with the aid of prompts:

   - Type **p**
   - The following prompt displays:

     ```
     Type an operating system command and press Enter.
     >
     ```

   Type the command you want to run after the prompt and press Enter.

2. To run a shell command directly from the Unconnected Main Menu:

   - Type **p** *shell_command*
   - Press Enter.

Example:

Display a file named **cheerfile** on your screen.

- `>p cat cheerfile`
- Press Enter.

Note: For information on the **cat** command, see *AIX Operating System Commands Reference.*

quit

The **quit** command exits the Asynchronous Terminal Emulation program, and returns you to the operating system. To use this command, do the following:

- Type **q**
- Press Enter.

Example:

```
>q
```

$ (system prompt)

Control Key Sequences

Note: To perform a control key sequence, first hold down the **Ctrl** key and then press the key named after the - (hyphen or minus sign). If you are not using a keyboard listed in "Before You Begin" on page vi, results may be unpredictable. See "Remapping Keys" on page 3-65 for information on remapping keys.

Ctrl-b

This sequence allows you to start or stop saving the data displayed on your screen in the file named in the Modify Menu while you are in an active session. You *initiate* the save action the first time you perform this control key sequence. You *reverse* the action the second time you press the two keys. This back-and-forth feature is called *switching on* and *switching off*. You can switch **Ctrl-b** on or off only when Asynchronous Terminal Emulation is *connected* and you are *not* transferring files.

Pressing **Ctrl-b** and Enter from the Connected Main Menu sends the **Ctrl-b** sequence to a remotely connected system that is running Asynchronous Terminal Emulation.

You receive message 062-003 if you press **Ctrl-b** from the Unconnected Main Menu. Since control characters do not normally display, the message may indicate that you entered *" "*.

Ctrl-v

The **Ctrl-v** sequence lets you access the Connected Main Menu from the connection screen when you use it in the connected state. Pressing **Ctrl-v** and Enter from the Connected Main Menu sends the **Ctrl-v** sequence to a remotely connected system that is running Asynchronous Terminal Emulation.

You receive message 062-003 if you press **Ctrl-v** from the Unconnected Main Menu. Since control characters do not normally display, the message may indicate that you entered *" "*.

Ctrl-r

This sequence returns you to a previously displayed screen. You can use this sequence to stop a file transfer when one is in progress and to display a prompt that asks if you want to disconnect.

Example:

FROM THE CONNECTED STATE

Where You Are	*What Happens when You Use Ctrl-r*
Connected Main Menu	Returns to the connection screen
Help Menu	Returns to the connection screen
Modify Menu	Returns to the connection screen
Alter Menu	Returns to the connection screen
During a perform	Displays a message: Press any key
During a quit	Ignored
During a terminate	Ignored
During a break	Ignored
During a connection	Displays a message asking if you want to terminate the connection. If you choose the yes option, either the Unconnected Main Menu or the Directory Menu appears, depending on which one you used to make the connection. If you choose the no option, the connection screen appears.
While sending a file	File transfer ends and the connection screen appears
While receiving a file	File transfer ends and the connection screen appears

See the previous section on the Unconnected Main Menu for further details on the use of this key sequence.

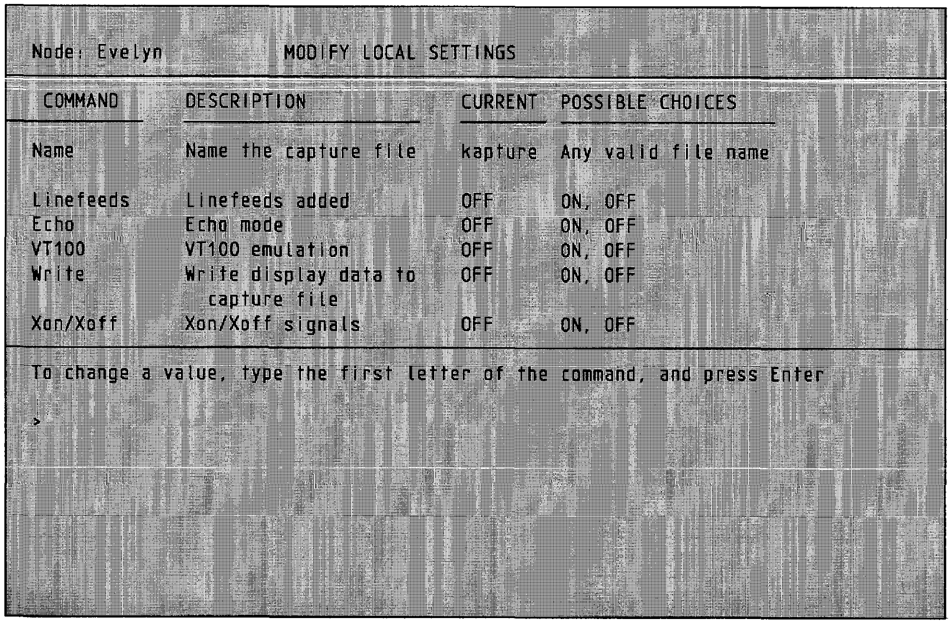
The Asynchronous Terminal Emulation Modify Menu

You can invoke the Modify Menu from either the Unconnected Main Menu or the Connected Main Menu. This menu allows you to modify local terminal variables.

To change the current value of the **name** command, type the command initial, a space, and then the name of the capture file you want to store the information in. Press Enter. (See Example 1 at the end of this section). To change any of the other values, type the command initial and press Enter.

You may change as many values at one time as you like. (See Example 2 at the end of this section).

This is what the Modify Menu looks like:



```
Node: Evelyn            MODIFY LOCAL SETTINGS

 COMMAND      DESCRIPTION               CURRENT  POSSIBLE CHOICES

Name          Name the capture file     kapture  Any valid file name

Linefeeds     Linefeeds added           OFF      ON, OFF
Echo          Echo mode                 OFF      ON, OFF
VT100         VT100 emulation           OFF      ON, OFF
Write         Write display data to     OFF      ON, OFF
                capture file
Xon/Xoff      Xon/Xoff signals          OFF      ON, OFF

To change a value, type the first letter of the command, and press Enter

>
```

The Modify Menu lets you do the following tasks:

- Specify the name of the capture file
- Add a linefeed character at the end of each line of data being received from the remote computer
- Set echo mode to on or off
- Make Asynchronous Terminal Emulation function like a DEC VT100 terminal with the remote system
- Send incoming data to a capture file as well as to the display during a connection
- Permit an Xon/Xoff (transmitter on/transmitter off) signal.

Command Definitions

name

Definition: The **name** command specifies the file that incoming data goes to if the **write** command is on, or if the **Ctrl-b** key sequence is used during a connection.

Options: Any valid file name that is less than 41 characters long. (But only the first 18 characters are displayed in the Modify Menu). The original default is set to *kapture.*

linefeeds

Definition: When the **linefeeds** command is on, a linefeed character is added after every carriage return it detects in the incoming data stream.

Options: On, Off.

echo

Definition: When you type a key on the keyboard, that character displays on your screen. When you are operating with a modem that supports duplex capability, each character sent to the remote computer returns to the local terminal and appears on your screen, thus showing you the character twice. The **echo** off command allows you to suppress the display of the characters you type, thus making them appear only once with full duplex.

Options: On, Off.

VT100

Definition: DEC VT100 emulation allows you to use DEC VT100 codes during a connection. When the **VT100** command is on, the local console emulates a DEC VT100 terminal with the remote system. When off, the local console functions like an IBM RT PC with the remote system.

Options: On, Off.

Note: Use this command only if you have the keyboard that came with your system since some keys are remapped. Any other keyboard gives you unpredictable results if you switch on **VT100**. Some DEC VT100 codes are not supported. Examples of these are 132 column, double height and width lines, origin mode, and scrolling regions.

write

Definition: When the **write** command is on, Asynchronous Terminal Emulation routes the incoming data to a file, specified by the **name** command, as well as to the display. When capturing a file, Asynchronous Terminal Emulation converts any carriage return/linefeed combinations to linefeeds before writing them into the capture file.

Options: On, Off.

Xon/Xoff

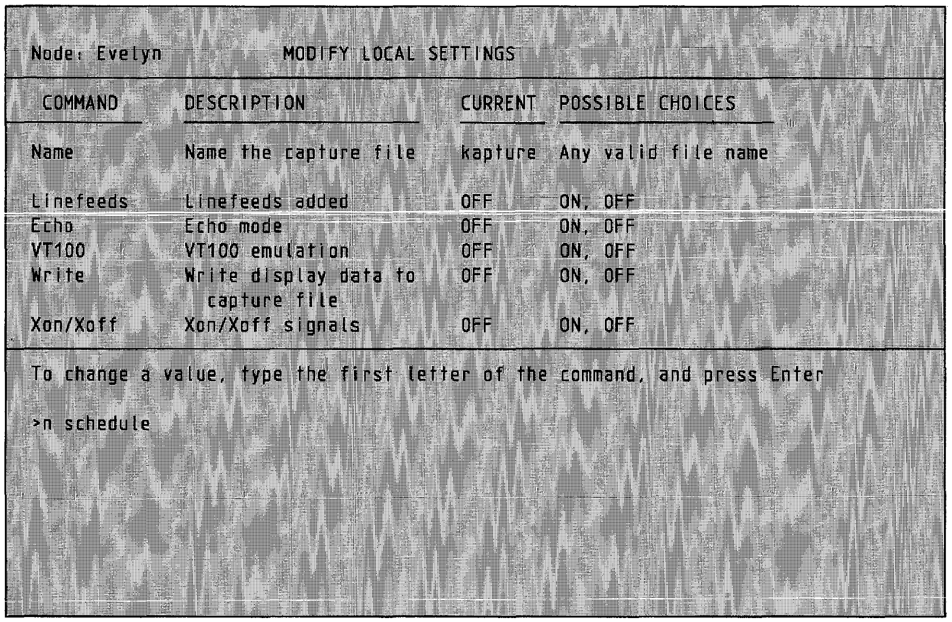
Definition: When the **Xon/Xoff** command is on, the system controls data transmission at the port in the following manner:

- When the system receives an Xoff, it suspends the transmitting of data until it receives an Xon.
- When the system receives an Xon, it resumes sending data if previously suspended.
- The system sends an Xoff when its receive buffer reaches a near full condition.
- The system sends an Xon when the receive buffer is no longer full.

Example 1:

The following is a representation of the Modify Menu. The current values shown here are the program default values you get each time you start using Asynchronous Terminal Emulation. In this example, notice that `n schedule` appears on the screen right after the prompt sign. This is where you type in the new name value. This causes the current value of the **name** command to change from kapture to schedule. Any information written to a capture file during this session now goes into the schedule file.

Note: Press Enter again to leave the Modify Menu.

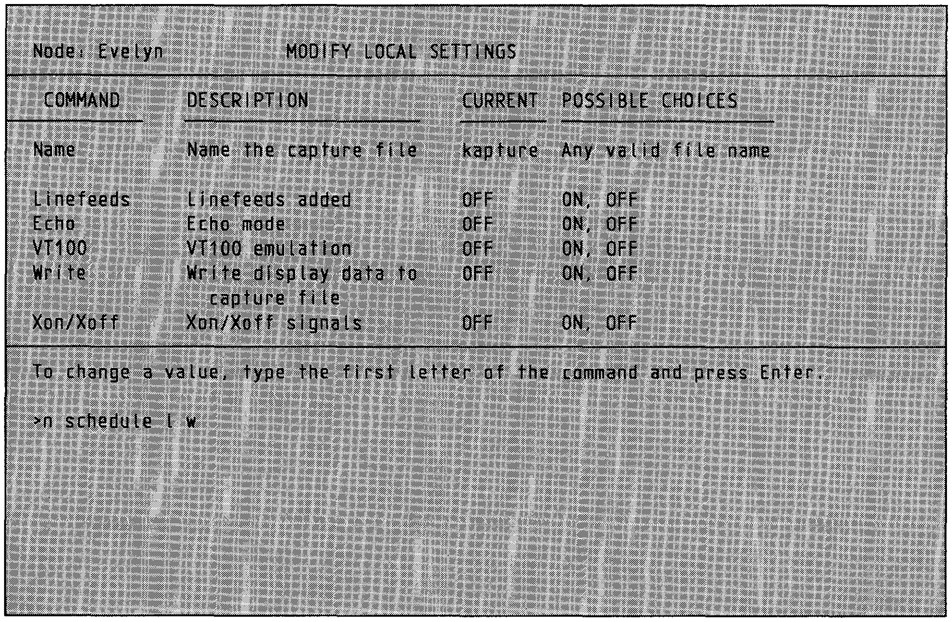


```
Node: Evelyn            MODIFY LOCAL SETTINGS

 COMMAND       DESCRIPTION               CURRENT  POSSIBLE CHOICES

Name          Name the capture file     kapture  Any valid file name

Linefeeds     Linefeeds added           OFF      ON, OFF
Echo          Echo mode                 OFF      ON, OFF
VT100         VT100 emulation           OFF      ON, OFF
Write         Write display data to     OFF      ON, OFF
                capture file
Xon/Xoff      Xon/Xoff signals          OFF      ON, OFF

To change a value, type the first letter of the command, and press Enter

>n schedule
```

Example 2:

The following is a representation of the Modify Menu. The current values shown here are the program default values you get each time you start using Asynchronous Terminal Emulation. In the example, note that `n schedule l w` appears on the screen after the prompt sign. This is where you type in the new values. The current value of **name** changes to schedule, and the **linefeed** and **write** commands change to on.

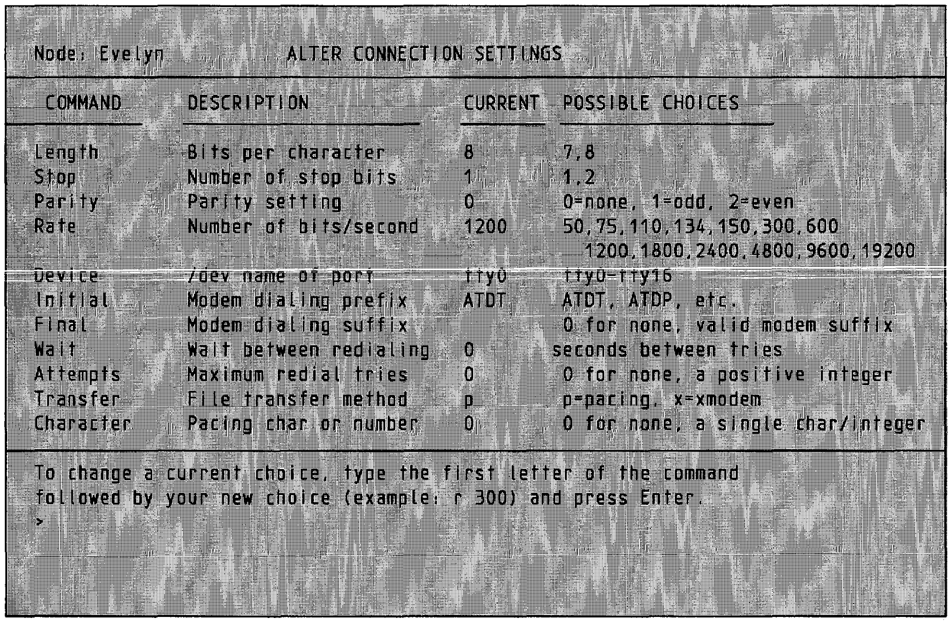
Note: Press Enter again to leave the Modify Menu.



```
Node: Evelyn          MODIFY LOCAL SETTINGS

 COMMAND     DESCRIPTION              CURRENT  POSSIBLE CHOICES

Name         Name the capture file    kapture  Any valid file name

Linefeeds    Linefeeds added          OFF      ON, OFF
Echo         Echo mode                OFF      ON, OFF
VT100        VT100 emulation          OFF      ON, OFF
Write        Write display data to    OFF      ON, OFF
               capture file
Xon/Xoff     Xon/Xoff signals         OFF      ON, OFF

To change a value, type the first letter of the command and press Enter.

>n schedule l w
```

The Asynchronous Terminal Emulation Alter Menu

You can invoke the Alter Menu from either the Unconnected Main Menu or the Connected Main Menu. This menu allows you to specify or alter data transmission characteristics. To change any of the current values, type the command initial, a space, and then the new value after the prompt sign on the Alter Menu. Press Enter. Press Enter again to leave the Alter Menu. You may change as many values at one time as you like. This is what the Alter Menu looks like:



```
 Node: Evelyn          ALTER CONNECTION SETTINGS

  COMMAND     DESCRIPTION              CURRENT  POSSIBLE CHOICES

 Length       Bits per character       8        7,8
 Stop         Number of stop bits      1        1,2
 Parity       Parity setting           0        0=none, 1=odd, 2=even
 Rate         Number of bits/second    1200     50,75,110,134,150,300,600
                                                  1200,1800,2400,4800,9600,19200
 Device       /dev name of port        tty0     tty0-tty16
 Initial      Modem dialing prefix     ATDT     ATDT, ATDP, etc.
 Final        Modem dialing suffix              0 for none, valid modem suffix
 Wait         Wait between redialing   0       seconds between tries
 Attempts     Maximum redial tries     0        0 for none, a positive integer
 Transfer     File transfer method     p        p=pacing, x=xmodem
 Character    Pacing char or number    0        0 for none, a single char/integer

 To change a current choice, type the first letter of the command
 followed by your new choice (example: r 300) and press Enter.
 >
```

The Asynchronous Terminal Emulation Alter Menu lets you do the following tasks:

- Set the number of bits in the character
- Set the number of stop bits to be appended to the character
- Specify the parity setting
- Choose the bit rate
- Select the name of the port you wish to use to establish a connection
- Specify the dialing prefix compatible with your modem
- Choose the dialing suffix compatible with your modem
- Set the **redial** command to keep dialing every *n* seconds until a connection is made
- Set the maximum number of times Asynchronous Terminal Emulation attempts to redial
- Specify the type of transfer method to be used during the connection
- Select the pacing character or delay time to be used.

Command Definitions

length

Definition: The **length** command specifies the number of bits that are in the data characters being transmitted or received. The length specified must match the length expected by the remote system.

Options: 7 or 8.

stop

Definition: The **stop** command sets the number of stop bits appended to the character being sent or received. The number of stop bits specified must match the number of stop bits used by the remote system.

Options: 1 or 2.

parity

Definition: The **parity** command is a simple means of providing error detection on a character basis. For more on parity, see **parity** in the Directory Entry Options earlier in this chapter.

Options: 0=none, 1=odd, 2=even.

rate

Definition: The **rate** command specifies the speed at which the bits are transmitted or received. The speed must match the speed of your modem and the speed expected by the remote system.

Options: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19200.

device

Definition: The **device** command specifies the name of the asynchronous port that should be used to connect to a remote system.

Options: tty0 - tty16, or locally created port names. Only the first 8 characters are displayed in the Alter Menu.

initial

Definition: The **initial** command specifies the dial command that must precede the telephone number when you are auto dialing with a modem. Consult your modem user's guide for the proper dial commands.

Options: ATDT, ATDP, etc. Only the first 8 characters are displayed in the Alter Menu.

final

Definition: The **final** command specifies the dial command that needs to follow the telephone number when you are auto dialing with a modem. Consult your modem user's guide for the proper dial command.

Options: 0 = none, valid modem suffix. Only the first 8 characters are displayed in the Alter Menu.

wait

Definition: The **wait** command tells the Asynchronous Terminal Emulation program to wait *n* seconds between redialing attempts. If **attempts** is set to 0, no redial attempt occurs. The wait period does not begin until the connection attempt times out or until you interrupt it.

Options: 0 = none, a positive integer.

attempts

Definition: The **attempts** command specifies the maximum number of times Asynchronous Terminal Emulation tries to redial if it cannot make the initial connection.

Options: 0=none, a positive integer.

transfer

Definition: The **transfer** command specifies the asynchronous protocol to be used when transferring files from one system to another.

- Pacing. This is a file transfer protocol that controls the rate of transmission either by waiting for a specified character before the next transmission or by waiting for a certain number of seconds to be left between transmissions. It may help prevent systems from losing data when the transmission blocks are too large or are sent too quickly for the system to process them.
- XMODEM. This is an 8- bit file transfer protocol used to detect data transmission errors and retransmit data that had errors. It provides a higher degree of data transmission integrity than most other modes of asynchronous transmissions. For more information, see "Xmodem File Transfer" on page 3-70.

Options: p = pacing, x = xmodem.

character

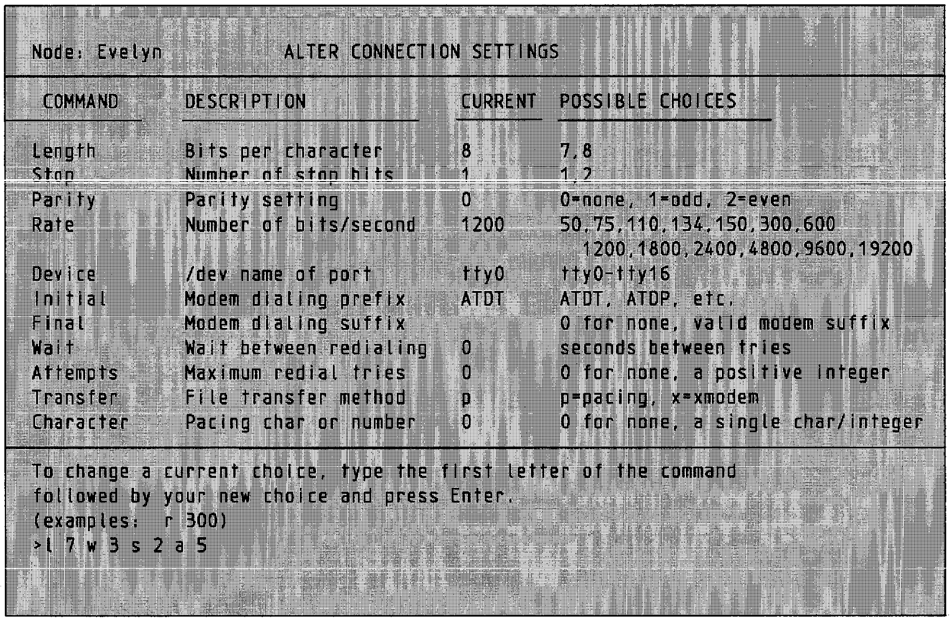
Definition: The **character** command specifies the type of pacing protocol to be used.

0	The 0 (zero) setting means that data is transmitted without pacing.
One character	Character pacing allows you to select the character that is to be recognized as the signal to transmit a line.
One integer	Integer pacing specifies how many seconds to wait between each line that is transmitted when sending a file.

Example:

The following is an example of an Alter Menu. The command initials and values typed in at the prompt sign cause the following actions:

- The **length** changes to 7
- The **wait** value changes to 3
- The **stop** value changes to 2
- The **attempts** value changes to 5.



```
Node: Evelyn           ALTER CONNECTION SETTINGS

 COMMAND      DESCRIPTION                CURRENT  POSSIBLE CHOICES

Length        Bits per character         8        7,8
Stop          Number of stop bits        1        1,2
Parity        Parity setting             0        0=none, 1=odd, 2=even
Rate          Number of bits/second      1200     50,75,110,134,150,300,600
                                                    1200,1800,2400,4800,9600,19200
Device        /dev name of port          tty0     tty0-tty16
Initial       Modem dialing prefix       ATDT     ATDT, ATDP, etc.
Final         Modem dialing suffix                0 for none, valid modem suffix
Wait          Wait between redialing     0        seconds between tries
Attempts      Maximum redial tries       0        0 for none, a positive integer
Transfer      File transfer method       p        p=pacing, x=xmodem
Character     Pacing char or number      0        0 for none, a single char/integer

To change a current choice, type the first letter of the command
followed by your new choice and press Enter.
(examples:  r 300)
>l 7 w 3 s 2 a 5
```

Press Enter again to leave the Alter Menu.

The Default File

This section is for the system user with some computer and data processing experience. The Asynchronous Terminal Emulation default file **ate.def** is an AIX file that contains default values for the Asynchronous Terminal Emulation program.

This section contains:

- A representation of the default file as it originally appears
- A description of how the file is created
- Instructions on how to modify the default file to suit your own needs.

The Original Default File

When Asynchronous Terminal Emulation is first installed, the **ate.def** file does not exist. You create it automatically the first time you run the program. The original defaults and the format of the file created by Asynchronous Terminal Emulation look like this:

```
LENGTH          8
STOP            1
PARITY          0
RATE            1200
DEVICE          tty0
INITIAL         ATDT
FINAL
WAIT            0
ATTEMPTS        0
TRANSFER        p
CHARACTER       0
NAME            kapture
LINEFEEDS       0
ECHO            0
VT100           0
WRITE           0
XON/XOFF        0
DIRECTORY       /usr/lib/dir
CAPTURE_KEY     2
MAINMENU_KEY    22
PREVIOUS_KEY    18
```

How to Change Your Default Values

You can edit the default file, **ate.def**, by using any AIX file editor. To change your default values, edit your **ate.def** file in the following manner:

1. Delete all variables which you *do not* want to change. (You do not have to do this, but it helps you keep straight about what you are changing and what you are not changing). Only the defaults that you want to change need to appear in the **ate.def** file.

2. Change the values of the remaining variables to what you want them to be.

When editing your default file always remember:

- Only variables with default values *different* from the system's default values need to appear in the file. You may delete the remaining variables from the file.

- Variable names may be in uppercase or lowercase, but you must spell them *exactly* as they appear in the sample default file on the preceding page.

  Note: If you enter an incorrect value, you receive a system message, and the program continues. It uses its default value in place of the value in the file.

- Only one variable per line is permitted.

From now on when you start using Asynchronous Terminal Emulation, it looks for a file in the current directory named **ate.def**. Then it reads the values in the file and changes the system's defaults to the current defaults you set earlier.

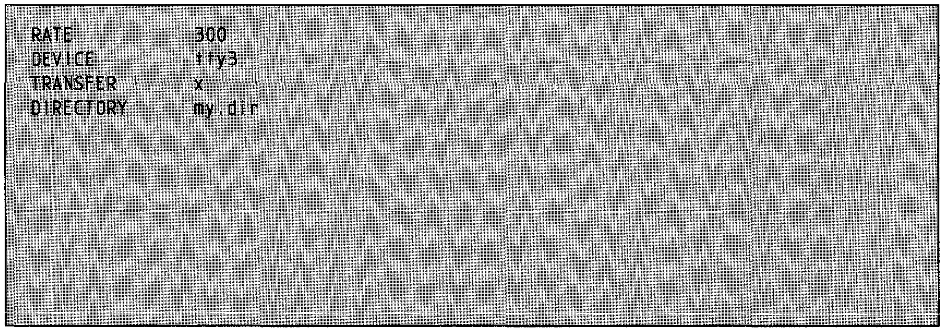
Example

You start using Asynchronous Terminal Emulation for the first time. The program automatically creates the **ate.def** file and stores it in the current directory. You exit from Asynchronous Terminal Emulation and view the **ate.def** file. It looks like the representation of the original default file.

You want to make the following changes:

- Change the **rate** default from `1200` to `300` bps.
- Change **device** to `tty3` indicating the port the modem is plugged into.
- Change **directory** so that it defaults to a dialing directory file you have created called `my.dir`.
- Change **transfer** to default to `x` (XMODEM).

After you make these changes, **ate.def** looks like this:



```
RATE         300
DEVICE       tty3
TRANSFER     x
DIRECTORY    my.dir
```

Default File Commands

length

Definition: The **length** command specifies the number of bits that are in the data characters being transmitted or received. The length specified must match the length expected by the remote system.

Options: 7 or 8.

Default: 8

stop

Definition: The **stop** command specifies the number of stop bits appended to the character being sent or received. The number of stop bits specified must match the number of stop bits used by the remote system.

Options: 1 or 2.

Default: 1

parity

Definition: Parity is a simple means of providing error detection. For *even* parity, if the number of 1 bits in a character is odd, the parity bit is turned on to make an even number of 1 bits.

Options: 0 = none, 1 = odd, 2 = even.

Default: 0

rate

Definition: The **rate** command specifies the speed at which the bits of characters are transmitted or received. Your speed must match the speed of the modem and the speed expected by the remote system.

Options: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19200.

Default: 1200

device

Definition: The **device** command specifies the name of the asynchronous port that should be used to connect to a remote system.

Options: tty0 - tty16, or locally created port names.

Default: tty0

initial

Definition: The **initial** command specifies the dial commands that need to precede the telephone number when auto dialing with a modem. Consult your modem user's guide for the proper dial commands.

Options: ATDT, ATDP, etc.

Default: ATDT

final

Definition: The **final** command specifies the dial command which needs follow the telephone number when auto dialing with a modem. Consult your modem user's guide for the proper dial command.

Options: 0 = none, valid modem suffix.

Default: (blank)

wait

Definition: The **wait** command builds in a pause of *n* seconds between redialing attempts. If redial is set to 0, no redial attempt occurs.

Options: 0 = none, a positive integer.

Default: 0

attempts

Definition: The **attempts** command specifies the maximum number of times you want to try redialing if the initial connection cannot be made.

Options: 0 = none, a positive integer.

Default: 0

transfer

Definition: The **transfer** command specifies the asynchronous protocol to be used when transferring files from one system to another. Two choices follow:

- Pacing. Pacing is a file transfer protocol which controls the rate of transmission on either a character basis or on the basis of how many seconds are to be left between transmissions. It prevents certain systems from losing data when the transmission blocks are too large or are sent too quickly for the system to process them.

- XMODEM. XMODEM is an 8-bit file transfer protocol used to detect data transmission errors and retransmit data that had errors. It provides a higher degree of data transmission integrity than most other modes of asynchronous transmissions.

Options: p = pacing, x = xmodem.

Default: p

character

Definition: The **character** command specifies the type of pacing protocol to be used.

Options:

- 0: A zero setting means that data is transmitted without pacing.
- A single character: Character pacing allows you to select the character which is to be recognized as the signal to start transmission between lines.
- An integer: Integer pacing specifies how many seconds should be left between transmissions.

Default: 0

name

Definition: The **name** command specifies the file that incoming data goes to if the **write** command is on or if you enter the **Ctrl-b** key sequence during a connection.

Options: Any valid file name that is less than 41 characters long, or a file named *kapture.*

Default: kapture

linefeeds

Definition: When the **linefeeds** command is on, a linefeed character is added after every carriage return it detects in the incoming data stream.

Options: 1 = On, 0 = Off.

Default: 0

echo

Definition: When you type a key on the keyboard, that character appears on your screen. When operating with a modem that supports duplex capability, each character sent to the remote computer returns to the originating terminal and appears on the screen, thus displaying the character twice. The **echo** off command lets you suppress the display of typed characters, so that they appear only once.

Options: 1 = On, 0 = Off.

Default: 0

VT100

Definition: DEC VT100 emulation allows you to use DEC VT100 codes during a connection. When the **VT100** command is on, the local console emulates a DEC VT100 terminal with the remote system. When off, the local console functions like an IBM RT PC with the remote system.

Options: 1 = On, 0 = Off.

Default: 0

write

Definition: When the **write** command is on, Asynchronous Terminal Emulation routes the incoming data to a file, specified by the **name** command, as well as to the display.

Options: 1 = On, 0 = Off.

Default: 0

Xon/Xoff

Definition: When the **Xon/Xoff** command is on, the system controls data transmission at the port in the following manner:

- When system receives an Xoff, it suspends transmitting data until it receives an Xon.
- When system receives an Xon, it resumes sending data if previously suspended.
- The system sends an Xoff when its receive buffer reaches a near full condition.
- The system sends an Xon when the receive buffer is no longer full.

Options: 1 = On, 0 = Off.

Default: 0

directory

Definition: The **directory** command specifies the name of a dialing directory file that stores up to 20 telephone numbers. The file name may be any sequence of characters less than 41 characters long.

Options: Any character string less than or equal to 40 characters in length.

Default: /usr/lib/dir

Remapping Keys

Changing the control keys is an advanced feature that you should not attempt if you are inexperienced with computers. You should change control keys only with a specific application in mind. One possible need for this feature is a situation where the Asynchronous Terminal Emulation control keys conflict with the control keys used by the editor. In this instance, it may be necessary to remap the existing control keys.

capture-key

Definition: The **capture-key** is used to switch data capture on/off during an active connection.

Options: Any ASCII control character

Default: ASCII octal 002 (STX), which corresponds to the **Ctrl-b** key sequence on the system keyboard.

main-menu-key

Definition: The **main-menu** key is used to return to the Connected Main Menu to issue a command during an active connection.

Options: Any ASCII control character

Default: ASCII octal 026 (SYN), which corresponds to the **Ctrl-v** key sequence on the system keyboard.

previous-key

Definition: The **previous-key** is used to return to a previous screen at any time during the program.

Options: Any ASCII control character

Default: ASCII octal 022 (DC2), which corresponds to the **Ctrl-r** key sequence on the system keyboard. This value is mapped to the AIX interrupt signal.

Note: You may enter a new control character in one of three formats:

octal	000 through 037 (The leading zero is required)
decimal	0 through 31
hexadecimal	0x00 through 0x1F (The leading 0x is required. The x may be uppercase or lowercase).

Once Away Connections

Asynchronous Terminal Emulation can be used to connect to a remote computer through an intermediate computer that also has Asynchronous Terminal Emulation installed. This type of connection is referred to as *once away.*

Example:

```
Computer A ----------->  Computer B ----------->  Computer C
```

The middle computer, Computer B, must have the correct equipment to connect to both Computer A and Computer C. It can have two modems, one modem and a direct connect cable, or two direct connect cables. For more information, see "Interconnecting Terminals and Modems" on page 5-9.

If modems are used, the modems must be turned on. The modem at Computer A and one modem at Computer B must be in *call* mode. The second modem at Computer B and the modem at Computer C must be in *auto answer* mode. Also, the port being called at Computer B and at Computer C must be enabled with the **penable** command. For more information, see "How to Use Port Commands" on page 5-5.

To establish a **once away** connection, take the following steps:

1. At Computer A, login. Enter the Asynchronous Terminal Emulation program by keying in `ate` and connect to Computer B.

2. Now login to Computer B by following the prompts, and get connected to it. At this point, you can work on Computer B as though you were at its console using its keyboard.

3. Next, enter Computer B's Asynchronous Terminal Emulation program by keying in `ate`, and connect to Computer C.

4. Next, login to Computer C by following the prompts and get connected. At this point, you are connected to Computer C

through Computer B. You can work on Computer C as though you were at its console using its keyboard.

If Computer C has an *xmodem file transfer* program, you can receive a file at Computer A that this program on Computer C sends. If you want to send a file in the other direction, from Computer A to Computer C, you need to use the **xmodem** shell command with the *pass-through* option at Computer B. For more information, see "Xmodem File Transfer" on page 3-70. transfer later in this chapter for instructions on how to do this.

If Computer C has a *pacing file transfer* program, you can both send and receive files just as you would if Computer A and Computer C were connected without Computer B in between them.

While you are connected, you can access Computer B by using the **Ctrl-v** key sequence to display Computer A's Connected Main Menu. While in this menu, you can press **Ctrl-b** and Enter to start data capture into a file at Computer B, or you can press **Ctrl-v** and Enter to access Computer B's Connected Main Menu.

5. Finally, when you are ready to disconnect, press **Ctrl-v** to access Computer A's Connected Main Menu. Press **Ctrl-v** and Enter to get to Computer B's Connected Main Menu. Select **terminate** from the menu. This returns you to Computer B's Asynchronous Terminal Emulation program where you can either make another connection or select **quit**.

   After you have **quit** Computer B's Asynchronous Terminal Emulation program, you can disconnect from Computer B by using the **Ctrl-v** key sequence to access Computer A's Connected Main Menu, where you should select the **terminate** option.

Note: The **once away** feature can actually link more than three computers. However, the operation becomes time consuming because you must follow a lengthy disconnection procedure. If you do not disconnect carefully, you could leave processes running on the remote computers and leave ports open and vulnerable to unauthorized access.

Example:

To access the Connected Main Menu of the third machine (Computer C) so you can disconnect, take these steps while at Computer A:

What You Press	***What You See***
Ctrl-v	Computer A's Connected Main Menu
Ctrl-v, Enter	Computer B's Connected Main Menu
Ctrl-v	Computer A's Connected Main Menu again
	(Computer B's Connected Main Menu is awaiting a command)
Ctrl-v, Enter, Enter	Computer C's Connected Main Menu

Xmodem File Transfer

The *xmodem file transfer* program is a protocol that consists of a menu command and a shell command.

Xmodem Protocol

Note: Xmodem is a file transfer protocol in the public domain that was designed by Ward Christensen.

The two computers involved in a file transfer are known as the *sending side* and the *receiving side.* The sending side does not begin to transfer data until the receiving side signals that it is ready to receive data. Every 10 seconds until transfer begins, the receiving side sends the Negative Acknowledgment (NAK) character to the sending side. If the file transfer does not begin after 9 NAKs are sent, you must restart the process manually.

Once the sending side receives a NAK, it uses a Start Of Header (SOH) character and two block numbers to signal that a 128-byte block of data has started to transfer. (One block number is the true number and the other is the 1's complement of that number). The sending side then sends the block followed by an error-detection *checksum.* The checksum is calculated by adding the ASCII values of each character in the 128-character block, dividing the sum by 256, and retaining the remainder as the checksum.

After each block is transferred, the receiving computer calculates the checksum and compares the result with the checksum from the sending side. If the two values are the same, the receiving side sends an Acknowldege (ACK) character to tell the sender to send the next sequential block.

If the two values are not the same, the receiving side sends the transmitting side a NAK to request a re-transmission of the last block. The re-transmission is repeated until the block of data is correctly received or until there have been 9 attempts to transmit the block.

When all data has been successfully transmitted, the sending side gives the receiving side an End Of Transmission (EOT) character to indicate the end of file condition.

Example:

```
Receiving Computer                Sending Computer
Ready to Receive                  Ready to Send

----------------------/NAK/------------------------>
<----/SOH/Block #1/Block #1/Good Data/Checksum/-----
----------------------/ACK/------------------------>
<----/SOH/Block #2/Block #2/Good Data/Checksum/-----
----------------------/ACK/------------------------>
<----/SOH/Block #3/Block #3/Good Data/Checksum/-----
----------------------/NAK/------------------------>
<----/SOH/Block #3/Block #3/Good Data/Checksum/-----
----------------------/ACK/------------------------>
----------------------/EOT/------------------------>
----------------------/ACK/------------------------>

File Receipt Ends                 File Sending Ends
```

Xmodem Shell Command

Asynchronous Terminal Emulation provides the shell command **xmodem** (typed in lowercase) that you can use to complete an xmodem protocol file transfer from a remote system that has Asynchronous Terminal Emulation installed but is not currently running the program. The format of the command is as follows:

`xmodem` *-opt filename*

The *-opt* parameter indicates whether you want to *send* the file, *receive* the file, or use the *pass-through* option.

Note: You can interrupt the **xmodem** shell command by pressing **Ctrl-x**.

Sending and receiving with **xmodem** are complementary operations. One system must be set to send while the other is set to receive. If both are set to send or both are set to receive, the transfer does not work.

Sending a File

After you have established a connection to a remote computer, you can send a file from it to the local system by keying in the following to the shell of the remote terminal:

- `xmodem -s` *filename*

  The *filename* field should contain the name of the file you want to send.

- Press **Ctrl-v** to return to the Connected Main Menu of the local system.

- Select **receive** from the menu.

  After the file is transferred, you return to the connection screen.

Receiving a File

After you have established a connection to a remote computer, you can receive a file from the local system by keying in the following:

- `xmodem -r` *filename*

  The *filename* field should contain the name of the file you want to receive.

- Press **Ctrl-v** to return to the Connected Main Menu of the local system.

- Select **send** from the menu.

  After the file is transferred, you return to the connection screen.

Using the Once Away Feature

In the **once away** setup, you can use the **pass-through** option with the **xmodem** command at the middle computer. However, use this option ONLY when you are sending a file from the local computer to the remote computer. You do NOT need to use it if you are receiving a file from the remote computer.

Note: Because of the lengthy setup time for a **once away** connection, sending a file from the local computer is likely to fail if the remote computer is using an **xmodem** protocol that sends only one NAK, instead of several, to indicate it is ready to receive.

To set up a file transfer, take the following steps:

1. At the remote system, key in `xmodem -r` *filename.*

   Xmodem responds with this message:

   ```
   062-005  The system is ready to receive file "filename".
            Use Ctrl-x to stop xmodem.
   ```

2. Press **Ctrl-v** to return to the Connected Main Menu of the local system. Then press **Ctrl-v** and Enter to access the Connected Main Menu of the middle computer.

3. Select **perform** to issue the **xmodem -p** command. You actually key in the following line:

   ```
   p xmodem -p
   ```

   No messages appear at this time.

4. Press **Ctrl-v** to return to the Connected Main Menu of the local computer. Select **send**. When the file transfer finishes, a completion message appears:

   ```
   062-015  The file transfer is complete.
   ```

   Finally, you return to the connection screen.

If you prefer, you can send a file to the intermediate computer first and then re-send it to the remote computer in a two-stage process.

Pacing File Transfer

Pacing is a file transfer protocol used to allow computer-to-computer transfers that would normally overload the receiving system and result in lost data. It is useful for sending text files only. Files containing embedded control characters cannot be transferred using this protocol.

UNIX files traditionally contain only linefeeds (called *newlines*) to delimit the end of a line. DOS or CP/M files usually use only carriage returns. For this reason, the pacing **send** protocol converts linefeeds to carriage returns. It does not expect to find any carriage return/linefeed combinations in a file. The pacing **receive** protocol converts any linefeed/carriage return combinations or single carriage returns to linefeeds before saving the file.

Asynchronous Terminal Emulation supports two types of pacing control protocols, *character pacing* and *interval pacing.*

Character Pacing

The character pacing **send** protocol transmits data it reads from the file. When it finds a linefeed, it sends a carriage return and then waits 30 seconds for the agreed-upon pacing character before sending any further data. If no character arrives in that time, or if one comes but it is not the right one, the **send** operation ends.

The character pacing **receive** routine sends the agreed-upon pacing character when it first begins and then waits 30 seconds for the first line of data to arrive. If no data arrives in this time period, the **receive** operation ends and must be restarted.

After data transmission starts, the **receive** routine sends the agreed-upon pacing character to the sending side whenever it finds a carriage return in the data. The program ends when it receives nothing for 30 seconds.

Example:

```
Receiving Computer                Sending Computer
Ready to Receive                  Ready to Send


------------------------/P/------------------------->
<------------------/Line of Data/---------------------
------------------------/P/------------------------->
<---------------/Last Line of Data/-------------------
------------------------/P/------------------------->

File Receipt Ends                 File Transmit Ends
```

Interval Pacing

The pacing **send** routine begins to transfer data as soon as it is started. When it finds a linefeed, it transmits a carriage return and then waits the interval that you specified before it sends the next line.

The pacing **receive** routine begins looking for data as soon as it is started. It expects the sending system to send data within 30 seconds. It ends when it receives nothing for 30 seconds.

Example:

```
Receiving Computer                 Sending Computer
Ready to Receive                   Ready to Send

<--------------------/Line of Data/-------------------
                   (Wait interval)
<-----------------/Last Line of Data/-----------------
                   (Wait interval)

File Receipt Ends                  File Sending Ends
```

Trouble Shooting

Problems may occur when you are trying to establish a connection or while you are using a connection. During a connection, you may experience problems with either pacing or xmodem. Following are several problem areas and steps to take to correct the trouble.

Problems while Establishing a Connection

If you cannot dial, check the following:

- Make sure your modem is turned on.
- Make sure the modem switches are set correctly. Refer to the instruction book for your modem.
- Sometimes a previous connection leaves a modem in a state from which it cannot dial. Try turning the modem off and then back on again to reset all the default values in the modem.
- Make sure the modem cable is connected at both ends, into the modem and into the computer. Sometimes the cable can work loose.
- Make sure the computer end of the cable is plugged into the correct slot on the RS-232 card for the port you are trying to open. For more on RS-232 and RS-422 cards, see the "Interconnecting Terminals and Modems" on page 5-9.
- Check that the cable connecting the computer to the modem is the right type. For more information, see "Interconnecting Terminals and Modems" on page 5-9.
- A modem must receive characters at a certain speed in order to dial. For instance, one popular modem dials only when it receives characters at 75, 150, 300, or 1200 bps. Check the **rate** setting in the Alter Menu to make sure it is set at a valid speed.

- Check that the dialing prefix in the Alter Menu is correct for your modem. Hayes[1] compatible modems usually use a prefix of ATDT or ATDP, in uppercase. Make sure they are not in lowercase.

- Check that the dialing suffix in the Alter Menu is correct for your modem. You may not need one at all; if you use one, it could reset your modem to a state that does not allow your system to communicate.

- Make sure the phone line is plugged into the modem and that the phone line is active. Try dialing a number manually on the line. If you cannot dial out manually, the tip and ring phone wires may be reversed.

If you cannot open a port, check the following:

- Make sure the port you are trying to use is listed in the **/dev** directory. You can check this by keying in the following:

  ```
  li -l /dev/*tty*
  ```

  If the port you want to use is not there, you must create it. You can use the **devices** command to create it. See "A Note on the Minor Device Number" on page 5-7, "Example: Modem Connect for a Call-Out Port Setup" on page 4-48, "Example: Modem Connect for a Call-In Port Setup" on page 4-52, and *Installing and Customizing the AIX Operating System* for more information.

- If the port exists, it may not be defined correctly for a modem. Use the **devices** command to check and reset the parameters. Some of the settings you may need to change are the following:

[1] Trademark of Hayes Microcomputer Products, Inc.

Name	*Value*
dvam (Device Attachment Method)	1 (remote)
bpc (Bits per Character)	5, 6, 7, or 8
pt (Parity Type)	none, odd, or even
ixp (Include Xon/Xoff Protocol)	true
rts (Receive/Transmit Speed)	300 or 1200 for modems
sns (Switched/Nonswitched)	true
tt (Terminal Type)	dumb, VT100, etc.
nosb (Number of Stop Bits)	1, 1.5, or 2

If you change port settings, you must disable the port with the **pdisable** command before you use the **devices** command. After you leave **devices**, issue **penable** and then **pdisable** to put the new settings into effect on the port.

- If you are using a port to call out, you must disable it with **pdisable**. You can check its status by issuing the **penable** command. Any port listed is enabled. To disable one, use the **pdisable** command followed by the port name. For more information on call-out ports, see "Getting Started with Ports" on page 5-4 and "How to Use Port Commands" on page 5-5. guide).

  Example:

  ```
  pdisable tty0
  ```

If the other side does not answer, check the following:

- Make sure that the remote modem is turned on, that its switches are set correctly, that the phone line is connected, and that the modem cable is plugged in correctly at both the modem and computer ends.

- Check that the remote modem is in auto answer mode. You can set this by a switch on the modem or by typing a command to a smart modem from the remote keyboard.
- The remote port must be enabled or have another communications package running on it if it does not have the same operating system that the local system has.
- Verify that the number you are dialing is correct.

Problems during a Connection

After you have successfully established a connection, you may have problems while you are using it. Following are several problems and possible solutions.

If garbage or nonsense appears on your screen, check the following:

- Try changing the communication parameters in the Alter Menu. The **length**, **stop**, **parity**, and **rate** parameters must all match what is being used on the other end.
- It is possible in AIX for two system users to open the same port. When this happens, both users are trying to read the port at the same time. One user gets some of the characters, and the other gets the rest. One user must stop trying to use the port. To do this, press **Ctrl-r** to disconnect, or press **Ctrl-v** to get the Connected Main Menu and then select **terminate**.
- When you are using a port to call out, it must be disabled. If someone else enables it during your connection, the send and receive lights on your modem appear to be in constant use and the program does not appear to work. In this case, press **Ctrl-r** to disconnect.

If you are losing characters, check the following:

- At high transfer speeds (9600 and 19200 bps), you may find that you are losing characters or getting buffer overflow messages. This may happen when the computer is very busy and several programs are running at once. You can continue to use these speeds if the remote computer supports Xon/Xoff protocol. Use the Modify Menu to set Xon/Xoff to on, and continue with your connection session.

If control keys do not work, check the following:

- Since control keys do not display on the screen or print on paper, this problem may be difficult to trace. The main symptom is that when you press one of the control key sequences (**Ctrl-r**, **Ctrl-b**, or **Ctrl-v**), nothing happens. Look in the **ate.def** file to see if the control keys have been reset. If so, you can delete the file, and the next time you run Asynchronous Terminal Emulation, the program creates a new file, using the program defaults.

- Your terminal may use a different control key combination to produce the code recognized by the program. The program defaults are the following:

Ctrl-b	002 octal, 0x02 hex, STX ASCII
Ctrl-r	002 octal, 0x12 hex, DC2 ASCII
Ctrl-v	026 octal, 0x16 hex, SYN ASCII

  Check your terminal book to see which control key combinations produce the codes you need.

If your modem does not hang up, check the following:

- You may need to reset the modem if you were disconnected abruptly. Try turning the modem off and then back on.

- There may be too many processes using the port. You can check this by issuing the shell command **ps -el** and looking for processes that should not be running. If you find any, you can eliminate them by using the shell command **kill -9** *PID*, where PID is the process identification number obtained from the **ps** command.

  When you are using Asynchronous Terminal Emulation, one or two processes may be running:

- **ate** runs all the time you are using the program
- **atefork** runs when you have an active connection.

On multiple user systems, more than one person may be running the program at a given time. Make sure you eliminate only processes that should not be running.

Problems while Using Pacing

If a file does not transfer, check the following:

- If you receive message 062-002 while using either interval or character pacing to receive a file at the local site, possibly the remote program did not send an EOT character. Check the file you received to make sure all the data arrived. If so, no further action is necessary. If not, you may want to retry.
- Files sent or received using the pacing protocol should not contain embedded control characters. If they do, the results are unpredictable. Files should contain only text.

Problems while Using Xmodem

If linefeed/carriage return characters are not correct, check the following:

- UNIX files traditionally contain only linefeeds (called newlines) to delimit the end of lines. DOS or CP/M files usually use a carriage return/linefeed combination. After you transmit the file, if the linefeed/carriage returns do not look right, you can use one of the two following AIX procedures on the file either before you send it or after you receive it:
  - BEFORE sending from AIX to DOS or CP/M, key in the following line to change the newlines to a combination of carriage return/linefeeds:

```
sed 's/$/' 'echo '\r\c''/ file1 > file2
```

The *file1* field contains the original name of the file, and the *file2* field contains a new name for the file. Send the modified file, *file2.*

Note: See the **sh** command in *AIX Operating System Commands Reference* for other quoting mechanisms in command substitution.

- AFTER receiving from DOS or CP/M to AIX, key in the following two lines to remove the carriage returns from the file:

  ```
  tr -d '\015' < file1 > file2

  mv file2 file1
  ```

If you get garbage or nonsense on the screen, check the following:

- When you interrupt an **xmodem receive** with **Ctrl-r**, the **xmodem** shell command continues to send. The characters it sends appear on the connection screen. This is normal. Press **Ctrl-x** to end the **xmodem** shell command and continue with your connection screen.

Chapter 4. Communicating: UUCP

CONTENTS

About This Chapter

This chapter explains how to use the UNIX-to-UNIX Copy facility to:

- Send and receive files locally or between a remote system and the local system
- Run commands remotely
- Manage certain administrative tasks
- Customize the UUCP facility for your own needs.

Note: This chapter is for readers who are thoroughly familiar with UNIX or AIX and with IBM RT PC. Also, you should have Extended Services installed on your system. *Installing and Customizing the AIX Operating System* tells you how.

Overview of the UUCP Facility

The term *uucp* has three different meanings in this guide:

- A way for two computer systems to communicate, using several directories, files, programs, and commands
- A program that initiates file transfers
- A command to enter with various options, depending on the task you want to accomplish.

To help you keep these meanings clear, references to the whole facility appear in capital letters (UUCP). Lowercase letters in boldface with the word **program** following indicate the program that starts the transfer (**uucp program**). References to the command that you key in to access the program appear in lowercase boldface letters with the word command following (**uucp command**).

The UNIX-to-UNIX Copy facility (UUCP) is a group of primary and administrative programs and files designed to support communications between your local system and another UNIX system once the configuring and customizing are finished.

The UUCP facility lets you communicate with a remote computer system either directly over a dedicated (hardwired) line or indirectly over a telephone line.

An advantage of UUCP over other ways to send and receive material between two systems is that it is a batch operation that runs as an AIX background process. Once you start a UUCP task running, you can then use your work station for other jobs without having to wait for the task to finish. For example, you can send a large file to a remote system for printing on a particular printer and at the same time you can edit a document at your work station.

There are two different ways to set up communications between your own computer and other systems. One gives you a call-in line that people on other systems can use to transfer files and commands. The other method gives you a call-out line that lets you call another system, login, and transfer files and commands. You can have both methods available on your system, but you need to make sure that a line is dedicated to UUCP. This chapter provides material on how to use both methods.

Directories

While a UUCP task is running, it uses files in the three following directories:

- **/usr/lib/uucp**

  Used for system files and some executable programs

- **/usr/spool/uucp**

  Used for running the **uucp command** and storing temporary spool files.

- **/usr/spool/uucp/.XQTDIR**

  Used for execution files that contain lists of commands remote systems may run.

Spool Directory and Types of Files

There is also a *spool directory* (**/usr/spool/uucppublic**) for temporary storage of files in transit. It stores files created for processing by UUCP. It uses three types of files to perform UUCP tasks:

- *Data files* contain the data to send to remote systems.

- *Work files*, also called *command files*, contain directions for the communications program **uucico**.

- *Execute files* contain instructions for running commands that require the resources of a remote system.

Programs in the UUCP Facility

There are 10 primary and administrative programs associated with the UUCP facility and one command that you may need:

The command **uuname** gives you the names of the remote systems that you can communicate with.

Primary Programs

- **uucp program**

  Creates work files and gathers data files into the spool directory so that you can send them to other systems. The program **uucico** actually performs the transfer work.

- **uux**

  Creates work files and execute files and gathers data files so you can run AIX commands on remote systems.

- **uucico**

  Follows the directions in the work files for sending and receiving files and performs the transfer task that the **uucp command** requests. (Not usually run by the average user).

- **uuxqt**

  Follows the instructions for running commands that are in the execute files. (Not usually run by the average user).

- **uuto**

  Uses UUCP to send files to a user on another system. (Uses a particular access scheme for added security).

- **uupick**

  Gives you several options for processing a file received from a remote system using the **uuto** program. (Uses a particular access scheme for added security).

Administrative Programs

- **uulog**

  Collects information from temporary log files, summarizes information about **uucp program** and **uux** transactions, and displays it, such as work done for a particular person or system.

- **uuclean**

  Removes selected files of a specified age from the UUCP spool directory or from another directory you can name.

- **uustat**

  Displays and manipulates the status of previously specified uses of the **uucp command** and displays the status of links to remote systems.

- **uusub**

  Displays and manipulates the status of UUCP subnetworks and monitors connections and traffic among them.

Using the UUCP Facility

This section describes how to accomplish four tasks with the UUCP facility:

- Learn the UUCP names of systems you can communicate with.
- Transfer files (send and receive) within the local system or between a remote system and the local system.
- Send mail to someone else or to yourself to give notice that a transfer took place.
- Carry out a command on a remote system.

Finding System Names with uuname

One of your first tasks in using the UUCP facility is to find out the names of systems you can communicate with from the local system you are using. To do this, right after the **$** (dollar sign used as the system prompt), key in:

```
uuname
```

To find out the name of the local system you are logged into, use the **-l** flag with the command as follows:

```
uuname -l
```

Transferring Files

With the UUCP facility, you can send and receive files within the local system you are logged into, between a remote system and the local one, or between two remote systems.

Local Transfers

You can send one or more files to the directory of another person on the local system. You can also obtain one or more files from someone else and place them in one of your own directories. Locally, UUCP works like **cp**. See *AIX Operating System Commands Reference* for more on **cp**.

Sending or receiving a file within the local system requires that you know the name that the system uses for both the *source-file*, or original location, and the *destination-name*, the file or directory where the new copy goes. In the source-file, you can use special shell characters (?, *, [, and]). The remote shell expands them. To prevent the local shell from interpreting them, surround the line with double quotes.

The name of a source-file or destination-name is a *path name*.

Path Names: A path name may be one of the following:

- A full path name. This lists all the directories by name that lie along the path from the root to the specific file and concludes with the file name. By convention, the names are separated by slashes.

- A directory name. The file name comes from the source.

- A partial path name. This lists only the directories from the current working directory on down to the file name. If you are already in the `/usr/bin/tony/mice` directory, then keying in `diets` (with no slash before `diets`) sufficiently identifies the file `/usr/bin/tony/mice/diets`.

- A special ~*user* prefix, where *user* is a login name on the system. The ~ *user* refers to the login directory of the user directory.

- A file name beginning with ~/ refers to files in the public directory (usually **/usr/spool/uucppublic**) on the remote system. This is the public UUCP directory for sending and receiving material as defined in the UUCP source.

- A system name followed by ! (exclamation point) and a file or directory name, such as `sysabc!` *pathname*.
- Several system names, each followed by ! (exclamation point), and a file or directory path name, such as `sysx!sysy!sysz!` *pathname*.

Note: The path name should not have the form ../.

If the destination-name is a directory name, the **uucp command** forms a name for the file from the last part of the source-file. You should not use special shell characters (?, *, [, and]) in the destination-name.

Permissions: When transmitting files, the **uucp command** searches for the directories, preserves execute permissions, and grants read and write permissions of 0666. The command also assigns and displays a job number for each remote use of the command. For more information on permissions, see *Using and Managing the AIX Operating System.*

Remote Transfers

To make transfers outside the local system, you add the name of the remote system to the destination-name for sending or to the source-file for receiving.

Note: Normally, remote transfers are restricted to the public spool directories.

Sending Files: To send to a remote system, key in the following:

`uucp` *{options} source-file(s) system-name!destination-name*

Substitute actual names for the system and file fields. The following example sends a copy of your local file, `/meteors/small.c`, to the `/solar/stats.c` file on the remote system, `galaxy`.

```
uucp /meteors/small.c galaxy!/solar/stats.c
```

The UUCP facility also lets you forward files through intermediate systems that the local system cannot access directly. For example, if you cannot connect with the `galaxy` system but you can contact `milkyway` that has access to `galaxy`, then you can transfer the file by keying in the following:

```
uucp /meteors/small.c milkyway!galaxy!~/solar/stats.c
```

The **uucp command** attempts to use the specified route and to send the source-file(s) to a destination in the remote system's public directory, **/usr/spool/uucppublic**.

Receiving Files: To receive a file or files from a remote system that the local system can connect with, key in the following, making the appropriate substitutions for file and system names:

`uucp` {*options*} *system-name!source-file(s) destination-name*

You can list several system names to give a path for the source files to travel along. The following example requests a copy of a file, `/cells/typel`, that resides on remote system `biochem`. The file travels through system `cellsys` before coming to your file, `/drjay/research`, on the local system:

```
uucp cellsys!biochem!/cells/typel !/drjay/research
```

Notice that there is no system name in front of the ! (exclamation point) preceding your file name, `/drjay/research`. The **uucp command** uses the local system as the default.

In the example, the local system connects with `cellsys`, which connects with `biochem`. If there are no restrictions or other problems, you should receive the files.

The uucp Command

To use the **uucp** command, key in `uucp` and substitute the correct path names for the source and destination files:

`uucp` {*options*} *source-file(s) destination-name*

UUCP checks each source-file and destination-name, the system field, and the options. It classifies the work into these types:

1. Copy the *source-file(s)* to the *destination-name* on the local system.

   When the transfer is local, the **-m** and **-n** options do not work. See "UUCP User Notification" on page 4-14.

2. Receive a file or files from other system(s).

   For each request to receive files, UUCP adds a one-line entry to a *work file* or creates a new one if the current one is full. The **uucp.h** file sets the upper limit of a work file, with a default of 20 lines. A work file has five fields, with a blank as the field separator:

   - R (for receive)
   - The full path name of the source file or a *~something/pathname* (The remote system expands the *something* field)
   - The full path name of the destination-name (If the *~something* notation is used, it is expanded immediately)
   - User's login name
   - A - (hyphen) followed by an option list. You may use **-m** to send yourself a mail message when the work finishes or **-d** to create any directories necessary to copy the file. See *AIX Operating System Commands Reference* for other options.

3. Send a file or files to a remote system.

A copy of each source-file goes into a *data file* in the spool directory, unless you specified **uucp -c** to prevent generation of the data file. In that case, the transfer goes directly from the source to the destination with no stopover in the spool directory.

There are eight fields in a data file:

- S (for send)
- The full path name of the source-file
- The full path name of the destination-name or *~something/file_name*
- User's login name
- A - (hyphen) followed by an option list (You may specify **-d** or **-m**, as in a work file, or **-n** to notify the recipient that you sent the file)
- The name of the data file in the spool directory (A dummy name, **D.0**, is used when you specify the **-c** option)
- The file mode bits of the source-file in octal print format, such as 0666
- The user on the remote system to notify when the transfer finishes, if you used the **-n** option.

4. Send one or more files from one or more remote systems to another remote system.

   UUCP generates a **uucp command** and sends it to the remote machine whose **uucico** program runs the command.

5. Receive one or more files from a system when the source-file contains special shell characters (?, *, [, and]).

   UUCP generates a **uucp command** and sends it to the remote machine whose **uucico** program runs the command.

6. Request that the **uucp command** be executed by a remote system.

   This happens when you specify **uucp -e**. The **uux** program creates and sends the request if the remote machine's **uuxqt** program permits the **uucp** command. See "The uux Command" on page 4-18.

After the work has been set up in the spool directory, the **uucico** program starts trying to contact the other machine to perform the work. See "The uucico Command and Problem Determination" on page 4-31.

UUCP User Notification

UUCP provides two flags or options for automatic mail. One lets you notify another person that you have sent a file, and the other gives you a message about whether the file transfer completed normally.

The `-n` flag notifies a system user to expect a file. For example, to let system user `labdir` on system `lab3` know that you sent file `/cells/changes` to file `/research/new`, key in the following:

```
uucp -nlabdir /cells/changes lab3!/research/new
```

To send a message to yourself on whether a transfer finished normally, use the `-m` flag, as in the following example:

```
uucp -m /cells/changes lab3!/research/new
```

If you would like the message to go to a certain file instead of to your mailbox, you can add the name right after the flag, like this example:

```
-mreport
```

Finally, you can combine the two flags so that one **uucp command** transfers files, notifies another system user, and sends a message to you:

```
uucp -mreport -nlabdir /cells/changes lab3!/research/new
```

Note: The **uucp command** has seven other flags or options that you can use. See the *AIX Operating System Commands Reference* for more information.

Using uuto and uupick

These two commands use the UUCP facility to transfer files with special handling. They use PUBDIR, the public directory defined in the UUCP source. The **uuto** command sends files to a destination in this format:

`PUBDIR/receive/`*userID/system/destination-name*

The **uupick** command searches PUBDIR (**/usr/spool/uucppublic**) specifically for files sent to your user ID. The *spool* field indicates that a file being transferred does not move directly from one system to another but is first spooled into PUBDIR where it awaits transfer.

Using uuto: The **uuto** command uses the UUCP facility to send files. A remote system user wanting a certain file must ask you to send it to that specific user ID. The destination-name has this format:

remote-system-name!remote-user-logname

For example, if `smythe` on system `bravo` cannot get access to file `/usr/bin/data/private` because it is restricted to local use, you can send it with the following line:

```
uuto /usr/bin/data/private bravo!smythe
```

The **uuto** command also sends `smythe` a mail message when the file you sent arrives at system `bravo`. No mail message is sent in a local transfer.

Using uupick: The **uupick** command lets you read and dispose of files that someone at a remote system sent specifically to you.

When you get a mail message that material from another system has arrived, you should key in `uupick` right away.

The **uupick** command gives a message in this format:

```
from system system-name: {file file-name} {dir
directory-name}
?
```

The { } (braces) indicate that the fields are optional.

If `smythe` sent `/info/local` to you with **uuto**, you should receive a message like the following one:

```
from system bravo: file /smythe/info/local
?
```

The **?** (question mark) is a prompt requiring you to take some action. You can respond to it by choosing one of the items from the box on the next page. This is similar to disposing of mail messages, as explained in Chapter 2 of this guide.

To Dispose of Files

Action	*Result*
Press <new line>	Go on to the next file.
Press d	Delete this file.
Press m{*dir*}	Move the file to directory *dir*. Leave the *dir* field blank to move the file to the current directory.
Press a{*dir*}	Move all files to directory *dir*. Leave the *dir* field blank to move the files to the current directory.
Press p	Display the contents of the file on the screen.
Press q	Stop using **uupick** without taking any action about the files.
Press Ctrl-d	Same as **q**.
Key in !*command*	Escape to the shell to run *command* and then return to **uupick**.
Press *	Display this list of options.

Note: Both **uuto** and **uupick** can be used with flags or options. See *AIX Operating System Commands Reference* for more information.

The uux Command

The **uux** command lets you carry out certain commands on a remote system. For security reasons, many installations permit **uux** to run only the **mail** command. Each system lists the commands accessible by a remote system in a file called **/usr/lib/uucp/QTCMDS**. A typical listing might include the following commands:

Example /usr/lib/uucp/QTCMDS File

```
uucp

rmail

print

who
```

To send the file /book/chapter01 to system bravo for printing, key in the following line:

```
uux bravo!print /book/chapter01
```

A mail message from the remote system informs you when it does not permit a command you requested.

If the command generates output that you want to store on a different system than the one that runs the command, you can specify a destination-name for the results to go into. It is important to handle the destination-name for an output file in a particular way, since the **uux** command tries to move all the files that you list to the system where the commands are to run. To prevent this action, use parentheses around the name of the destination-name and also escape the parentheses by surrounding them with double quotes or backslashes.

The following example gathers two files, /pat/sun and /jon/sun from two different systems, xox and yoy. It runs the **diff** command on the local system to generate a list of differences in the two files, and stores the list in an output file called /sun.diff on the local system:

```
uux "!diff xox!/pat/sun yoy!/jon/sun > !/sun.diff"
```

To get the output file to system zoz instead of the local one, key in this line:

```
uux wow!diff xox!/pat/sun yoy!/jon/sun \(zoz!/sun.diff\)
```

You need the parentheses to prevent **uux** from moving the file from system `zoz`. You need the backslashes to keep the local shell from interpreting the parentheses.

Note: The sequence `\( ... \)` must surround output files as well as elements that are not files but that contain an exclamation point.

Theoretically, you could specify any number of commands and any number of system names, but in practice, only the first command in your list is preceded by a system name. To run commands on several different systems, key them in one at the time on separate lines. Only the first command of a shell pipeline may have a system name, and all other commands are run on the system of the first command.

Names for source-file(s) and the destination-name may be a full path name, a partial path name, or a path name preceded by *~user*, where *user* is a login name on the specified system that expands to that system user's login directory.

You may use shell special characters (?, *, [, and]) in a path name and let the remote system expand them, but escape them with double quotes or backslashes to keep your local shell from interpreting them. Use them only in source files, not destinations.

You can also use other shell characters (<, >, ;, and |) and either escape them individually or quote the entire command string. It is important not to try using shell redirection characters (< < and > >) because they do not work in the UUCP facility. See *AIX Operating System Commands Reference* for more on output redirection.

The **uux** command generates an *execute file* that contains the names of the files required for running the command, including standard output, the system user's login name, the destination of the standard output, and the command to be run. This file either goes to the spool directory for local execution or is sent to the remote system with a send command.

For required files that are not on the machine that is to run the command, **uux** generates *receive command files*, which are placed on that machine. Then **uucico** runs those files.

The *execute file* contains a script that **uuxqt** processes. This file contains several lines, each with an identification character and one or more additional entries in the line, as follows:

user line	*U user system* *User* and *system* are the requester's login name and system.
required file line	*F file_name real_name* *File_name* is a unique name used for file transmission and *real_name* is the last part of the actual file name and contains no path information. Zero or more of these names can be present. The **uuxqt** program checks for the existence of all these files before running the command.
standard input line	*I file_name* The standard input is either specified by a < (less than symbol) in the command string or inherited from the standard input of the **uux** command if the - (dash) option is used. If standard input is not specified, **/dev/null** is used. Note that if there is a standard input specified, it also appears in an F (required file) line.
standard output line	*O file_name system_name* The standard output is specified by a > (greater than symbol) within the command string. If a standard output is not specified, **/dev/null** is used. Note that use of > > is not implemented.
command line	*C command {parameter(s)}* The parameters, zero or more in number, are those specified in the command string. The standard input and standard

	output do not appear on this line. All *required files* go to the execution directory (usually **/usr/lib/uucp/.XQTDIR**) and the shell specified in the **uucp.h** header file runs the NP command. In addition, a shell PATH statement appears in front of the command line as specified in the **uuxqt** program. Note that UUCP checks to see that the command is allowed, as specified in the **uuxqt** program. After execution, the standard output is copied or sent to the proper place.
mail suppression line	*Z* or *N* Z: After the command runs, do not return a message to the originator unless the command returned a non-zero status, meaning that it failed. N: After the command runs, never return a message to the originator.

The work of **uux** is carried out by the **uuxqt** program. You do not have direct access to **uuxqt** since **uux** and two other programs call it when needed. See *AIX Operating System Commands Reference* for details about **uuxqt**.

The **uucico** command also assists the **uux** command by scanning the spool directory for work, placing a call to a remote system, negotiating a line protocol to be used, running all requests from both systems, and logging work requests and work completions. Normally, **uucico** is started by the **cron** system daemon due to a **crontab** entry. The **uucico** program is also started by the **uucp**, **uux**, or **uuxqt** programs. The main reason you might start it directly is for testing or debugging. See "Problem Determination with the uustat and uucico Commands" on page 4-28.

For more on remotely executable commands, see "The QTFILE File" on page 4-46.

Managing a UUCP Network

To manage a UUCP network, you should have extensive knowledge of AIX or UNIX, UUCP, and IBM RT PC. It would also be helpful to have experience configuring and customizing a system.

Security and Protection

The UUCP facility, left unrestricted, lets anyone run a command and copy in or out a file that is readable or writable by a UUCP login user. The only requirement is that the system include *uucp* as a login name in the **/etc/passwd** file. To set up the UUCP facility, each system must install *uucp* as a login name. Each individual system is responsible for its own protection and security.

There are eight precautions you can take:

- File mode protection: *Using and Managing the AIX Operating System* has information on using the **chmod**, **chown**, and **chgrp** commands to set permission bits on your own files and directories. You should never change permission bits on UUCP files and directories.

- Login shell: The UUCP facility does not get a standard shell. Instead, the **uucico** program starts running and does all the work requests.

- The UUCP owner: The owner of the UUCP programs should be an administrative login. It should not be one of the logins used for remote system access to UUCP.

- A path check: You can set up the **USERFILE** to perform a path check on file names requested for sending or receiving and to require a call-back for specified login IDs. See also "Permitted File Access (USERFILE)" on page 4-41.

- Conversation sequence count: You can keep track of who has called in and can gather other information about communications between your own system and others.

- Restricting **uuxqt**: This program comes with a list of commands it can run, but you can modify the file**/usr/lib/uucp/QTCMDS** to expand or restrict the list. Some systems permit only the **mail** command to run on the UUCP facility.

- **The L.sys** file: The UUCP administrative login should own this file and should give it mode 0600 to protect the phone numbers and login information for remote systems. For more on this file, see "The L.sys File (Special Notes)" on page 4-46.

- Program ownership: The programs **uucp**, **uucico**, **uux**, **uulog**, and **uuclean** should have the UUCP as owner, should have execute permissions, and should have the *setuid* bit set. See "System Names (The L.sys File)" on page 4-43 and "Customizing the UUCP Facility" on page 4-36.

Running Utilities (uulog and uuclean)

Two programs, **uulog** and **uuclean**, are useful in managing the routine activities of the UUCP facility.

Getting Status Information with the uulog Command

UUCP normally makes an entry directly into **/usr/spool/uucp/LOGFILE** each time the facility is used. It lists the userID, the system name, the date and time, a sequence number, and the result of each UUCP transaction. If UUCP cannot access the LOGFILE, as may happen when more than one UUCP process is running, it creates an individual file with prefix LOG. Periodically, **uulog** may be run to append these files to the LOGFILE.

The **uulog** program can also display a summary of information about UUCP and **uux** transactions from **/usr/spool/uucp/LOGFILE**. You can request output for one user with `uulog -u`*user* or output for one system with `uulog -s`*system,* substituting actual user or system names.

An example of a typical log file follows:

Example /usr/spool/uucp/LOGFILE File

```
root bravo (04/12-9:07-69) DIAL FAILED
       (Busy or no answer)
chris xox (04/12-9:10-72) FAILED (LOGIN)
root bravo (04/12-9:13-115) XQT QUE'D (print)
chris xox (04/12-9:20-85) SUCCEEDED (Call to xox)
```

Removing Old Files with the uuclean Command

Normally started by the UUCP daily daemon, **uudemon.day**, the **uuclean** command by default deletes files in the spool directory that are more than 72 hours old. Files left in the spool directory for three days are usually files for work that can not be completed. The **uuclean** program notifies the user requesting the work that the files have been removed.

The **uuclean** command has four options that are explained in *AIX Operating System Commands Reference.* You have direct access to the command, but it is common for the **cron** daemon to call it so that you do not have to remember to remove old files periodically.

Note: **Crontab** must be set up to call **uuclean**.

Using uudemon Files for Automatic File Deletion

The three sample files that follow (**uudemon.hr, uudemon.day,** and **uudemon.wk**) contain typical routines for hourly, daily, and weekly maintenance of the UUCP facility. Place entries in the **/usr/lib/crontab** file so the **cron** daemon can run them at appropriate times. It is customary to run **cron** from the system initialization process through the file **/etc/rc**.

Example uudemon.hr File

```
# 'perform every hour on the 56-minute mark'
PATH=:/bin:/usr/bin
cd /usr/lib/uucp
uucico -r1
uulog
```

Example uudemon.day File

```
# 'perform once per day at 0400 hours'
PATH=:/bin:/usr/bin
cd /usr/lib/uucp
uuclean -p -m -n168 >/dev/null 2>/dev/null
uuclean -d.XQTDIR -p -n72 >/dev/null 2>/dev/null
uustat -c168 >/dev/null 2>/dev/null
uusub -u24
cd /usr/spool/uucp
mv LOGFILE temp
uniq -c temp >> Log-WEEK
rm temp
cd /usr/spool/uucppublic
find . -type f -mtime +30 -exec rm -f {} \ ;
```

The **uuclean** lines clean up the execution directory. The **uustat** line cleans up old status entries in **/usr/lib/uucp/L_stat** and **/usr/lib/uucp/R_stat**. The **uusub** line gathers statistics. The **mv** line places LOGFILE in a temporary location. The **uniq** line adds any unique entries to the weekly log. The **rm** line deletes the daily log. The **find** line removes any file in **/usr/spool/uucppublic** that has not been modified in 30 days.

Example uudemon.wk File

```
# 'perform once a week, Sunday, at 0530 hours'
PATH=:/bin:/usr/bin
cd /usr/spool/uucp
rm -f o.Log-WEEK* o.SYSLOG*
mv Log-WEEK o.Log-WEEK
(date; echo ============) >> o.Log-WEEK
mv SYSLOG o.SYSLOG
pack o.Log-WEEK o.SYSLOG
```

The **rm** line removes old packed logs. The first **mv** line moves last week's log into old logs. The next two lines add a date to the old logs. The second **mv** line moves last week's system log into old logs. The **pack** line packs the old logs.

Establishing a Subnetwork with the uusub Command

This command defines a UUCP subnetwork and monitors the connections and traffic among the systems in the subnetwork. By using the various flags, you can add a system to the subnetwork, delete a system, get connection statistics from the **/usr/lib/uucp/L_sub** file, get traffic statistics from the **/usr/lib/uucp/R_sub** file, and perform other tasks. The command also uses the system log file, **/usr/spool/uucp/SYSLOG**.

Connection statistics include the number of times the local system tried to call a system since the last flush, the number of successful connections, the latest successful connect time, the number of connections that were unsuccessful because no device was available, because of login failure, because of no response, and because of other reasons.

Traffic statistics include the number of files sent and received and the number of bytes sent and received over the period of time indicated in the latest **uusub** command having the -**u***hr* option.

Typically, **cron** starts this command once a day, using flags similar to those in the following example:

```
uusub -c all -u 24
```

This **uusub** command gathers the traffic statistics for all the systems in the subnetwork.

For complete information on flags, see *AIX Operating System Commands Reference.*

Problem Determination with the uustat and uucico Commands

You can use these two commands to gather information about problems that may occur in using the UUCP facility. These commands can help you debug the files, directories, and programs that UUCP uses. For hardware problems, see the section on tracing port failures later in the last chapter of this guide.

The uustat Command

With this command, you can display the status of previous **uucp commands** or connections with other systems. You can also cancel a **uucp command**.

By using one or more of the flags available with this command, you can get information on the status of specific UUCP jobs, systems, and users, over various time periods.

Flag	***Result***
-**m***mch*	Report accessibility status of machine *mch*. If you specify *all*, you get the status of all machines known to the local UUCP facility.
-**k***jobno*	End the UUCP job number *jobno*. You must own that request or else be the superuser to use this option.
-**c***hour*	Remove the status entries older than *hour* hours. This administrative option can be initiated only by the user *uucp* or by the superuser.
-**u***user*	Report the status of all UUCP requests issued by *user*.
-**s***sys*	Report the status of all UUCP requests for remote system *sys*.
-**o***hour*	Report the status of all UUCP requests older than *hour* hours.
-**y***hour*	Report the status of all UUCP requests younger than *hour* hours.
-**j***all*	Report the status of all the UUCP requests.
-**v**	Report the UUCP status verbosely. See the following list for return codes and explanations.

Note: Options -**j**, -**m**, -**k**, and -**c** are mutually exclusive and cannot be combined. Use one of them either alone or with one or more of the other flags. If you do not specify an option, **uustat** gives you the status of all the UUCP requests issued by the current user.

The -v flag is particularly useful. When you key in `uustat -v`, you get a descriptive report on the **uucp command** status. Words accompany and explain the base 8 numbers. If you do not specify this option, you get only the numbers. The list of numbers and words describing the status follows:

Octal	**Status**
000001	The copy failed but the reason is not known.
000002	Permission to access local file is denied.
000004	Permission to access remote file is denied.
000010	Bad **uucp command** is generated.
000020	Remote system can not create temporary file.
000040	Can not copy to remote directory.
000100	Can not copy to local directory.
000200	Local system can not create temporary file.
000400	Can not run **uucp command**.
001000	Copy succeeded.
002000	Copy finished and job is deleted.
004000	Job is queued.
010000	Job is halted (incomplete).
020000	Job is halted (complete).

The uucico Command and Problem Determination

This command is called by the **uucp command** to perform the work of transferring files. It does the following five tasks:

1. Scans the spool directory for a job. In the spool directory, the names of work-related files have the following format:

   type . system_name grade number

Field	**Description**
type	An uppercase letter • C=copy command file (work file) • D=data file • X=execute file.
System name	The remote system
Grade	A character indicating priority
Number	A four-digit, zero-padded sequence number

   For example, the file `C.lab9n0031` is a work file for a transfer between the local machine and the `lab9` machine.

   The **uucico** command scans the spool directory looking for work files (those with a prefix of C) and makes a list of all systems to be called. The program calls each system and processes the work files.

2. Places a call to a remote system.

   The **uucico** program uses information in several files in the UUCP directory, usually **/usr/lib/uucp**. At the start of the calling process, the program sets a lock to prevent multiple conversations between the same two systems. See "Cleaning up Lock Files" on page 4-35.

The program uses the information in the **L.sys** file to make the call and also checks the **L-devices** file to find an available device for the call. It creates a lock file if a device successfully opens.

The conversation between the local and the remote **uucico** programs starts with a handshake initiated by the called or secondary system. It lets the calling or primary system know that it is ready to receive the system identification and conversation sequence number. The primary system sends these, the secondary system verifies the information, and protocol selection begins. If the secondary system responds with a *call-back required* message, the current conversation ends and the primary system calls later.

3. Negotiates a line protocol to be used.

   The secondary system sends a message containing a string of characters, each representing a line protocol. The message has this format:

   P*proto-list*

   The calling system checks the list for a letter corresponding to an available line protocol and returns a message telling the secondary system which one to use. The message has the following format:

   U*code*

   If the two systems have no common protocol, the *code* field contains **N**.

4. Carries out work requests from both systems.

   The primary system searches for work requests from the secondary system and sends appropriate messages during the processing: **S** for Send a file, **R** for Receive a file, **C** for Copy complete, **X** for Execute a UUCP command, and **H** for Hangup. The primary system sends R, S, or X messages until all work for the remote system is finished and then sends H to end the conversation.

The secondary system responds with SY, SN, RY, RN, XY, XN, HY, or HN, corresponding to *yes* or *no* for each request.

The secondary system's S and R replies are based on permission to access the requested file or directory. The permissions come from the USERFILE and from the read and write permissions of the requested file or directory. After each file is copied into the spool directory of the receiving system, the receiver of the file sends a C (copy complete) message. It sends CY if the file has successfully moved from the spool directory to the destination. Otherwise, it sends CN, and the file is put in the public directory, usually **/usr/spool/uucppublic**, and the requester gets a mail notice.

The response to the H message is determined by a scan of the secondary system's spool directory. If work for the primary system exists, the secondary system sends HN, and the two systems switch roles. If no work exists, HY is sent.

5. Logs job requests and completions.

   Both systems log the requests and the results. See "LOG (Log Entry Files)" on page 4-55.

6. Ends the conversation.

   When the primary system receives HY and the secondary one echoes it back, both systems turn off the protocols and send a final oo (over and out) to each other.

Generally, another program (**uucp**, **uux**, **uuxqt**, or a prior request to **uucico**) or a system daemon starts **uucico** running. You also have direct access to it, usually for testing purposes.

Two of the command's three flags are useful for finding problems in UUCP. When you key in `uucico -d`*directory*, with an actual directory name in the *directory* field, the command uses the specified directory for storing files, instead of the spool directory. This feature gives you a safe place to examine the first step of the UUCP mechanism before the files are in a position to be moved to a remote system.

When you key in uucico -x*num*, with a number between 1 and 9 in the *num* field, you get additional information. The higher the number, the more information you get. For example, entering the command /usr/lib/uucp/uucico -r1 -sbravo -x9 may produce the following type of output. Only the first few lines are shown here, but you usually get much more:

```
**START**
UID  user zomix
uucico -x9 -sbravo
finds called
getto called
call no.8385530< for sys bravo dial 8385530<
suspend tty0
dc - /dev/tty0, acu -/dev/tty0
ACU write ok
dcf is 6
dcr returned as 6
login called
wanted "" got that
wanted gin: login: got that
wanted ord: \40\15\12login:
      \40uucp\15\12Password: got that
```

If you have trouble logging in to a remote system, and most people do at first, you can use the **uucico** command to tell UUCP to display at your terminal a transcript of the login attempt. Following is a series of commands to help you do this:

```
cd /usr/spool/uucp
rm -f STST.*  #(Status files, useless while tracing errors)
/usr/lib/uucp/uucico -r1 -sremote_sys -x9
```

The flags used here with **uucico** are the following:

- `-r1` Sets the local system to primary mode.
- `-s` Says to call the remote system named after `-s`.
- `-x9` Gives the most tracing information, `-x1 the least.`

Using the UUCP ERRLOG File

UUCP creates this file in the spool directory to record UUCP system errors, which should not happen often. The messages come from the ASSERT statements in the various programs. Wrong modes on files or directories, missing files, and read/write system call failures on the transmission channel can cause entries in the **/usr/spool/uucp/ERRLOG** file.

Cleaning up Lock Files

UUCP creates a *lock file* for each device in use and each system conversing. This action prevents duplicate conversations and multiple attempts to use the same device. The form of the system lock file is **LCK..***sys*, where *sys* is a system name. Device lock files have the form **/etc/locks/***dev*, where *dev* is the simple name of the device. You can leave the files in the spool directory if UUCP fails, for example if the system goes down. The files are reused after 1.5 hours. If you do not want to wait that long before retrying, remove the lock files.

Customizing the UUCP Facility

After you have installed Extended Services, supplied with your system, you do not have to take any further steps to install UUCP. However, you need to customize it before you can use it successfully. UUCP needs to know what kinds of communication links it can use and which systems it can communicate with. It also needs to know which local and remote users can transfer which files and what commands can be executed remotely. Once you have customized the UUCP facility, you must configure UUCP. See "Configuring UUCP" on page 5-4.

UUCP User ID and Files

The **/etc/passwd** file must contain a user ID named *uucp* with the following characteristics:

Field	**Description**
login	uucp
password	Your choice.
uid	5
gid	1
directory	/usr/spool/uucppublic
program	/usr/lib/uucp/uucico

The following directories, also necessary to run UUCP, came with your system, but it is a good idea to check the owner and protection values:

Name	**Owner**	**Protection**
/usr/lib/uucp	uucp	0755

/usr/spool/uucp	uucp	0777
/usr/spool/uucppublic	uucp	0777
/usr/lib/uucp/.XQTDIR	uucp	0777

For UUCP to work, do not place more restrictive protections on spool directories than those in the list.

You may want to verify the existence of the following files. They should have the listed owner IDs and file protections:

Name	**Owner**	**Protection**
/usr/lib/uucp/L-devices	uucp	0600
/usr/lib/uucp/L-dialcodes	uucp	0644
/usr/lib/uucp/L.sys	uucp	0600
/usr/lib/uucp/QTCMDS	uucp	0600
/usr/lib/uucp/SEQF	uucp	0666
/usr/lib/uucp/SQFILE	uucp	0600
/usr/lib/uucp/USERFILE	uucp	0600
/usr/lib/uucp/uucico	uucp	4111
/usr/lib/uucp/uuclean	uucp	4111
/usr/lib/uucp/uusub	uucp	0100
/usr/lib/uucp/uuxqt	root	4111
/usr/bin/uucp	uucp	4555
/usr/bin/uulog	uucp	4555

/usr/bin/uuname	uucp	4555
/usr/bin/uustat	uucp	4555
/usr/bin/uux	uucp	4555

Some of the files in **/usr/lib/uucp** may need editing, as explained in the following list:

Name	**Description**
L-devices	File describing the communication lines that UUCP can use.
L-dialcodes	File describing phone number abbreviations, but frequently empty.
L.sys	File describing the remote systems you can contact.
QTCMDS	File listing the commands that remote UUCP users can run on the local system.
SEQF	File containing a four-digit number that UUCP uses for forming spool file names.
SQFILE	File that optionally keeps track of the number of conversations you have with each remote system, but frequently empty.
USERFILE	File describing the local files that UUCP permits remote and local users to access.
uudemon.hr	Shell script to be run hourly from crontab.
uudemon.day	Shell script to be run daily from crontab.
uudemon.wk	Shell script to be run weekly from crontab.

Choosing a Name for Your Site (chparm)

If you have not selected a site name, you should choose one no longer than seven characters in lowercase. Since there are already several thousand machines that communicate by means of UUCP, the name should be unique and memorable. Names like *machineA* or *sys2* are not good choices. To name your site **grendal**, do the following:

```
chparm nodename=grendal
```

Note: To have the name take effect, you must reboot the system.

Defining Outgoing Physical Links (L-devices)

The **L-devices** file defines the physical intermachine links used to make calls. Each line in the file has the following format:

type device dialer speed

An explanation of each field follows:

Name	Description
type	You can specify one of these four types: • DIRECT: A line permanently cabled to the other system. • MANUAL: A line with a modem, but dialed manually. • ACU: A call-out line with a dialer attached by a DN-11 or other similar device. • ACUdialdev: A call-out line with an autodialer modem of type *dialdev.* See the last chapter of this guide for information on supported autodialers.

device	The simple name of the device. If the device is named **/dev/tty1**, place **tty1** in this field.
dialer	The simple name of the dialer device, if different from the **device**. Otherwise, place the simple device name in this field.
speed	The line speed to be used, measured in bps. Examples of acceptable values are 300, 1200, 4800, 9600, and 19200. If a line can accommodate more than one speed, place an entry for each speed in the file, using the same device name. The entry for **/dev/tty1** to call out at 1200 bps through a modem of type **hayes_1200** is as follows:

ACUhayes_1200 *device_name* tty1 1200

If the local machine is a secondary or call-in site and never initiates dialing out to a remote system, the **L-devices** file should be empty.

An example of an entry for a direct connection is ths following line:

```
DIRECT tty5 tty5 19200
```

An example of an entry for a modem connection is the following line:

```
ACUhayes_1200 tty4 tty4 1200
```

Phone Number Abbreviations (L-dialcodes)

If your telephone system uses tie lines or other dialing codes, you can define and store number prefixes in the **L-dialcodes** file. Each entry has the following form:

code number

The *code* field contains a short alphabetic code that you define to represent a dialing code, and the *number* field contains the dialing prefix to be used. For example, if your dialcodes file had the entry `nyc 9-1-555-`, then a phone number in **L.sys** called `nyc111-1111` would be dialed as `9-1-555-111-1111`.

Dial codes are useful for tie lines or for codes required by long distance telephone companies.

If you do not use dial codes, specify the full phone number in the **L.sys** file entry.

Permitted File Access (USERFILE)

The USERFILE contains information on the users that can access certain files. The file has four fields in each line:

- The files accessible to a normal user on the local system
- The files accessible to a remote computer
- The login name for each remote computer that communicates with the local system
- The call-back field that tells whether a remote system attempting to login to the local one should have its identity confirmed by the local system calling it back.

Each entry in the USERFILE has this format, with the { } (braces) showing optional fields:

login,sys {c} path_name {path_name}

Name	Description
login	The login name for a user or a remote computer
sys	The system name of a remote computer
c	The optional *call-back required* flag
path_name	A path name prefix acceptable for the machine named in the *sys* field.

The four permitted file access fields work in the following implementations:

- When the program is running a command stored on the local machine in primary or call-out mode, the allowable path names are those given in the first USERFILE entry that contains the login name of the user issuing the command. If the file does not have such an entry, the first entry with a *null* login name is used.

- When the program is responding to a command from a remote machine in secondary or call-in mode, the allowable path names are those in the first USERFILE entry that contains the system name matching the remote machine's name. If the file does not have such an entry, the first entry with a *null* system name is used.

- When a remote computer logs in, the login name it uses must appear in the USERFILE. The file may contain several entries with the same login name, but one entry must hold either the name of the remote system or a *null* system name.

- If the login entry matching the remote system's login name contains a **c**, the local system hangs up and calls the remote system back before permitting any transactions. This helps to prevent unauthorized use of the local system.

In the following example entry, machine `bon` with login name `iota` can request the transfer of files that have names starting with `/usr/mobiz`:

```
iota,bon /usr/mobiz
```

In the next example, any remote machine can login with login name *uucp* since no system name is specified, but it can transfer only those files in the spool directory:

```
uucp, /usr/spool
```

To give the *uucp* login access to file /usr/joy as well, add the name of the file to the entry:

```
uucp, /usr/spool /usr/joy
```

The next example permits any login name from any system to access any file whose name starts with */usr*:

```
, /usr
```

It is possible, though rare, to give an ordinary user, mickey for example, permission to issue commands for file names that start with /usr/mickey. The following entry accomplishes this:

```
mickey, /usr/mickey
```

Note: The owner of a file to be transferred must set the permission bits so that the file is readable by anyone.

System Names (The L.sys File)

The **L.sys** file stores names and descriptions of remote systems that the local system can communicate with. Each entry in the file has the following format:

name time device speed phone login

There may be several entries for a system if you can contact it with several different phone numbers or several baud rates. Following is an explanation of each field in a file entry:

Name	Description
name	The UUCP site name of the remote system.
time	The *days* of the week and *times* of day when the local system is permitted to call the remote system. The *days* portion of the field may contain any combination of *Mo Tu We Th Fr Sa Su Wk Any*, where *Wk* means any weekday and *Any* means any day at all. The *times* field should use the 24-hour clock format. If the field is blank, any time is acceptable. The following

example lets you call only outside the 9-5 business day and has the system retry every minute, on unsuccessful attempts, instead of using the default retry time:

```
Wk0000-0859,Wk1701-2400,Sa,Su,0001
```

device	The device to be used for making the connection. If the remote system is directly connected with a cable and does not use a modem, place the device name, such as **tty17**, in the field. For a call-out line, place **ACU** in the field. If the particular remote system that this entry describes is only a call-in site and is never to initiate contact with the local sytem, place *SLAVE* in the field.
speed	The line speed in bps, but omit the speed if you never call this system.
phone	The phone number to dial, but omit it if you never call this system.
login	The login sequence for the remote system, since it usually logs in as user *uucp*. The entry has the following format:

expect send expect send ...

The *expect* field contains the string expected from the remote system, and the *send* field contains the string the local system sends when it receives the expected string.

The *expect* field may have the extended form, *expect-send-expect-send-....*

This can permit several remote login attempts where *send* is sent if the previous *expect* string is not received.

Special *send* values are the following:

- EOT sends an end of transmission character

- "" sends a *null* send field ("" symbols are not otherwise special).

The *send* field normally has a new-line character appended.

In both the *expected* and the *send* fields, the following escape sequences have a special meaning:

Symbol	**Meaning**
\n	New-line (line-feed) character
\r	Carriage return character
\s	Space character
\t	Horizontal tab character
\c	Special indicator that means not to append a new-line character to this field when it is sent.

A typical **L.sys** entry to dial remote system broo that the local system calls through a modem is the following example:

```
broo Any ACU 300 5551111 login:-BREAK-login: uucp ord: bliz
```

The local system usually accepts incoming calls at any time. Permitted times refer only to outgoing calls. A typical entry to allow remote system fring to call into the local system over a direct connection is the following:

```
fring Any Slave
```

Note: Since the **L.sys** file contains sensitive security information, the login sequences and passwords for remote computers, make sure the file remains protected with at least 0600 permissions and is owned by *uucp*. Users generally have no need to examine the **L.sys** file directly, since **uuname** gives them access to the names of systems they can contact.

L.sys file directly, since **uuname** gives them access to the names of systems they can contact.

The QTFILE File

The **uux** command can run only those commands listed in QTFILE. To allow a remote user to run any command on the local system, put *Any* as the first line in the file. Most systems severely limit remotely executable commands. To accomplish this, list the permitted commands in **/usr/lib/uucp/QTCMDS**. See also "The uux Command" on page 4-18.

The QTCMDS file should contain, at a minumum, the following command:

```
rmail Receive network mail
```

Other commands can be added as appropriate, such as `print`, `ls`, `man`, `cmp`, and `diff`.

The L.sys File (Special Notes)

Arranging the contents of the **L.sys** file to login successfully to a remote system takes some practice. If the local system dials the remote one directly, rather than through a port selector or data PBX, and the remote system has a normal login sequence, the following entry in the **L.sys** file should work for system `zo` with password `bo`:

```
zo Any ACU 1200 5551111 gin:-BREAK-gin: uucp ord: bo
```

This says to wait for the remote system `zo` to say `gin:` (the last part of its **login:** prompt). If it does not, send a break signal and try again. Then send `uucp` (the login name), and wait for `ord:` (the last part of the password prompt). Then, send the password, `bo` to log into the remote system.

By default, UUCP sends a new-line character after each string. You can suppress it by ending the string with the \c sequence.

If the remote system expects the local system to send first, you can put an explicit null in the *expect* field with two double quotes. In the following example, the remote system expects a null string and a carriage return without a new-line before it sends a login prompt:

```
zo Any ACU 1200 5551111 ""\r\c gin:-BREAK-gin: uucp ord: bo
```

Each entry in the **L.sys** file can be up to 300 characters long, and entries are often up to 150 characters long because of the special nature of the login sequence. Since call-in sequences, login prompts, user names, and passwords differ from system to system, one of the most important parts of setting up the UUCP facility is getting the login sequence correct for each system you plan to communicate with. See also "The uucico Command and Problem Determination" on page 4-31.

Example: Modem Connect for a Call-Out Port Setup

Following is an example of the steps you can take to customize and configure the UUCP facility for a call-out port.

Characteristics

modem :	Hayes 1200 baud
local sitename :	earth
device name :	tty1
remote sitename :	moon
phone number	9-1-555-999-9999
login name to remote site :	uucp
password :	hoo

Steps:

1. Type the command line `chparm nodename=earth`.
2. Add this entry to **/usr/lib/uucp/L-devices:**

   ```
   ACUhayes_1200 tty1 tty1 1200
   ```

3. Add this entry to **/usr/lib/uucp/L-dialcodes**:

   ```
   aus 9-1-555-999-
   ```

4. Add this entry, all on one line, to **/usr/lib/uucp/L.sys**:

   ```
   moon Wk0800-1700 ACU 1200 aus9999 gin:-BREAK-gin: uucp
   ord: hoo
   ```

   This line allows calls to site *moon* on weekdays from 8 a.m. to 5 p.m.

5. Add these commands to **/usr/lib/uucp/QTCMDS**:

   ```
   rmail
   rnews
   who
   echo
   uumove
   ```

   This allows remote UUCP users to run **rmail, rnews, who** and **echo** and allows local UUCP users to run **uumove**.

6. Add device **tty1** by entering the following:

Enter	Comment
devices	(in response to the $ prompt)
add	(in response to the devices menu)
ttydev	
tty	
rs232c1	(the first 232C, 4-port asynchronous adapter)
1	(port 1)
tt dumb	(set up as a dumb terminal)
ae false	(automatic enable is false; do not allow people to login through this port)
Press enter	
yes	(you wish to change other options)
bpc 8	(data is sent in 8 bits per character)
pt none	(parity is none)
rts 1200	(receive and transmit speed is 1200 bps)
sns true	(transmission is by modem over telephone line)
aa false	(no automatic answer for call-out)
nr true	(no ROM)
ddbw 8	(device data bus is 8 bits wide)

dmas false	(no Direct Memory Access)
eil true	(enable interrupts)
sil true	(have shared interrupts)
dvam 1	(device attached by modem)
nosb 1	(data has one stop bit)
om full	(mode of operation is duplex)
pro dtr	(protocol is dtr--data terminal ready)
ixp false	(do not include Xon/Xoff protocol)
	(use the default for all other options)
Press enter	(finished changing options)
Press enter	(device will now be set up)

Example: Modem Connect for a Call-In Port Setup

Following is an example of the steps you can take to customize and configure the UUCP facility for a call-in port.

Characteristics:

modem :	Hayes 1200 baud
local sitename :	earth
device name :	tty1
login name to local site :	uucp
remote sitename :	moon

Take the following steps:

1. Type the command line `chparm nodename=moon`.
2. Add the following to **/usr/lib/uucp/L-devices**:

   ```
   ACUhayes_1200 tty1 tty1 1200
   ```

3. Add the following to **usr/lib/uucp/USERFILE**:

   ```
   uucp,moon /usr/spool /tmp
   ```

   This allows remote system `moon` UUCP access only to files with pathnames **/usr/spool** and **/tmp**.

4. Add device **tty1** by entering the following:

Enter	**Comment**
devices	(in response to the $ prompt)
add	(in response to the devices menu)
ttydev	
tty	
rs232c1	(this is the first 232C, 4-port asynchronous adapter)
1	(port 1)
tt dumb	(terminal type is dumb terminal)
ae true	(automatic enable is true; allow people to login through this port)
Press Enter	
yes	(you wish to change other options)
bpc 8	(data is sent in 8 bits per character)
pt none	(parity is none)
rts 1200	(receive and transmit speed is 1200 bps)
sns true	(transmission is by modem over telephone line)
aa true	(automatic answer for call-in)
nr true	(no ROM)
ddbw 8	(device data bus is 8 bits wide)
dmas false	(no Direct Memory Access)
eil true	(enable interrupts)
sil true	(have shared interrupts)

dvam 1	(device attached by modem)
nosb 1	(data has one stop bit)
om full	(mode of operation is duplex)
pro dtr	(protocol is dtr--data terminal ready)
ixp false	(do not include Xon/Xoff protocol)
	(use the default for all other options)
Press Enter	(finished changing options)
Press Enter	(device will now be set up)

5. Type `penable tty1` to complete the set up for the call-in port.

Administration

There are several events and files to administer for the UUCP facility. You can accomplish some administrative tasks through *shell files* initiated by **crontab** entries. Others require manual intervention. See also "Using uudemon Files for Automatic File Deletion" on page 4-25.

SQFILE (Sequence Check File)

This file is set up in the **uucp program** directory and contains an entry for each remote system with which you agree to perform conversation sequence checks. The initial entry is the system name of the remote system. The first conversation adds the conversation count as well as the date and time of the most recent conversation. These items are updated with each conversation. If a sequence check fails, you then need to adjust the entry manually.

SQFILE (Sequence Check File)

This file is set up in the **uucp program** directory and contains an entry for each remote system with which you agree to perform conversation sequence checks. The initial entry is the system name of the remote system. The first conversation adds the conversation count as well as the date and time of the most recent conversation. These items are updated with each conversation. If a sequence check fails, you then need to adjust the entry manually.

TM (Temporary Data Files)

These files are created in the *spool* directory while a file is being copied from a remote machine. The name of a temporary file has the following format:

TM.pid.ddd

The *pid* is a process ID and *ddd* is a sequential three-digit number starting at 0. After the spool directory receives the entire file that is being copied into the **TM** file, this file is moved or copied to the requested destination. If processing ends abnormally, this file remains in the spool directory. You should periodically remove them with the **uuclean** command. The following line removes all **TM** files older than three days:

```
/usr/lib/uucp/uuclean -pTM
```

LOG (Log Entry Files)

While UUCP is running, log information is added to the **LOGFILE**. If this file is locked by another process, the log information goes in individual log files that have the prefix **LOG**. You can combine these files and add them to information already in the **LOGFILE** with the **uulog** command. See *AIX Operating System Commands Reference* for information on flags to use with this command.

The **LOG.files** are created initially with mode 0222. If the program that creates it ends normally, the mode changes to 0666. If it ends abnormally, it may leave the mode at 0222. If so **uulog** cannot read or remove the file. To remove the file, use **rm** or **uuclean** or change the mode to 0666 and let **uulog** merge it into the **LOGFILE**

STST (System Status Files)

The **uucico** program creates these files in the spool directory. They contain information on login, call-out, or sequence check failures or, while two machines are communicating successfully, they contain a **talking** status. The name of each file has the following format, where *sys* is the name of the remote system:

STST.*sys*

For ordinary failures, such as login or dialing, the file prevents another try for a default time of about 55 minutes.

For more serious sequence check failures, you must remove the file before trying again to communicate with that system.

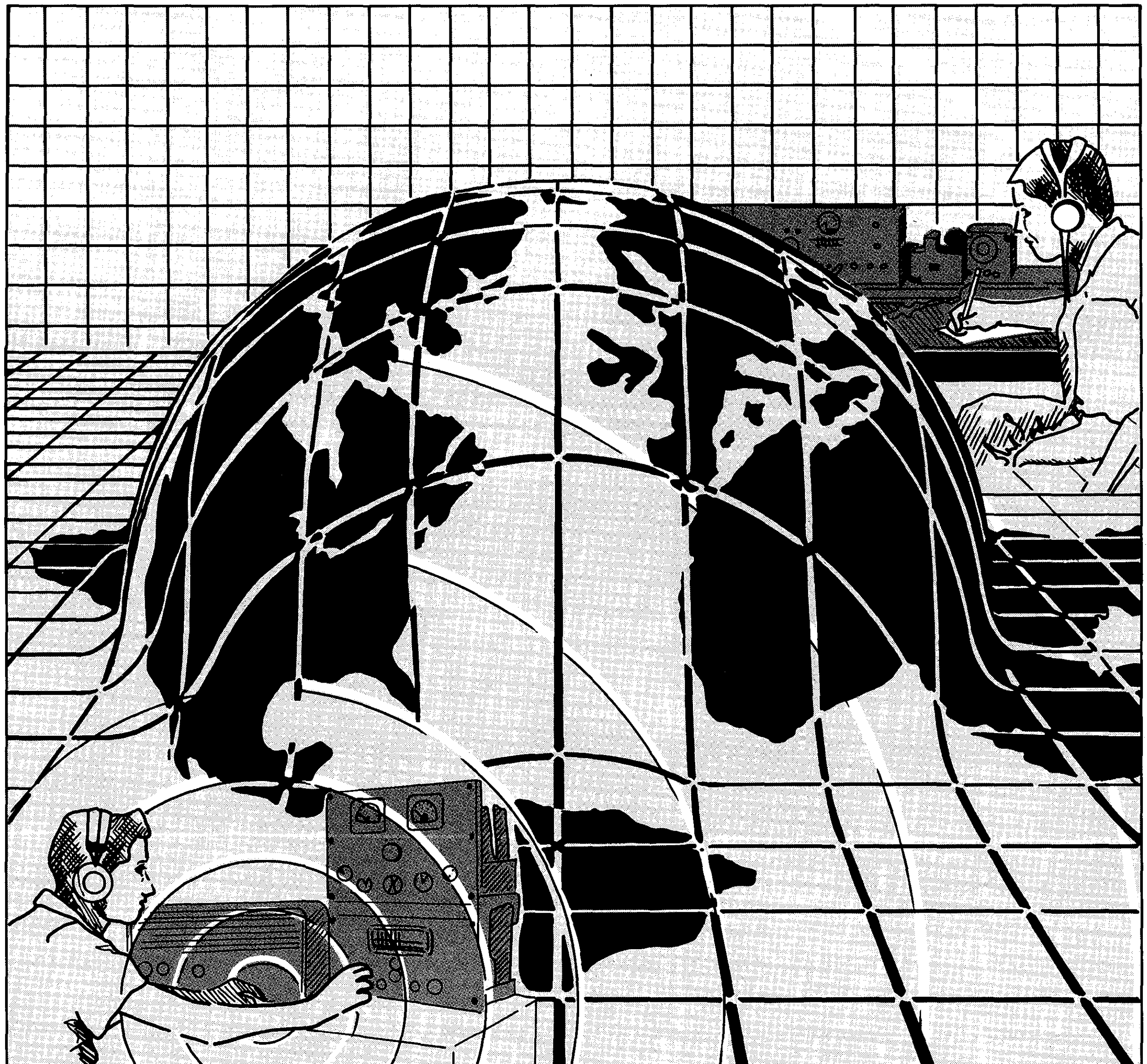
Other Administrative Files

For information on the ERRLOG file, see "Using the UUCP ERRLOG File" on page 4-35.

For information on lock files, see "Cleaning up Lock Files" on page 4-35.

For information on shell files, see "Using uudemon Files for Automatic File Deletion" on page 4-25.

Chapter 5. Configuring for Communications



CONTENTS

About This Chapter

This chapter gives information on using ports for communications. It also tells you how to interconnect cables for direct connections and how to use modems with telephone lines.

Configuring UUCP

This section explains the following:

- Getting started with ports
- How to set up a port
- How to use use port commands.

Getting Started with Ports

A *port* is a logical connection between a computer, such as IBM RT PC Model 20 and Model 25, and another unit (a computer, a printer, or a terminal, for example). Its physical component is one of several types of adapters. It also has programs and files associated with it.

These are some examples of the physical connection supporting the port:

- A direct cable on which a specific terminal is permanently connected (hardwired) to the computer
- A modem and common carrier line for communications over short or long distances
- A modem eliminator, or device that functions like a modem.

Communications Ports

At a given time, a port is set up as a **call-out** or a **call-in** port. On many systems, you can use the same port for both functions by changing the configuration with port commands. For more information on **penable**, **pdisable**, and **phold**, see "How to Use Port Commands" on page 5-5.

A call-out port is the gateway that gives you access to the facilities of remote systems even though these are not directly connected to your own terminal. The facilities might include printers, disk drives, files, plotters, and library routines. The call-out port initiates the connection and is termed the *primary* site. A call-out port is DISABLED for logins.

A call-in port, supported by an asynchronous adapter or other hardware, makes it possible for a remote system to contact and use your resources. A system having only a call-in port is called a *secondary* site. A call-in port is ENABLED for logins.

How to Set Up a Port

To add a port, you should use the **devices** command. (You may also use this command to delete, change, and show any current device). Information on devices and the **devices** command is in *Installing and Customizing the AIX Operating System.* See also "Example: Modem Connect for a Call-Out Port Setup" on page 4-48 and "Example: Modem Connect for a Call-In Port Setup" on page 4-52.

How to Use Port Commands

Sometimes it is useful or necessary to disable a port, and then enable it later, so that another system user will not be able to login while a process is running.

Three commands temporarily override the **enabled** value set by the **devices** command. (See the automatic enable option in the **devices** command). These commands can make ports available or unavailable for logging in:

- **penable** requests that loggers be created for the specified port or ports so that a person or another system can log in and gain access to the system.
- **phold** arranges for ports to become disabled and therefore unavailable to other system users as soon as the current one logs out.
- **pdisable** prevents running any loggers on the specified port or ports, making it impossible for a system user to login.

These commands work by changing the status of the port or ports in the **/etc/portstatus** file. This precautionary feature insures that the effect of the three commands will be temporary rather than permanent. Their influence will disappear when the system is rebooted.

You can use these three commands to change a single port or a class of ports. Examples and explanations follow:

- `pdisable term = dialin`

  Keying in this line will disable all ports configured for calling in.

- `penable speed = 9600`

  Keying in this line will cause all ports with 9600 baud to be enabled so that system users can login.

A Note on the Minor Device Number

In addition to the electrical and hardware information contained in *User Setup Guide*, you need to look at the special file associated with the device on a port.

Asynchronous adapters that support ports are character, rather than block, devices. Each device has a major device number that shows which device driver in the operating system is to handle I/O requests for that device. Each device also has a minor device number, but the device driver may interpret the minor device number in one of several ways. In the case of an asynchronous I/O device, the device driver uses part of the minor device number to select the port and uses another part of the number to determine whether modem control (DCD) or a direct connection should be used on the port.

Bit 4 determines whether the adapter will have modem control or not. Set it to 1 to enable it for modem control; make it 0 to disable it for no modem control.

For example, a minor device number between 16 (10000 in binary) and 32 (11111 in binary) indicates an adapter with modem control enabled for use.

Diagnosing a Port Failure Problem

Hardware problems in a port failure may occur because of difficulties in a piece of communications equipment rather than in the port itself. You may need the help of the telephone company or communications services and the equipment vendor. Failures sometimes occur in the communicator (a *rotary* that permits several numbers to be reached by dialing one number), in the Data Access Arrangements (DAAs), or in modems.

Sometimes the problem is line noise. Terminals may pick up noise if they are not properly terminated. Some modems make noise when no connections have been established or when they are busied out. The noise shows up as transients on the modem control signal *carrier detect*. The modem may be connecting and then immediately disconnecting many times a minute. As soon as a line becomes active, it immediately goes inactive, sending a hangup signal to the process that was waiting for the line. The process halts and **init** starts another one to take its place. Since this action may happen many times a minute, it severely degrades the performance of the system. The system initialization process monitors the activity on each line to prevent this from happening. If a line goes through a certain number of loggers in a specified period of time, an error message goes to the operator's console and the line is disabled for a few minutes. You should investigate any line that is disabled often.

Other problems may include a port that has not been properly closed and made available to someone else because the previous system user left a process running instead of completing the hang up operation. The terminal driver continues to pay attention to the line, even though there is no carrier on it, and may pick up line noise. You can find these problems with the **ps** command and then get rid of them with the **kill** command.

Also, if the special file for a port can not be opened, the **logger** program will end as quickly as it begins. Check the special file to make sure its values are correct. When you issue an **open** call on a communications port, it will hang until it receives a carrier detect signal.

Another way to approach problems is to use the -**d** flag with the **getty** program. This helps you find errors in the device configuration. If you invoke `getty -d`, it does not actually run the logger or other program it is supposed to run. Instead, it displays on your screen the name of the logger or other program it would have run. You can then see if the name is the one you expect.

Also see "Suggestions for Tracing Port Failures" on page 5-15.

Interconnecting Terminals and Modems

This section explains how to connect terminals and modems to a system through asynchronous, duplex ports that implement all or part of the RS-232C or the RS-422 protocol. For further information, see *User Setup Guide.*

Cables and Connections

You can use one of three cards or one of the serial ports on the system board for communications between the local system and a remote one:

- An AT serial/parallel card (for a modem attachment)
- A multi-port RS-232 card (for a modem attachment or a direct connection)
- A multi-port RS-422 card (for a direct connection).

There are certain restrictions on the cabling you can do with each card:

- No cable is provided with the AT serial/parallel card or with the system planar. In a two-system configuration, you can use the AT serial/parallel card or the system board at only one end of the connection. At the other end, use a multi-port card (either RS-232 for a modem attachment or direct connection or RS-422 for a direct connection). You also need a 9-pin adapter cable for the AT serial/parallel card and a 10-pin adapter cable for the system planar.

- A 20-meter cable is available for use with the RS-232 card. You can also install your own cable up to a maximum of 4000 feet for the RS-422, if you also have adequate surge protection, and a maximum of 50 feet for the RS-232.

Drawings of the various cable connections are in Chapter 2 of *User Setup Guide.*

Additional Information

Generally, the RS-232C standard requires that you connect devices with 25-pin connectors. It recognizes two types of equipment: Data Communications Equipment (modems or DCEs) and Data Terminal Equipment (terminals and computers or DTEs). The function of a particular device as a modem or as a terminal or computer determines how it uses the particular pins on its connector.

The standard also specifies that modems have receptacle connectors while terminals and computers have plug connectors. However, the standard does not always apply to DTEs (terminals and computers). In one case, a DTE has a receptacle connector but still functions as a DTE. In another case, a DTE has a receptacle connector and functions as a DCE (modem) so that local terminals can be plugged directly into it. A receptacle connector usually, but not always, indicates a DCE (modem) *function*, even though the particular piece of equipment may be a computer or terminal.

Note: You should have a thorough knowledge of RS-232C protocol and access to a Volt-Ohmmeter (or preferably a breakout box) to correct any problems you may run into while interconnecting terminals or computers and modems. The reason is that different pieces of communications hardware may implement different subsets of the EIA RS-232C protocol. You may have to create a customized cable to interconnect the hardware.

Computer-to-Modem Cables

When connecting a modem to a computer that functions as a DTE (as the standard specifies), you need a *straight-through* cable. Typically, it connects the following pins on the connector at one end to the same pins on the connector at the other end:

1	Protective ground (connected to the cable shield, if there is one)
2(XMT)	Transmitted data (passed from the terminal to the modem)
3(RCV)	Received data (passed from the modem to the terminal)
4(RTS)	Request to send (a signal that the terminal generates when it is ready to send data to the modem)
5(CTS)	Clear to send (a signal that the modem generates to permit the terminal to send data)
6(DSR)	Data set ready (a signal that the modem generates to indicate that it is ready for the terminal to use it)
7(GND)	Signal ground (a standard--all other signals are measured relative to the signal ground)
8(DCD)	Data carrier detect (a signal that the modem generates when a connection has been established)
20(DTR)	Data terminal ready (a signal the terminal generates to indicate that it is ready for communications)

Note: Some modems may ignore pins 4, 5, and 20, or may require other signals. Your modem instruction manual should give you this information.

Most modems do not answer an incoming call unless the terminal has generated the DTR (data terminal ready) signal. Also, most modems hang up on the caller if the DTR signal is broken during a transmission.

Most duplex modems generate DCD (data carrier detect) and DSR (data set ready) signals where a connection has been established. However, a modem that comes with switches that generate DCD and DSR at all times should have the switches removed when the modem is used for incoming calls. The manufacturer's specifications should contain information on what other signals the particular modem may require or generate.

Terminal-to-Computer Cables (Null Modems)

When two devices that both function as DTEs (terminals or computers) are to be directly connected, the interconnecting cable must transpose most of the signals. This kind of cable, called a *null modem*, is usually required to connect a terminal directly to a computer port. Typically, a null modem is a cable with connectors wired symmetrically so that on each end the pin arrangements are the following:

- Pin 7 (GND or signal ground) is passed straight through to the other end.
- Pin 2 (XMT or transmitted data) is passed to pin 3 (RCV or received data) on the other end.
- Pin 3 (RCV or received data) is passed to pin 2 (XMT or transmitted data) on the other end.
- Pin 6 (DSR or data set ready) is passed to pin 20 (DTR or data terminal ready) on the other end.
- Pin 20 (DTR) is passed to pin 6 (DSR) on the other end.
- Pins 6 (DSR) and 8(DCD) are shorted locally at each end.
- Pins 4 (RTS) and 5 (CTS) are shorted locally at each end.

Transposing 2 (XMT) and 3 (RCV) passes data sent by one side into the received data signal of the other side.

Transposing 6 (DSR) and 20 (DTR) allows each side to inform the other of its readiness to communicate.

Locally shorting 6 (DSR) and 8 (DCD) allows both of these signals to be driven by the other side's 20 (DTR). Different devices have different configurations. Some wait to receive DSR, some wait to receive DCD, and some wait to receive both. Having the other side's DTR drive both signals should cover all cases.

Locally shorting 4 (RTS) and 5 (CTS) gives each side permanent permission to transmit data. This action may be necessary to achieve duplex communication with some equipment.

If you have only *3-* or *4-conductor* cable available, you can achieve similar results with the following wiring:

- Wire pins 2, 3, and 7 the same way as for straight-through cable.
- Locally short pins 4 connected to 5 and 6 connected to 8 the same way as for straight-through cable.
- Remove the wire from pin 20 on the computer end to pin 6 on the terminal end. This action removes the status indication from the computer to the terminal and also prevents using the DCD (carrier detect) on the terminal as an indicator that the system is up. Replace the removed wire with a jumper from pin 20 to pin 6 on the terminal end. This causes the terminal to drive its incoming DCD (carrier detect) signal with its outgoing DTR (data terminal ready).
- Also remove the wire from pin 20 on the terminal end to pin 6 on the computer end. This action removes the status indication from the terminal to the computer. Replace the removed wire with a jumper from pin 20 to pin 6 on the computer end. This wiring does not let you disconnect from the computer or log out of the system merely by turning off the terminal, which is the equivalent of hanging up on a dialed connection. But if you have only three wires in the cable, this wiring plan gives you the signals you need.

If your terminal does not use anything associated with RS-232C protocol except the connector, you should study the manufacturer's documentation carefully to determine which signals the terminal must see and which ones it drives. Without this documentation, you still may be able to construct a cable through trial and error by using a breakout box.

Device Drivers and Possible Transmission Problems

After you have completed the cabling, you should test for problems. A device driver within the system controls all ports and determines many characteristics of the terminals on the system. For example, a terminal does not generate a DTR (data terminal ready) signal to a modem until there is an **open** system call pending on the associated port. A modem connected to that port does not answer incoming calls unless a process is trying to open the port. If the port is not open, input on the line is lost.

Once the modem answers and a connection is established, the modem must then generate DCD (carrier detect) for the **open** call to complete. An **open** call that never returns a value may be due to the modem's failure to generate the DCD signal.

After an **open** call completes, you may still send or receive data incorrectly because of mismatch problems in speed, parity, or character length. Typically, a newly opened terminal is set for 300 baud and 8 data bits. You may need to change the parameters associated with the port by using the **stty** or the **ioctl** command. The *AIX Operating System Commands Reference* has information on these.

If a modem connection is broken but the program does not detect the break, you should check the minor device number in the special file to make sure that modem control is enabled. Some modems have switches or straps to configure so that if the connection breaks, the DCD (carrier detect) signal also fails.

High-Speed Lines

High-speed lines (over 2400 baud) may cause several problems:

- RS-232C was designed for high-speed transmission over distances of no more than 50 feet. Cables longer than 50 feet have significant inductances and capacitances that can interfere with the pattern of high-frequency square waves. The inductive coupling rounds off the waves, while the capacitive coupling induces cross-talk between adjacent conductors. Since longer cables also experience resistive and reactive signal attenuation or reduction, you should use shielded cable. You should also consider using short-haul modems or some balanced transmission scheme for distances over 50 feet.

Suggestions for Tracing Port Failures

In addition to the following hardware diagnostics, you may want to refer to material on using **getty -d** in "Diagnosing a Port Failure Problem" on page 5-8.

If you have a line failure problem, take the following steps:

1. Check the obvious:
   - Make sure the terminal is on line.
   - See that the cables are connected.
   - Verify that the transmission speeds match.
2. Connect a different terminal and modem.
   - If a dial-in line is causing a problem for one person, see if others are having the same difficulty.
   - If not, first try to dial the modem where the problem is occurring. Swap modems if you cannot dial it; swap terminals if you can dial it.
3. Determine whether more than one port has failed, especially contiguous ones.
   - If you are using a rotary or distribution panel, check the rotary if dial-in lines are connected to adjacent ports that have failed.
4. Determine whether the problem lies on the computer side or the terminal side of the distribution panel. At the distribution

panel, switch a line that does not work with a line of the same type that does work.

- If the problem moves to the new line, there is probably a hardware or software problem within the system.

  - First, check the hardware again and make sure all cables are tight. If the problem still exists, find out what in the system has changed recently, and focus on that area.

  - Then check the software. Make sure the special file entry for the port in the **/dev** directory is still intact. Then try to use the port as an output device by using the **cat** command to read a text file and direct its output to the port's special file. If this attempt is successful, the problem is very likely in the software. If the attempt fails, the problem is very likely an electronic one.

- If the problem remains on the same line where it first occurred, there is something wrong with the wire or the equipment along its length.

  - If you have not already tested the terminal, you should do so now.

  - Do the same for the modem by swapping in an identical unit.

  - Finally, look for the problem in the phone company's network, in the rotary, in the DAA (Data Access Arrangement), or in a dial-in modem by means of the following symptoms:

    - If someone tries to dial in but gets no ring or busy signal, the problem may lie in the phone company's network.

    - If the person gets a busy signal but verifies that not all the lines are in use, first switch modems and DAAs or lines. If the problem moves with the modem, get a new unit.

- If the line rings but does not answer, the problem lies in the modem or in the DAA. But check the ring indicator on the modem to make sure the correct line is in fact ringing.

- If the line rings and answers but no carrier comes up, then most likely either the dial-in or the dial-out modem is at fault.

- If the line answers and the carrier comes up but you do not receive a prompt or the system ignores your input, your terminal or modem may be causing the problem.

Autodialer Programs

Many brands of modems can dial phone calls, but each type requires its own commands to initiate the dialing. Special autodialer programs handle these details. The system provides several, and you can also write new autodialer programs to interface with new kinds of modems as they are developed.

Invoking Autodialer Programs

All autodialer programs are in the file **/usr/lib/INnet/dialers**, so that the dialer for a Hayes Smartmodem 1200[1], for example, is called **/usr/lib/INnet/dialers/hayes_1200**.

When an autodialer program runs, the *tty* port will already have been opened on file descriptor 3. The caller passes the program two arguments: the phone number to dial and, optionally, the name of the dialer hardware to use, if it is different from the modem itself. If the call succeeds, the program exits with a return code of 0. If it fails, it returns one of the following codes with its associated meaning:

1 Trademark of Hayes Microcomputer Products, Inc.

Return Codes

1	Cannot open dialer.
2	Busy or no answer.
3	Unable to work.
4	Stopped before completion.
5	Communication failure.
6	Busy.
7	No answer.
8	Phone not working.
9	Incorrect phone number.
10	Cannot open device.

The values of 8, 9, and 10 indicate unrecoverable errors.

Phone Numbers: Many autodialer programs use the following conventions in interpreting telephone numbers. If you write your own dialer, you may want to use them:

<space>	Ignored. (But spaces may improve readability of output).
(	Ignored.
)	Ignored.
-	Ignored.
T	Tone dial.
,	5 second delay.

Switch Settings for Supported Modems

Hayes Smartmodem 1200: In the settings that follow, up means open and down means closed.

1. Up. DTR (yes/no)
2. Up. Result code (digits/words)
3. Up. Result code (yes/no)
4. Down. Echo (yes/no)
5. Up.
6. Up. Autoanswer (yes/no)
7. Up if using standard RJ11 phone jack; down if using multiline phone system. Carrier detect (yes/no)
8. Down. Command recognition (enabled/disabled)

Configure the associated terminal port with modem control. (See the earlier section on minor device numbers).

VEN-TEL MD212-Plus[2]: This modem has two switch packs, F and H, with 10 switches each. In the following settings, up means open or off and down means closed or on:

F1. Up.	H1. Up for pulse dialing; down for tone dialing (See vendor's instructions).
F2. Down.	H2. Up.
F3. Up.	H3. Down.
F4. Down.	H4. Up if phone has standard dial tone; down if not (See vendor's instructions on blind dialing).

2 Trademark of VEN-TEL, Inc.

F5. Down.	H5. Down.
F6. Down.	H6. Up.
F7. Down.	H7. Up.
F8. Up.	H8. Down.
F9. Up.	H9. Up.
F10. Up.	H10. Down.

Configure the associated terminal port with modem control.

VADIC 3451[3]***:*** This modem has two switch packs on the bottom board, with 7 switches each. Set them as follows:

A1. Off.	Prevents hanging in automatic answer mode.
A2. Off.	Prevents remote testing of modem.
A3. On.	Default vendor setting.
A4. On.	Default vendor setting.
A5. Off.	Default vendor setting.
A6. On.	Enables other switches.
A7. On.	Default vendor setting.
B1. Off.	Ignored when not testing.
B2. Off.	Default vendor setting.
B3. Off.	Default vendor setting.
B4. On.	Prevents data loss from a slightly fast transmitter.

[3] Trademark of VADIC Corporation.

B5. On.	Default vendor setting.
B6. Off.	Default vendor setting.
B7. On.	Ignored when not testing.

Also, set the Manual/Voice/Data switch on the front of the modem to the Voice position.

Configuring Supported Modems

Since most modems use a single line for both the dialer and the data transfer device, you have to send data to the modem in order to place a call. You cannot do this if the associated port is configured with normal modem control because this kind of port delays the **open** call until the connection has been established. To support an autodialer, you have to reconfigure the port without modem control and cable it accordingly. See *Installing and Customizing the AIX Operating System* for information on adding a device with the **devices** command.

Sample Autodialer Program

On the following pages is a sample C program for an autodialer. You can use it as a model for writing your own autodialer programs.

```
#
/*
 *  Hayes Smartmodem 1200
 *
 *  Makes a phone call using the Hayes autodialer
 *
/*

static char sccsid [ ] = "@(#)
hayes_1200.c   5.3 - 84/07/13";

#include <signal.h>
#include <IN/standard.h>
#include <termio.h>

#define PORT        3


main (argc, argv)
int argc;
char *argv [ ];
{    register char *p;
     int state;
     char numbuf [ 40 ];

     static struct termio hmodes;


     static char cmd0 [ ] = "ATQ0\r";
     static char cmd1 [ ] = "ATE0\r";
     static char *status [ ] =
          { "OK\r\n",
               "CONNECT\r\n",
               "RING\r\n",
               "NO CARRIER\r\n",
               "ERROR\r\n",
               CONNECT 1200\r\n",
               0};
```

```
if (!CSnil (argv [ 1 ]))
{    p = CScpy (numbuf, "ATD ");
     p = CScpy (p, argv [ 1 ]);
     p = CScpy (p, " \ r");

     ioctl (PORT, TCGETA, &hmodes);
     hmodes.c_iflag = ISTRIP | IGNBRK;
     hmodes.c_oflag = 0;
     hmodes.c_cflag &= CBAUD;
     hmodes.c_cflag |= CS8 | CREAD | HUPCL;
     hmodes.c_lflag = 0;
     hmodes.c_cc [VMIN] = 1;
     ioctl (PORT, TCSETAF, &hmodes);

     state = 0;
     for (;;)
     {   switch (state)
         {   case 0:
                     write (PORT, cmd0, (sizeof cmd0)-1);
                     ++state;
                     break;
                case 1:
                     write (PORT, cmd1, (sizeof cmd1)-1);
                     ++state;
                     break;
                case 2:
                     write (PORT, numbuf, p-numbuf);
                     ++state;
                     break;
                case 3:
                     exit(9);
          }
          switch (reply (status, 60))
          {   case -1:
                     exit(5);
                case 0:
                     break;
```

```
                                        case 2:
                                        case 4:
                                             exit(9);
                                        case 3:
                                             exit(2);
                                        case 1:

                                             hmodes.c_cflag &= ~CBAUD;
                                             hmodes.c_cflag |= B300;
                                             ioctl (PORT, TCSETAF, &hmodes);

                                             exit(0);
                                        case 5:

                                             hmodes.c_cflag &= ~CBAUD;
                                             hmodes.c_cflag |= B1200;
                                             ioctl (PORT, TCSETAF, &hmodes);

                                             exit(0);
                              }
                         }
                    }
                    exit(0);
               }


               reply (possible, timer)
               char *possible [ ];
               int timer;
               {   register char *p;
                    register char **s;
                    register int n;
                    int timeout ( );
                    char buf [ 64 ];

                    p = buf;
                    signal (SIGALRM, timeout);
                    for (;;)
```

```
          {    alarm (timer);
               n = read (PORT, p, 1);
               alarm (0);
               if (n < 0)
               {    signal (SIGALRM, SIG_IGN);
                    return(-1);
               }
               if (*p) {

                    *++p = '\0';
                    if (p >= buf + 63)
                        p = CScpy (buf, buf + 32);
                    for (s = possible; *s; s++)
                    {   if (CSlocs (buf, *s) != p)
                        {    signal (SIGALRM, SIG_IGN);
                             return (s-possible);
                            }
                        }
                    }
               }
     }

     timeout ()
     {
               signal (SIGALRM, SIG_IGN);
     }



     /* @ (#)CScpy.c 5.1 - 84/03/14 */

     char *
     CScpy(dst, src)
          register char *dst, *src; {

          if (dst && src) {
               while (*dist++ = *src++)
                      ;
          --dst;
     }
     return dst;
}


/* @(#)CSlocs.c 5.1 - 84/03/14 */

char *
CSlocs(str, pat)
     register char *str, *pat; {
     register char *strp, *patp;

     if (str)
          while (*str) {
               if (CScmpp(pat, str) == 0)
                    break;
          ++str;
          }
     return str;
}
```

Glossary

address. (1) A name, label, or number identifying a location in storage, a device in a network, or any other data source. (2) A number that identifies the location of data in memory.

addressing. (1) In data communications, the way that the sending or control station selects the station to which it is sending data. (2) A means of identifying storage locations.

American National Standard Code for Information Interchange (ASCII). The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

argument. An expression that is passed to a function, subroutine, or procedure for evaluation.

asynchronous transmission. In data communications, a method of transmission in which the bits included in a character or block of characters occur during a specific time interval. However, the start of each character or block of characters can occur at any time during this interval. Contrast with *synchronous transmission.*

authorize. To grant to a user the right to communicate with, or make use of, a computer system or display station.

auto-call unit. A common carrier device that allows IBM RT PC to automatically call a remote location.

background process. (1) An activity that does not require operator intervention that can be run by the computer while the work station is used to do other work. (2) A mode of program execution in which the shell does not wait for program completion before prompting the user for another command.

bad block. A portion of a disk that can never be used reliably.

basic addressable unit (BAU). The smallest piece of storage that can be addressed. On the IBM RT PC, a byte is the BAU.

batch compilation. A method of compiling programs without the continual attention of an operator, as a background process.

batch processing. A processing method in which a program or programs process records with little or no operator action. This is a background process. Contrast with *interactive processing.*

binary synchronous communications (BSC). A form of communications line control using transmission control characters to control the transfer of data over a communications line.

block. (1) A group of records that is recorded or processed as a unit. Same as *physical record*. (2) Ten sectors (2560 bytes) of disk storage. (3) In data communications, a group of records that is recorded, processed, or sent as a unit.

bps. Bits per second.

breakpoint. A place in a computer program, usually specified by an instruction, where execution may be interrupted by external intervention or by a monitor program.

BSC. See *binary synchronous communications*.

buffer. (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written.

bus. One or more conductors used for transmitting signals or power.

carrier. A continuous frequency that can be modulated with a second (information-carrying) signal.

carrier return. (1) In text data, the action causing line ending formatting to be performed at the current cursor location followed by a line advance of the cursor. Equivalent to the carriage return of a typewriter. A keystroke usually indicating the end of a command line. This is generally accomplished by pressing the Enter key.

character string. A sequence of consecutive characters.

clocking. In data communications, a method of controlling the number of data bits sent on a communications line in a given time.

cluster. Any configuration of interconnected workstations for the purpose of sharing resources (e.g. Local Area Networks, host attached workstations, etc.)

command. A request to perform an operation or execute a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command.

command name. (1) The first or principal term in a command. A command name does not include parameters, arguments, flags, or other operands. (2) The full name of a command when an abbreviated form is recognized by the computer (for example, print working directory for pwd).

communications adapter. A hardware feature enabling a computer or device to become a part of a data communications network.

configuration. The group of machines, devices, and programs that make up a data processing system. See also *system customization.*

coupler. A device connecting a modem to a telephone network.

customize. To describe (to the system) the devices, programs, and users for a particular data processing system.

cylinder. All disk or diskette tracks that can be read or written without moving the disk drive or diskette drive read/write mechanism.

cursor. (1) A movable symbol (such as an underline) on a display, used to indicate to the operator where the next typed character will be placed or where the next action will be directed. (2) A marker that indicates the current data access location within a file.

daemon. A process begun by the super user or its shell that can be stopped only by the super user. Daemon processes generally provide services that must be available at all times such as sending data to a printer.

data communications. The transmission of data between computers, or remote devices or both (usually over long distance).

data link. The equipment and rules (protocols) used for sending and receiving data.

data stream. All information (data and control information) transmitted over a data link.

debug. (1) To detect, locate, and correct mistakes in a program. (2) To find the cause of problems detected in software.

debugger. A device used to detect, trace, and eliminate mistakes in computer programs or software.

default. A value that is used when no alternative is specified by the operator.

default value. A value stored in the system that is used when no other value is specified.

device driver. A program that operates a specific device, such as a printer, disk drive, or display.

device manager. Collection of routines that act as an intermediary between device drivers and virtual machines for complex interfaces. For example, supervisor calls from a virtual machine are examined by a device manager and are routed to the appropriate subordinate device drivers.

dialog. The interchange of information between two people or between a person and a computer by questions and answers.

duplex. Pertains to communications data that can be sent and received at the same time. Same as *full duplex.* Contrast with *half duplex.*

editor. A program used to enter and modify programs, text, and other types of documents.

escape character. The character that changes the meaning of the characters that follow. For example, the backslash character is used to indicate to the shell that the next character is not intended to have the special meaning normally assigned to it by the shell.

file. A set of related records treated as a unit.

flag. A modifier that appears on a command line with the command name that modifies the action of the command. Flags in AIX almost always are preceded by a dash. Most commands permit the user to omit all flags.

foreground. A mode of program execution in which the shell waits for the program specified on the command line to complete before returning your prompt.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) To arrange such things as characters, fields, and lines.

group ID number. A unique number assigned to a group of related users. The group number can often be substituted in commands that take a group name as an argument.

half duplex. Pertains to communications in which data can be sent in only one direction at a time. Contrast with *duplex*.

hexadecimal. Pertaining to a system of numbers to the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

high-order. Most significant; leftmost. For example, bit 0 in a register.

home directory. (1) A directory associated with an individual user. (2) The user's current directory on login or after issuing the cd command with no argument.

input file. A file opened in the input mode.

input-output file. A file opened in the I-O mode.

interactive processing. A processing method in which each operator action causes response from the program or the system. Contrast with *batch processing*.

interface (n). A shared boundary between two or more entities. An interface might be a hardware component to link two devices together or it might be a portion of storage or registers accessed by two or more computer programs.

interrupt. (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission. (3) A signal sent by an I/O device to the processor when an error has occurred or when assistance is needed to complete I/O. An interrupt usually suspends execution of the currently executing program.

kill character. The character that is used to delete a line of characters entered after the user's prompt.

licensed programs. Software programs that remain the property of the manufacturer, for which customers pay a license fee.

local. Pertaining to a device directly connected to your system without the use of a communications line. Contrast with *remote.*

log. To record; for example, to log all messages on the system printer. A list of this type is called a log, such as an error log.

log in. To begin a session at a display station.

low-order. Least significant; rightmost.

mailbox. An area designated for storage of mail messages directed to a specific system user.

mask. A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

modulation. Changing the frequency or size of one signal by using the frequency or size of another signal.

modulator-demodulator (modem). A device that converts data from the computer to a signal that can be transmitted on a communications line, and converts the signal received to data for the computer.

multi-user environment. A computer system that provides terminals and keyboards for more than one user at the time.

network. A collection of products connected by communication lines for information exchange between locations.

node. An individual element of a full pathname. Nodes are separated by slashes (/).

nonswitched line. A connection between computers or devices that does not have to be established by dialing. Contrast with *switched line.*

online. Being controlled directly by, or directly communicating with, the computer, or both.

overflow condition. A condition that occurs when a portion of the result of an operation exceeds the capacity of the intended unit of storage.

password. A string of characters that, when entered along with a user identification, allows an operator to sign on to the system.

password security. A program product option that helps prevent the unauthorized use of a display station, by checking the password entered by each operator at sign-on.

pathname. A complete filename specifying all directories leading to that file.

permission code. A three-digit octal code, or a nine-letter alphabetic code, indicating access permissionsto a file or directory. Access permissions are read, write, and execute.

permission field. One of the three-character fields within the permissions column of a directory listing indicating the read, write, and run permissions for the file or directory owner, group, and all others.

polling. A method for determining whether each of the stations sharing a communications line has data to send.

port. (1) A part of the system unit or remote controller to which cables for display stations and printers are attached. (2) To transfer programs from one computer to another.

process. (1) A sequence of discrete actions required to produce a desired result. (2) An entity receiving a portion of the processor's time for executing a program. (3) An activity within the system begun by entering a command, running a shell program, or being started by another process.

process ID. A unique number assigned to a process that is running.

profile. Data describing the significant features of a user, program, or device.

protocol. In data communications, the rules for transferring data.

protocol procedure. A process that implements a function for a device manager. For example, a virtual terminal manager may use a protocol procedure to interpret the meaning of keystrokes.

public data network. A communications common carrier network providing data communications services over switched or nonswitched lines. an immediate message display.

recovery procedure. (1) An action performed by the operator when an error message appears on the display screen. Usually this action permits the program to continue or permits the operator to run the next job. (2) The method of returning the system to the point where a major system error occurred and running the recent critical jobs again.

run-time environment. A collection of subroutines that provide commonly used functions for system components.

sector. (1) An area on a disk track or a diskette track reserved to record information. (2) The smallest amount of information that can be written to or read from a disk or diskette during a single read or write operation.

server. A program that handles protocol, queuing, routing, and other tasks necessary for data transfer between devices in a computer system.

session. The period of time during which programs or devices can communicate with each other.

shell program. The command interpreter providing the user with an interface to the AIX kernel.

shell procedure. A series of commands combined in a file that carry out a particular function when the file is run or when the file is specified as an argument to the *sh* command. Shell procedures are frequently called shell scripts.

special character. A character other than an alphabetic or numeric character. For example, *, +, >, and % are special characters.

spool file. A disk file containing output that has been saved for later printing.

stand-alone work station. A work station that can be used to preform tasks independent of (without being connected to) other resources such as servers or host systems.

super user (SU). The user who can operate without the restrictions designed to prevent data loss or damage to the system. (User ID 0).

switched line. In data communications, a connection between computers or devices established by dialing. Contrast with *nonswitched line.*

synchronous. Occurring in a regular or predictable sequence.

synchronous data link control (SDLC). A form of communications line control using commands to control the transfer of data over a communications line. Compare with *binary synchronous communications.*

synchronous transmission. In data communications, a method of transmission in which the sending and receiving of characters is controlled by timing signals. Contrast with *asynchronous transmission.*

system user. A person, process, or other resource that uses the facilities of a computer system.

systems network architecture (SNA). A set of rules for controlling the transfer of information in a data communications network.

trace. To record data that provides a history of events occurring in the system.

track. A circular path on the surface of a disk or diskette on which information is magnetically recorded and from which recorded information is read. billing are examples of transactions.

transfer. To move data from one location to another in a computer system or between two or more systems.

transmission control characters. In data communications, special characters that are included in a message to control communication over a data link. For example, the sending station and the receiving station use transmission control characters to exchange information; the receiving station uses transmission control characters to indicate errors in data it receives.

user identification (user ID). A unique string of characters identifying an operator to the system. This string of characters limits the functions and information the operator is allowed to use. The user's ID can often be substituted in commands that take a user′s login name as an argument.

user list. A list, containing the user identification and access levels, of all operators who are allowed to use a specified file or library.

user profile. A file containing a description of user characteristics and defaults (for example, printer assignment, formats, group ID) to be conveyed to the system while the user is signed on.

voice-grade telephone line. A telephone line normally used for transmission of voice communications. The line requires a modem for data communications.

Index

Book Title ______________________ **SV21-8007-0**
Order No.

Book Evaluation Form

Your comments can help us produce better books. You may use this form to communicate your comments about this book, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Please take a few minutes to evaluate this book as soon as you become familiar with it. Circle Y (Yes) or N (No) for each question that applies and give us any information that may improve this book.

Y N Is the purpose of this book clear?

Y N Is the table of contents helpful?

Y N Is the index complete?

Y N Are the chapter titles and other headings meaningful?

Y N Is the information organized appropriately?

Y N Is the information accurate?

Y N Is the information complete?

Y N Is only necessary information included?

Y N Does the book refer you to the appropriate places for more information?

Y N Are terms defined clearly?

Y N Are terms used consistently?

Y N Are the abbreviations and acronyms understandable?

Y N Are the examples clear?

Y N Are examples provided where they are needed?

Y N Are the illustrations clear?

Y N Is the format of the book (shape, size, color) effective?

Other Comments

What could we do to make this book or the entire set of books for this system easier to use?

Optional Information

Your name ______
Company name ______
Street address ______
City, State, ZIP ______

No postage necessary if mailed in the U.S.A.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 997, Building 998
11400 Burnet Rd.
Austin, Texas 78758

Fold and tape Fold and tape

Cut or Fold Along Line

Tape Please Do Not Staple Tape

International Business
Machines Corporation
Department 997, Building 998
11400 Burnet Rd.
Austin, Texas 78758

Printed in the
United States of America

59X7668