UUCP, Remote Communications Utility

UUCP is a set of programs that together let you communicate in an unattended manner with remote COHERENT and UNIX sites. The term *UUCP* is an abbreviation for “UNIX to UNIX copy”; as its name implies, UUCP was developed under the UNIX operating system.

UUCP allows your COHERENT system to talk to other computers that also run COHERENT or UNIX. It can transmit files and mail to other systems and receive material from them, without needing you to guide it by hand every step of the way. Moreover, you can instruct UUCP to telephone other computers at the same time each day; this permits regular, orderly exchange of mail, news, and files among computers, and allows you to take advantage of lower telephone rates during off-peak periods. In a similar fashion, UUCP allows other systems to log into your system, to exchange mail or other information, and otherwise perform useful tasks.

Numerous UUCP systems have linked together to create an informal network called the *Usenet*. Many megabytes of source code, news, and technical information are available across the Usenet. Anyone who is connected to the Usenet can exchange mail with anyone else who is also connected to the Usenet. All that is required to hook into the Usenet is to obtain a UUCP connection to anyone else who is connected to the Usenet.

You can use UUCP only if you have telephone access to another computer that runs UUCP, and if your system and the remote system with which you wish to communicate have been described to each other. UUCP is standard with COHERENT and UNIX, and can be purchased for MS-DOS. If you wish to copy files from another system, you must arrange with the system administrator of that system before you can begin to use UUCP. Likewise, if you want someone else to dial into your system to upload or download files, you must first describe that system to your copy of UUCP.

Contents of This Tutorial

This tutorial describes UUCP and tells you how to set up and run your UUCP system. It contains the following sections:

- An overview of UUCP.
- How to set up your modem to dial out.
- How to set up UUCP to contact **mwcbbs**, the Mark Williams bulletin board.
- How to use the COHERENT utility **uuinstall** to set up UUCP to contact **mwcbbs**.
- How to set up your UUCP system to accept calls from remote systems.
- How to use the UUCP utilities to exchange files and mail with remote systems.
- How to debug some common problems with UUCP.
- How to administer your UUCP system.
- A brief introduction to networks.

Try as we might, there is no way to present all of UUCP in a brief tutorial. If you wish to explore the heights and depths of UUCP, we urge you to acquire the following books:

- O’Reilly, T.: *!%@:: A Directory of Electronic Mail Addressing and Networks.* Sebastapol, Calif, O’Reilly & Associates Inc., 1989.
- O’Reilly, T.; Todino, G.: *Managing UUCP and Usenet.* Sebastopol, Calif, O’Reilly & Associates Inc., 1987.
- Krol, E.: *The Whole Internet: User’s Guide & Catalog.* Sebastopol, Calif, O’Reilly & Associates Inc., 1992. *Highly recommended.*
- Seyer M.D.: *RS-232 Made Easy: Connecting Computers, Printers, Terminals, and Modems.* Englewood Cliffs, NJ, Prentice-Hall Inc., 1984.

An Overview of UUCP

UUCP is a set of programs that exchange files with other computers that run UUCP. You can set aside files or mail messages to be transferred to another computer; UUCP regularly checks to see if material has been set aside to be transferred, dials the remote system, and exchanges the files without requiring your assistance.

This appears to be a simple function, but it can be extremely useful to you. Suppose, for example, that you run a real-estate office that is a member of an organization with regional and national offices. You can tell UUCP to call your regional office each night, to send a file of your new listings and to accept a file of new listings in your district that had come from other local offices. Likewise, your association's regional office can telephone the national office each night to receive new listings in your region, which can then be passed on automatically to the appropriate neighborhood offices. All of this information can be transferred at night, when telephone rates are lowest, and without needing you to be at the console. When you come to work the next morning, you will have the latest listings instantly available on your terminal.

In brief, what UUCP offers is the ability to join a *network* of computers, in which every user of every computer can exchange information with every user on every other computer, automatically. What computer networks can do is limited only by your need to exchange information with other computer users, and by your imagination.

Implementations of UUCP

This version of COHERENT implements **Taylor UUCP**, a UUCP package written be Ian Taylor (**ian@airs.com**) with numerous contributions from other people in the USENET community. COHERENT does not implement the full suite of UUCP utilities provided with the Taylor UUCP package; however, it does implement the enough utilities for you to set up a robust UUCP site on your system.

The source code for the full Taylor distribution is available via **ftp** from various sources, and can be downloaded from **mwcbbs**. For a description of the full package, see Ian Taylor's documentation for his package, which is included with your COHERENT package.

This chapter presents examples to help you learn how to set up a simple UUCP configuration for a remote site. It does not discuss the more exotic features of UUCP; however, the information given in this chapter is sufficient for you to set up communications with most remote sites.

For more background information on the (sometimes arcane) subjects of serial ports and modems, see the Lexicon entries for **UUCP**, **modem**, **asy** (the serial-port driver), and **RS-232**. These discuss aspects of serial communication, and point you to other articles that you may find helpful.

Programs

The UUCP system uses the following programs to do its work:

uucico Call remote systems: log into the remote system and transfer files.

uuconv Convert configuration files into Taylor format. You will use this only if you are porting your system from another implementation of UUCP.

uucp UNIX-to-UNIX Copy: copy files from one computer to another. Be sure not to confuse the command **uucp** with the UUCP system, despite their similar names. (Note that the name of this utility is retained for the sake of convenience, and because the abbreviation of the phrase COHERENT-toCOHERENT Copy would remind users of the late Soviet Union.)

uuencode Translate binary files into printable ASCII characters for transmission to another system.

uudecode Translate files encoded by **uuencode** back into object code.

uuinstall This program displays a template on your screen, and helps you describe a system to UUCP relatively painlessly.

uulog Read the UUCP logs, which record what UUCP does.

uumkdir Create a directory for a remote site. This command is invoked by **uuinstall** and other programs.

uumvlog Copy the current UUCP log files into backup files. Throw away all log files older than a requested number of days. UUCP logs everything that it does; and since it does a lot, its log files can grow very large very quickly. **uumvlog** ensures both that you have enough information on your system to diagnose problems with uucp, and that the UUCP log files do not overwhelm your system.

uuname List the systems that your computer can reach.

uupick “Pick up” files that have been uploaded to your system from a remote site.

uusched This is a script which invoke the command **uucico** to call all systems that have jobs waiting for them.

uuto This is a script that invokes the command **uucp** to copy files from your system to another system via UUCP.

uutouch Create a file that triggers a call to a named remote system.

uutry Force a call to a remote system, for debugging purposes.

uux Execute a command on a remote system.

uuxqt Check directory **/usr/spool/uucp/***sitename* and execute all files therein that have the prefix “X.”

Two other programs, while not part of UUCP *per se*, are used by it:

ttystat Check the status of your asynchronous ports. If UUCP is not receiving files from other systems or not sending files to other systems, it may be because the appropriate ports have not been enabled.

mail Send “electronic mail” to another person, either on your system or on another system via UUCP.

Files and Directories

As mentioned earlier, your system can use UUCP to contact many different remote sites, and can have many different sites contact it. Each site differs from all others in many respects: by its name; by the telephone number at which you call it; by the permissions it may grant you and you may grant it; by the day of the week and the time of day during which you may wish to call it; and by the procedure you must follow to log into it. Remote sites may also differ with regard to the port by which you contact them; the manner in which you contact it (direct connect or via modem); the protocol with which you exchange files; and the name by which your system identifies itself to the remote system.

As you can see, UUCP needs a considerable amount of information before it can communicate with a remote site. UUCP reads this information from data files. The processing of setting up communicate with a remote system means that you write the correct information about that site into each of the appropriate UUCP data files. This process will be confusing at first — in part because some of the notation is rather obscure, and in part because there’s simply a lot of it, and some of the information needed may touch on aspects of your system about which you may not know very much.

Each implementation of UUCP has its own suite of data files that you must manipulate to set up a remote site. UUCP uses the following files and directories:

/etc/domain
This file lists the UUCP domain. It is read by **mail**.

/etc/uucpname
Holds the name of your system, as it is known to other UUCP sites.

/usr/bin/uucp
The **uucp** command. Copy a file to another system that runs UUCP.

/usr/bin/uulog
The **uulog** command.

/usr/bin/uuname
The **uuname** command.

/usr/bin/uudecode
The **uudecode** command.

/usr/bin/uuencode
The **uuencode** command.

/usr/bin/uupick
The **uupick** command.

/usr/bin/uuto
The script **uuto**.

/usr/lib/uucp
Contains UUCP commands and system data files.

/usr/lib/uucp/dial
This file tells **uucico** how to dial the modems on your system.

/usr/lib/uucp/port
This file describes the devices that **uucico** uses to call each remote UUCP site.

/usr/lib/uucp/sys
This file describes the remote UUCP sites that your site can call, or that can call your site. The command **uucico** (the command that actually talks with remote systems) reads the information in this file to to connect to remote systems. This file also names the directories on your system into which a each remote site may read or write files; and names the protocol or protocols that **uucico** uses to exchange files with a given remote system.

/usr/lib/uucp/uucico
The **uucico** command.

/usr/lib/uucp/uuconv
The **uuconv** command.

/usr/lib/uucp/uumkdir
The **uumkdir** command.

/usr/lib/uucp/uumvlog
The **uumvlog** command.

/usr/lib/uusched
The script **uusched**.

/usr/lib/uucp/uutouch
The **uutouch** command.

/usr/lib/uucp/uutry
The **uutry** command.

/usr/lib/uucp/uuxqt
The **uuxqt** command.

/usr/spool/logs/uucp
Log of UUCP activity.

/usr/spool/uucp/.Admin/audit.local
uucico stores logging information in this file when you invoke it with with logging specified.

/usr/spool/uucp/.Admin/xferstats
This file stores the transfer rates of files received or transmitted.

/usr/spool/uucp/.Log
Directory containing UUCP logfiles, as follows:

/usr/spool/uucp/.Log/uucico/*sitename*
/usr/spool/uucp/.Log/uux/*sitename*
/usr/spool/uucp/.Log/uucp/*sitename*
/usr/spool/uucp/.Log/uuxqt/*sitename*

/usr/spool/uucp/*sitename***/TM***
/usr/spool/uucp/.temp/*sitename***/***
These are temporary files that **uucico** generates when receiving files.

/usr/spool/uucp/*sitename***/C.***
Files that instruct the local system either to send or to receive files.

/usr/spool/uucp/*sitename***/D.***
Work files for outgoing and incoming files.

/usr/spool/uucp/LCK.*
The "lock" files UUCP uses to coordinate its resources. When a UUCP program attempts to access a remote site, it writes a "lock" file for that site. This is to prevent UUCP from accidentally attempting to access the same site more than once simultaneously. When the program that wrote the lock file exits successfully, it erases its lock files, and so makes that site accessible to other UUCP programs.

/usr/spool/uucp/.Sequence
This directory contains the sequence number of the last file handled by UUCP.

/usr/spool/uucp/*sitename***/X.***
Executed files. These files will be executed by the command **uuxqt**, and are generated by a remote system.

/usr/spool/uucppublic
Public directory accessible by all remote UUCP systems.

Attaching a Modem to Your Computer

You can use UUCP to network computers that are within the same office or the same building. It is far more common, though, to use uucp to connect computers that are far apart via modem. This tutorial assumes that you will be using uucp to exchange files via modem.

If you have not yet attached a modem to your computer, this section will give you some useful hints. It is straightforward to attach a modem to your computer, but you must be careful.

First, read the documentation that comes with your modem, and look for (1) the baud rate at which the modem operates, and (2) the command protocol that your modem uses.

Second, check the plug on the back of your modem. The modem will connect to your computer via a nine-pin or 25-pin D plug, also known as an "RS-232 interface". Such a plug can be either *male* or *female*: the male plug has nine or 25 small pins projecting from it, whereas the female does not.

Due to what can only be termed extreme stupidity, IBM AT and AT-compatible computers use RS-232D plugs for both serial and parallel ports. *Be sure to plug your modem into a serial port, not the parallel port, or you can damage your computer and your modem!*

Third, obtain a cable to connect one of the serial ports on your computer to the modem. The serial ports on an IBM AT or AT-compatible computer are almost always male. If your modem has a female plug, you will need a male-to-female cable, whereas if your modem's plug is male (which is very rare), you will need a female-to-female plug. Be sure to purchase a standard modem cable for an IBM AT; practically every computer dealer carries them. The cable you purchase should support "full modem control"; if it does not say on the package, be sure to ask your dealer before you buy it. If you are handy with a soldering iron you may be able to solder up such a cable for yourself, but unless you know precisely what you are doing it probably is not worth the trouble.

The Lexicon entry **RS-232** contains pinouts for both nine- and 25-pin connectors. When you plug in your cable, be sure to note whether you plugged it into port **com1**, **com2**, **com3** or **com4**.

Fourth, reconfigure the serial port to suit your modem. This involves the following steps:

1. Log in as the superuser **root**.

2. Edit the file **/etc/ttys**. This file normally has several lines in it, one that describes the console and one for each serial port. Each line has four fields: a one-character field that indicates whether a login prompt should be displayed (used only for devices from which people will be logging into your system); a one-character field that describes whether the device is local or remote (a local would be a modem from which you wished to dial out, a remote device would be a modem from which someone could dial in); a one-character field that describes the speed (or baud rate) at which the device operates; and a field of indefinite length that names the device being described.

If you have plugged into serial port **com1** a 9600-baud modem that will allow remote logins, edit the line for **com1** to read as follows:

```
1rPcom1r
```

If you have plugged into serial port **com2** a 2400-baud modem from which you are only going to dial out, edit the line for **com2** to read as follows:

```
0llcom2l
```

Note that the second and last character are a lower-case L, not a one. For more information, see the Lexicon entries for **ttys**.

3. Test if you have connected your modem. Turn on your modem; then log in as the superuser **root** and type the following command:

```
echo "foo" >/dev/com?l
```

where *?* is the number of the port. If you are addressing the correct port, the lights on your modem should blink briefly. For a more sophisticated test, try to communicate with your modem by using the command **ckermit**. If you are not familiar with **ckermit**, see its entry in the Lexicon for details.

4. When you have finished editing **/etc/ttys**, type the following command:

```
kill quit 1
```

This forces COHERENT to read **/etc/ttys** and set up its ports in the manner that you have configured them.

If you continue to have problems making connections with your modem, see the volume *RS-232 Made Easy*, referenced above. It describes in lavish detail how to connect all manner of devices via the RS-232 interface. also check the Lexicon articles **modem** and **RS-232** for helpful information.

Selecting Site and Domain Names

The first step to setting up UUCP is to select a site name for your system. You probably did this already when you installed COHERENT on your system, because the COHERENT mail system does not work unless you have named your system. If, however, you have not yet named your system, you can do so by editing the file **/etc/uucpname**. The name you select must have eight characters or fewer, and must be unique — or unique, at least, to the system into which you will log in. Avoid names taken from popular culture, such as “calvin,” “hobbes,” or “arnold” — these have already been used many times. See the Lexicon entry **uucpname** for more details.

Next, select a domain name for your system. Again, you probably did this when you installed COHERENT. A *domain* is a set of UUCP systems that together form one group with a common name. Even if you do not belong to a domain, you must set a domain for your system, because **mail** expects it. By convention, you can use your site name plus the suffix **.UUCP** to create a domain. The domain name is written into file **/etc/domain**. See the Lexicon entry **domain** for details.

You must edit **/etc/uucpname** and **/etc/domain** to install these names.

Set Up a UUCP Site by Hand

This chapter walks you through the setting up of a typical remote site, and explains what it’s doing (and why) at each step of the way. We hope that when you have finished reading it, you will grasp at least the principles of how UUCP works.

Setting up UUCP to call a remote system can be confusing and difficult, mainly because there are many points at which this task can fail. However, with patience and with the cooperation of the administrator of the system that you will be contacting, you can accomplish this task. Fortunately, once you have succeeded in exchanging files with another system, your connection should work indefinitely without needing modification. Fortunately, too, UUCP is designed in such a way that you can reuse system descriptions; so over time this process should become easier.

To set up UUCP so it can contact another system, you must enter information into files **/usr/lib/uucp/sys**, **/usr/lib/uucp/dial**, and **/usr/lib/uucp/port**. **uucico** reads these files to learn how to communicate with a given remote system.

The rest of this section describes how to configure your system so it can communicate with **mwcbbs**, Mark Williams Company’s bulletin board. A later section describes how to use UUCP’s commands and utilities to work with a remote system once you have established communications.

port: Describe a Serial Port

To call a given remote system, UUCP must know about the devices that it uses to communicate with that site. File **/usr/lib/uucp/port** describes ports that **uucico** can use.

Before you proceed any further, answer the following questions:

1 What serial device will be used for communications? For example, the port **/dev/com2l**.

2 At what speed will communication take place?

3 What name will you give to this port?

4 Does communication via this port involve a modem?

5 If a modem is used, what dialer entry from **/usr/lib/uucp/dial** should **uucico** use?

With the above questions answered, let's put together our **port** entry.

A **port** entry begins with the lines that names the port. Because **mwcbbs** is the system to be called for the purposes of this example, let's call this port **MWCBBS**. The port entry should look like this:

```
port MWCBBS
type modem
device /dev/com2l
baud 2400
dialer hayes
```

Note that in the **dialer** line of this entry, an underscore was used instead of a normal space character. Do not use spaces in the fields used to name the port, the actual device nor the dialer.

Note also the **type** line of this port entry. It says **modem**. The only other valid value for this line is **direct**, which you would use should you be contacting a system via a wire that runs directly from your serial port to the other system. Because the goal here is to call **mwcbbs**, you must specify **modem**.

dial: Describe a Modem

Because the port described above has a modem attached to it, you must tell **uucico** how to talk to that modem. The following example assumes that you will call **mwcbbs** via a 2400-bps, Hayes-compatible modem. If you are not familiar with sending instructions to a modem, stop here and find the manual for your modem, or find someone familiar with modem communications before continuing.

To write the **dial** entry for your modem, you must answer the following questions. Some of the questions may be unclear at first, but don't worry — we'll get into the details and get you through this:

1 What will you name this dial entry? For example, you could call your 2400-baud Hayes modem **hayes**.

2 What command tells the modem to dial out? Most Hayes-compatible modems use the command **ATDT** (for Touch-Tone telephone lines) or **ATDP** (for pulse-dial lines).

3 What message does the modem return when it connects to the remote system's modem? Most Hayes-compatible modems return the phrase **CONNECT 2400**).

4 What messages does the modem send when it fails to connect to the remote system's modem? Example include **BUSY**, **NO ANSWER**, **NO DIALTONE**, and **NO CARRIER**.

5 What command tells the modem to hang up? Most Hayes-compatible modems use the command **ATH0**.

6 How many seconds do you want to give the modem to make the connection before **uucico** times out and gives up try to connect via this modem?

Some above information is optional, but you should find all of it to write a thorough description of your modem.

An entry in **dial** begins with the line that names the modem. In the earlier example for the **port** file, we decided to use the modem named **hayes**. Therefore, the first line in our example is:

```
dialer hayes
```

Our next step is to write a chat script for the modem. At this point, a short discussion of chat scripts is in order. A *chat script* gives the dialogue (or "chat") that UUCP has with a device or a remote system. It consists of pairs of messages: an *expect* message, which is a prompt that your system expects to receive from the device or remote system, and a *response* message, which is what your system sends in response to it. Spaces separate messages from each other; therefore, you cannot use a space character within a message. Instead, use the escape sequence **\s** to represent a space character. If you want **uucico** to send a message immediately, or to send a response message without waiting for an expect message, use an empty string **""** to represent the expect string.

The next line in our example give the chat script with which **uucico** dials your modem. This is built from your answer to question 2, above. Because a Hayes modem normally does not send a prompt to request a dial command, use an empty string for the expect string. The usual command to dial a Hayes compatible modem is **ATDT** (as described above), followed by the telephone number to dial. The chat script for this dialogue is as follows:

```
chat "" atdt\D
```

Let's take a closer look at this chat script. It begins with **""**, which tells **uucico** to expect nothing from the modem. **atdt\D** is the message that **uucico** is to send to the modem. **atdt** is the dial command for a Touch-Tone telephone line, as noted above. The escape sequence **\D** represents the telephone number. We use an escape sequence instead of the literal telephone number because you can dial many different remote sites from the same modem. **uucico** finds the telephone number to dial from the entry for **mwcbbs** in the file **sys**, as will be described below.

Although this simple chat script will dial the desired telephone number, it can be improved. For example, **uucico** may fail to connect to the remote system for any number of reasons. The more information we can get back from the modem, the easier it will be to debug any problems we have in connecting to the remote system. If the modem is set up to return verbal result codes, **uucico** can check for these codes to determine if a connection to a remote modem succeeded. (For more information on *verbal result codes*, see the manual that came with your modem.) The following adds these features to our chat script for a Hayes-compatible modem:

```
chat "" ATQ0E1V1L2M1DT\D CONNECT\s2400
```

Look at this chat script carefully. Again, **uucico** expects nothing and sends a command to make the modem dial out. The command **ATQ0E1V1L2M1DT\D** has a lot of information packed into it. It is written mostly in Hayes-ese, so we'll break it up and show you what each portion means:

AT Attention: tell the modem that the following is a command.

Q0 Return result codes.

E1 Echo commands sent to it. (You can tell **uucico** to log what the modem returns, so you can see exactly what it is doing.)

V1 Use the long form of result codes.

L2 Set the loudness of the dial tone to medium.

M1 Turn on the speaker on the modem. Sometimes it is helpful to hear when the modem dials out.

DT Dial Touch-Tone.

\D The UUCP escape sequence that represents the telephone number, as described above.

Now, look at the end of this chat script: we have added a second expect message. This tells **uucico** that after it has dialed the telephone number for this site, it should expect the string **CONNECT 2400**. When **uucico** sees this message, it assumes that a modem connection was successfully established and that it can continue normally. If **uucico** does not see this message within a given period of time (as will be described below) or if it receives different message, it assumes that the connection attempt failed. When this occurs, **uucico** aborts the connection attempt.

Note that, as mentioned earlier, you must use the escape sequence **\s** to represent the space character in the phrase **CONNECT 2400**.

To review, the dial entry for dialer **hayes** now looks like this:

```
dialer hayes
chat "" ATQ0E1V1L2M1DT\D CONNECT\s2400
```

The next line, **chat-timeout**, gives the number of seconds that **uucico** should wait before it quits trying to connect to a remote system. The default is 40 seconds. You may or may not prefer to change this value; but for the purpose of this example, let's assume that you want **uucico** to wait 60 seconds before it times out. The **dial** entry for **hayes** now looks like this:

```
dialer hayes
chat "" ATQ0E1V1L2M1DT\D CONNECT\s2400
chat-timeout 60
```

Recall that our chat script for this modem turned on verbal result codes. Now, it is time to take advantage of this. Hayes compatible modems, will, in general, return any number of messages if a connection fails, depending upon the cause of the failure. The entry **chat-fail** defines a string that the modem returns when it has failed to connect. When **uucico** receives the string, it realizes that the attempt to connect has failed, and quits. The description for a modem can include any number of **chat-fail** entries. Let's add some messages that a typical Hayes-compatible modem might return when it fails to connect to a remote modem. The **dial** entry for the modem named **hayes** now looks like:

```
dialer hayes
chat "" ATQ0E1V1L2M1DT\D CONNECT\s2400
chat-timeout 60
chat-fail BUSY
chat-fail NO\sCARRIER
chat-fail NO\sANSWER
```

Modem-result codes offer many advantages. By looking for specific messages, **uucico** knows immediately when the modem cannot connect to a remote site, and quits immediately rather than waiting to time out. More important, **uucico**'s log files will show why the connection failed, which eases the debugging of connection problems.

We can add yet another safeguard to this dialer entry. By naming **abort-chat** and **complete-chat** scripts, we can tell **uucico** to chat again with the modem after a communication session has ended. These scripts can, for example, ensure that the modem hangs up the telephone and turns off error messages and echoing. If the port for this modem normally is enabled, then it is vital that you turn off echoing and error messages, to prevent the modem and your COHERENT getting into an infinite dialogue when the modem echoes COHERENT's login prompt. For details on this problem, see the Lexicon entry for **modem**. The following adds **abort-chat** and **complete-chat** scripts to our description of **hayes**:

```
dialer hayes
chat "" ATQ0E1V1L2M1DT\D CONNECT\s2400
chat-timeout 60
chat-fail BUSY
chat-fail NO\sCARRIER
chat-fail NO\sANSWER
complete-chat "" +++ OK ATH0E0V0Q1
abort-chat "" +++ OK ATH0E0V0Q1
```

Notice that the two chat scripts are identical. **uucico** runs this script whenever a session ends — regardless of whether it failed or completed successfully. As with the **chat** script, the first entry is an empty string, to indicate that **uucico** should expect nothing before it sends its first response message — in this case the string **+++**, which in Hayes-ese is the escape sequence that puts the modem into command mode. The modem replies with the message *OK*. When **uucico** sees **OK**, it sends the reply **ATH0E0V0Q1**, which is the Hayes command sequence to (1) hang up the telephone, (2) turn off echoing, (3) turn off verbose error messages, and (4) turn off error messages altogether.

Please note that the last six lines of this dialer entry are optional. The only required lines are the first two.

sys: Individual System Configuration

Having completed the **port** and **dial** entries for calling **mwcbbs**, the last — and most difficult — step is to describe **mwcbbs** to **uucico**. This is done by writing an entry in file **/usr/lib/uucp/sys**. The entry does the following:

- Names the remote system.
- Specifies valid times to call the system.
- Specifies telephone number to use when calling the system.
- Specifies valid protocols to use to exchange files with the system.
- Restricts read and write access for the remote system.

Before we continue, please note that the following example is typical of an entry in **sys**, but it is by no means exhaustive. Please refer to the Lexicon article **sys** for more information.

Now, be prepared to answer the following questions:

1 What is the name of the remote system?

2 When is it legal for **uucico** to telephone the remote system?

3 At what speed will communications take place?

4 On which port will the call to the remote system be made?

5 What telephone number, if any, must be dialed?

6 What is the “chat sequence” (chat script) to be used to log into the system?

7 What protocol should be used to transfer files?

8 How should your system identify itself to the remote system?

An entry in **sys** must begin with the line **system** *sysname*, where *sysname* is the name of the remote system. This both names the remote system and tells **uucico** that a new entry is beginning. Make this entry into **sys**:

```
system mwcbbs
```

The next line of the entry, **time**, restricts when the **uucico** may call the remote system. You can restrict calls to a given day of the week by using the following abbreviations:

Wk	Every weekday (Monday through Friday)
Su	Sunday
Mo	Monday
Tu	Tuesday
We	Wednesday
Th	Thursday
Fr	Friday
Sa	Saturday

You can restrict the time of contact; all time notation must be in military time. The following give some examples of day/time entries:

0100-0200	Valid to call every day between 1 and 2 AM
Mo0100-0200	Valid to call every Monday between 1 and 2 AM
Sa	Valid to call at any time on every Saturday
Any	Valid to call at any time
Never	Never call the system

To restrict the valid times for calling **mwcbbs** to Saturday and Sunday nights from 10 to 11:15 PM, add the line:

```
time SaSu2200-2315
```

Our **sys** entry now looks like this:

```
system mwcbbs
time SaSu2200-2315
```

The next line of the **sys** entry, **baud**, is self-explanatory and is answered by the third question above. For this example, we will assume a speed of 2400 bps. The **sys** entry now look like:

```
system mwcbbs
time SaSu2200-2315
baud 2400
```

The next line, **port**, names the port that **uucico** is to use to telephone the remote system. This must name a port that is defined in file **port**; in this case we'll use port **MWCBBS**, which we wrote earlier:

```
port MWCBBS
```

The next line, **phone**, is answered by question 5, above; this, too, is self-explanatory. Add the following line to our **sys** entry:

```
phone 17085590412
```

uucico uses this telephone number to expand the escape sequence **\D**, used in the **dial** entry defined above. Note, by the way, that you can access **mwcbbs** through the following three telephone numbers:

1200/2400-baud generic modem:	708-559-0412
9600-baud Trailblazer modem:	708-559-0445
9600-baud V.32 or HST modem:	708-559-0452

This example assumes that you have a generic 2400-baud modem; but you should select the number that best suits your modem.

The next line, **chat**, is the chat script that **uucico** uses to log into the remote system. Please refer to the previous section about the **dial** file for a brief description of what a chat script is and how it is laid out; or see the Lexicon entry for **sys**:

```
chat "" \n in:--in: nuucp word: public word: serialnum
```

Please note that the string *serialnum* represents the serial number of your COHERENT system. You must use this number to log into system **mwcbbs**.

Let's take a moment to review this line. By the time **uucico** needs the information from this line, it has already successfully dialed into and connected to one of the modems on **mwcbbs**. Now it must log into the system. The chat script tells **uucico** to expect nothing and to immediately send out a newline, as represented by the escape sequence **\n**.

After **uucico** sends the newline character, it waits for **in:**, i.e., the final characters in the **mwcbbs** login prompt **mwcbbs 386 login:**. It is not necessary to expect the entire login prompt, just enough of it to let **uucico** know that it can send the next message — in this case, your system's login identifier. If **uucico** does not receive **in:**, it sends another newline character and again waits for **in:**. (**--in:** is equivilant to **-\n-in:**).

Once it receives **in:**, **uucico** sends **nuucp**, which is the login name that **mwcbbs** expects to receive from your system. **uucico** now expects the message **word:** which is the is the tail of the prompt **mwcbbs** really transmits, **password:**. When **uucico** sees **word:**, it sends **public**, which is the password that **mwcbbs** expects. Once **mwcbbs** receives the password it expects, it sends out a prompt for a remote-access password — hence, this chat script again expects to see **word:** again. When it receives this second password prompt, this chat script tells **uucico** to send your system's serial number. This completes the chat script.

The **sys** entry for **mwcbbs** now looks like this:

```
system mwcbbs
time SaSu2200-2315
baud 2400
port MWCBBS
phone 17085590412
chat "" \n in:--in: nuucp word: public word: serialnum
```

Now, we will add the **protocol** line to this entry. This tells **uucico** which protocol to use when exchanging files with the remote system. Different implementations of UUCP use various protocols for exchanging files. The **g** protocol was the first UUCP protocol to be invented, and is still the most commonly used. (By the way, this protocol is named after its designer, Greg Chesson.) Since then, other protocols have been invented. Each protocol has its strengths and weaknesses; you should weigh carefully how you are communicating with the remote system and what protocols the remote system, supports before you select a protocol. Please refer to the Lexicon article **sys** for a complete list of the protocols that the Taylor UUCP package recognizes, and for information on when each should be used.

Please note, too, that the **g** protocol is not implemented uniformly by all versions of UUCP. For instance, some **g** protocols can only send data in packets of 64 bytes, using three *sliding windows*. Other implementations can support up to 4,096 bytes per packet, using seven sliding windows. The more windows supported, the greater that transfer rate of data. The larger the packet size per window, the greater the transfer rate of data. For a detailed discussion of the internals of UUCP protocols, please refer to third-party publications or to documentation for the Taylor UUCP system in the file **/usr/src/alien/uudoc.tar.Z**. Taylor UUCP can be configured from as little as three windows and 64 bytes per packet, to seven windows and 4,096 bytes per packet. You can set these parameters in the **sys** file; for details, see the Lexicon entry for **sys**. In this example, we will use the default values of seven windows and 64-byte packets.

After all of this discussion, add the following line to the **sys** entry we are building:

```
protocol g
```

For the next step, you must select the name by which your system identifies itself to **mwcbbs**, asked in question 8, above. "But wait!" you say. "Didn't I answer this in the chat script that we labored over just earlier? We identified ourselves as 'nuucp'." Well, not exactly, as we will make clear.

The login name of **nuucp** is not the same as the site name with which we wish to identify ourselves. Once our system has logged into the remote system (in this case, **mwcbbs**), the remote system fires up **uucico** on its end. Once again our system must identify itself — but this time to **uucico**, not to the login program.

Think of the login sequence this way: as a normal user must log in giving his name and a password or two just to run a shell, so to must **uucico** log into **mwcbbs**. It logs in with the name **nuucp** and gives the necessary passwords; however, **mwcbbs**, instead of invoking a shell for user **nuucp**, invokes **uucico** instead. (If you look in the file **/etc/passwd**, you'll see that the last entry for each user is the program that the system runs for that user; usually it is a shell, but sometimes it's another program, e.g., **uucico**). Now, **uucico** on **mwcbbs** begins to talk with **uucico** on your system. The real work of transfering files can begin, as long as **mwcbbs** recognizes the sitename with which your system identifies itself.

mwcbbs does not recognize many remote systems, yet it receives calls from more than 100 different systems per day. How does **mwcbbs** handle all of these calls if it really only knows a few dozen systems? Most systems identify themselves by the same name, **bbsuser**. **mwcbbs** is set up to always recognize the remote site **bbsuser**.

Where does this name come from? In normal circumstances, this name is read from the file **/etc/uucpname**. This file holds the name you gave your system when you installed COHERENT; you were required to select a name because electronic mail will not work without it. (For more information on this file, see its entry in the Lexicon.) The odds are that you did not choose the name **bbsuser**. Let's say that when you installed COHERENT onto your system, you chose the name **foobar**. Now, when you system calls **mwcbbs**, by default it will identify itself to **mwcbbs**'s edition of **uucico** as **foobar**. However, **mwcbbs** doesn't know **foobar** from Adam, so it logs your system off and hangs up the telephone.

To get around this, you must insert the line **myname** into the **sys** entry for **mwcbbs**. Add this line to the sys entry we are building:

```
myname bbsuser
```

Now, whenever your system calls **mwcbbs**, it will identify itself to **mwcbbs**'s **uucico** as **bbsuser**.

At this point, we could fill dozens of pages with discussions of the items you can configure in **sys**. However, we now have all of the information we need to call **mwcbbs**. For a fuller discussion of **sys**, look up its entry in the Lexicon.

For now, the completed **sys** entry for **mwcbbs** is as follows:

```
system mwcbbs
time SaSu2200-2315
baud 2400
port MWCBBS
phone 17085590412
chat "" \n in:--in: nuucp word: public word: serialnum
protocol g
myname bbsuser
```

There are a few other items that you will find yourself configuring as part of a typical entry in **sys**

First, you must indicate whether the remote system and request files from your system, and transfer files into it. You may wish to deny this capacity to some remote systems, but you do want to grant it to **mwcbbs**, as the whole point of access that system is to have it download files to you. So, add the following two lines to the bottom of the entry for **mwcbbs**:

```
request yes
transfer yes
```

Next, you must name the directories on your system from which the remote system can request files, and the directories on the remote system onto which your system has permission to write files. By default, remote systems are limited to requesting files from directory **/usr/spool/uucppublic** and its subdirectories. To change this default, add the instructions **remote-send** and **remote-receive** to the entry in **sys** for the remote site. For example, suppose you decided to let **mwcbbs** read the directory **/usr/private**, but not **/usr/private/myfiles**. You also decided to let **mwcbbs** write files into **/tmp**, but not into directory **/tmp/secret**. To do this, add the following instructions to the **sys** entry for **mwcbbs**:

```
remote-send /usr/private !/usr/private/myfiles
remote-receive /tmp !/tmp/secret
```

Naming a directory in **remote-send** or **remote-receive** lets the remote system, respectively, read from that directory and all of its sub-directories. However, if you prefix a directory name with an exclamation point '!', that directory and its subdirectories are specifically excluded from being accessed by the remote system. The following gives the normal settings for these directories:

```
remote-send /usr/spool/uucppublic /tmp
remote-receive /usr/spool/uucppublic /tmp
```

Next, you must name the directories from which your system can send files to the remote system, and into which your system can write the files that you have requested from the remote system. These are named by, respectively, the instructions **local-send** and **local-receive** The following gives the normal settings for these directories:

```
local-send /usr/spool/uucppublic /tmp
local-receive /usr/spool/uucppublic /tmp
```

If you are confused about how these instructions differ from the instructions **remote-send** and **remote-receive**, just remember that the **remote** instructions name directories for send/receive requests initiated by the remote system; where as the **local** instructions name directories for send/receive requests initiated by your system.

One last entry needs to go into this file: you need to name the commands that **mwcbbs** can execute on your system. It needs to execute at least the commands **rmail** and **uucp** so that it can send you mail and upload files to your system. So, add the following line to the end of the entry for **mwcbbs**:

```
commands rmail uucp
```

With these lines, our **sys** entry for **mwcbbs** is complete. It looks like this:

```
system mwcbbs
time SaSu2200-2315
baud 2400
port MWCBBS
phone 17085590412
chat "" \n in:--in: nuucp word: public word: serialnum
protocol g
myname bbsuser
request yes
transfer yes
remote-send /usr/spool/uucppublic /tmp
remote-receive /usr/spool/uucppublic /tmp
commands rmail uucp
```

This is a typical **sys** entry for calling a remote system.

Simplifying a UUCP Configuration With uuinstall

The program **/usr/bin/uuinstall** has been included to help build, modify, or delete entries in the files **sys**, **port**, and **dial**, making configuring UUCP easier. **uuinstall** uses a system of screens and windows to gather and organize the information UUCP needs to work with a remote system. While **uuinstall** does not remove all of the pain from setting up communications with a remote site, it does make this (admittedly complex) task easier.

This section shows how **uuinstall** can simplify the process of configuring UUCP to communicate with **mwcbbs**.

If you have not read the previous section of this chapter, regarding configuring UUCP to call **mwcbbs**, *please do so now. That section discusses in detail many topics that will not be repeated here. In particular, look at the questions asked in the previous section regarding configuring the files* **port, dial**, *and* **sys**, *and be prepared to answer them in this section.*

Invoking uuinstall

To invoke **uuinstall**, just type **uuinstall** at the shell prompt.

For security reasons, only the superuser **root** and user **uucp** can invoke this program. If other users could access this program, they would have access to the information necessary to log into the remote systems listed in **sys**. Only priviledged users should have access to this information.

When you invoke **uuinstall**, the following screen appears:

```
                 UUCP Configuration: Main menu



      <s>ys file      Configuration information for specific systems
      <p>ort file     Information for individual ports
      <d>ial file     Configuration information for dialing modems



       Press the letter corresponding to the file you wish to examine
                       or <q> to quit
```

The Port File

The menu says it all: choose the file to work with. For the sake of simplicity, we will modify the system files in the same order that we did in the extended example that appeared in the previous section; so, press **p** to select the file **port**.

When you make your selection, **uuinstall** displays its **action menu**:

```
                            Action menu

            You have selected to work with the port file.

                       Do you wish to:
                               <a>dd an entry
                               <d>elete an existing entry
                               <m>odify an existing entry
                               <v>iew an existing entry

       Press the letter corresponding to the action you wish to perform
                   or press <RETURN> for the main menu.
```

Press **a** to add an entry. The following screen appears:

```
                Port File Entry Screen


               port   [_              ]
               type   [     ]
               device [/dev/              ]
               baud   [     ]
               dialer [               ]


       Enter the name that you want associated with
          the device that this entry will define.
                Enter nothing to cancel.
```

It is time to enter our information. To do so, type the appropriate information into each field on the screen. When you have entered all of the information required by that field, press `(¢);` the cursor drops to the next field on the screen.

When the cursor enters selected fields, **uuinstall** displays a help message to describe the information that you must enter. The message will, in many cases, give an example of valid data.

If you wish not to enter anything into a given field, type `(¢)` when the cursor enters that field. Some fields are *required*: that is, the system demands that you type something into the field; in this case, the cursor will not move until you have entered something. After you have completed the last line, **uuinstall** prompts you to ask if it should add the new entry to file **port**.

From here, configuring a **port** entry is simple. Look at the questions that we asked in the previous section about the information that a **port** entry needs. Earlier, we decided to name this port **MWCBBS**; therefore, type **MWCBBS** into the first field on the screen — the one labelled **port**. Press `(¢);` the cursor jumps to the next field, which is labelled **type**.

The port can be either of two types: **direct** or **modem**. Since we're using a modem to telephone **mwcbbs**, type **m** into the field labelled **type**. You would type **d** (for *direct*) should the remote system be connected to yours via a line that runs directly from your serial port to the remote system. The cursor jumps to the next field, which is labelled **device**, and positions itself just to the right of the string **/dev/**.

Now, type the name of the device associated with this port. Earlier, we chose **/dev/com2l**. Because **/dev/** is already in place, just type **com2l**. Again, press `(¢)`; the cursor jumps to the field labelled **baud**.

Type the speed that communications will take place at: in this example, **2400**, then press `(¢)`. The cursor jumps to the field labelled **dialer**.

Finally, type the name of the dialer script that **uucico** is to use to talk with the modem plugged into this port. As you recall, our example uses the script named **hayes**; so type that into this field and press `(¢)`.

The completed entry, as we typed it here, looks like this:

```
        Port File Entry Screen


 Do you wish to write this entry? (y/n)_

         port   [MWCBBS]
         type   [modem]
         device [/dev/com2l]
         baud   [2400]
         dialer [hayes     ]
```

uuinstall now asks if you wish to write this entry into file **/usr/lib/uucp/port**. Type **y**, and press `(¢)`. **uuinstall** saves the information and returns to its main menu.

The Dial File

From the main menu, press **d** to select the file **dial**. This will take us to the action menu. Again, select **a** to add an entry. **uuinstall** then displays the following screen:

```
                  Dial File Entry Screen

 dialer:        [_            ]
 chat:          [                                            ]
 chat-timeout:  [   ]
 chat-fail:     [                         ]
 chat-fail:     [                         ]
 chat-fail:     [                         ]
 complete-chat: [                                            ]
 abort-chat:    [                                            ]


 Enter the name of the dialer that
 this entry describes.
 Leaving this field blank aborts entry.
```

Begin by typing the name of the dialer script. In the previous example, we named it **hayes**. Type this in the first line; then press `(¢)`. The cursor jumps to the next field, which is labelled **chat**.

Now, type the chat script that **uucico** will use to dial the modem. Please refer to the previous section for a discussion of what a chat script is, and of the elements that the dialer chat should contain. In the previous section, we decided to use the following script:

```
"" ATE0Q1V1L1M0DT\D CONNECT\s2400
```

Type this, then press `(¢)`. The cursor jumps to the next field, which is labelled **chat-timeout**.

Type the number of seconds that **uucico** should wait before it aborts its attempt to dial out on the modem. Press `(¢)`; the cursor jumps to first field labelled **chat-fail**.

Into the next three fields, enter messages that the modem might return when an attempt to connect to a remote system fails. In the previous section, we selected the messages **NO\sCARRIER**, **BUSY**, and **NO\sDIALTONE**. Type the first message, then press (¢). The cursor jumps to the second **chat-fail** field; type the second message and press (¢) again. The cursor jumps to the third **chat-fail** field; type the third message and press (¢) once again. The cursor jumps to the field labelled **complete-chat**.

Note that **uuinstall** gives you space to enter three **chat-fail** messages. You can, however, enter an indefinite number of these messages into a dialer script. If you wish to enter more than three **chat-fail** messages, you must edit the file **/usr/lib/uucp/dial** by hand.

The next two fields, respectively labelled **complete-chat** and **abort-chat**, let you enter the chat scripts that **uucico** should execute when, respectively, a call completes successfully or fails for some reason. As we noted in the previous section, these scripts are optional; however, you are well advised to enter them, to ensure that the modem is returned to its correct state after a call is completed. In the previous section, we devised the following script for our Hayes-compatible modem:

```
"" +++ OK ATH0E0V0Q1
```

Type that script into the field labelled **complete-chat**, then press (¢). Type again type it into the field labelled **abort-chat**, and press (¢) again.

The completed dial entry screen should now look like this:

```
           Dial File Entry Screen

dialer:        [hayes     ]
chat:          ["" ATE0Q1V1L1M0DT\D CONNECT\s2400]
chat-timeout:  [60]
chat-fail:     [BUSY]
chat-fail:     [NO\sDIALTONE]
chat-fail:     [NO\sCARRIER]
complete-chat: ["" +++ OK ATH0E0V0Q1]
abort-chat:    ["" +++ OK ATH0E0V0Q1]


     Do you wish to save this entry? (y/n)
```

If you are comfortable with your entry as it is, then press **y** to write it into **/usr/lib/uucp/dial**. If not, press **n**. In either case, **uuinstall** returns to its main menu.

The sys File

It is now time to describe a remote system, in this case **mwcbbs**, to this system. From the main menu, press **s** to select the **sys** file, then select **a** from the action menu to add an entry to the sys file. The following screen will appear:

```
                       Sys File Entry Screen


      system:     [_        ]
      time:       [                                    ]
      speed:      [     ]
      port:       [            ]
      phone:      [                                    ]
      chat:       [                                              ]
      myname:     [        ]
      protocol:   [   ]
      commands:   [                                              ]
      read-path:  [                                              ]
      write-path: [                                              ]


      Enter the name of a remote uucp system. You should
      limit the name to 8 characters.

      Leaving this field blank aborts entry.
```

Because we are using the remote system **mwcbbs** as our example, type **mwcbbs** into the field labelled **system**, then press `<¢>`; the cursor jumps to the field labelled **time**.

For this field, let's get a little creative. Let's limit calling **mwcbbs** to after 6 PM and before 6 AM on weekdays, but permit any time on weekends. Type the following line in the **time** field:

```
Sa,Su,Wk1800-0600
```

Press `<¢>`, which moves the cursor to the field labelled **speed**.

Now enter the speed or ("baud rate") at which communications will occur. Earlier, we selected 2400 bps; so type **2400** and then `<¢>`.

The cursor is now in the field labelled **port**. Type the name of the port via which we will call **mwcbbs**. As noted above, we will be using the port that we named **MWCBBS**; so type that into this field and then press `<¢>`.

The cursor is now in the field labelled **phone**, which holds the telephone number for the remote system. To find the telephone number for **mwcbbs**, check the release notes that came with your copy of COHERENT; then type it into this field and press `<¢>`. The cursor jumps into the field labelled **chat**, for the chat script.

We discussed the chat script in some detail in the previous section. Enter the following chat script in this field:

`"" \n in:--in: nuucp word: public word:` *serialno*

Remember to replace *serialno* with the serial number provided with your COHERENT package. When you have finished typing the chat script, press `<¢>`. this moves the cursor to the field labelled **myname**.

Because **mwcbbs** does not grant access to just any system, your system must identify itself as a system that **mwcbbs** already knows about. **mwcbbs** grants access to every system that identifies itself as **bbsuser**; so type **bbsuser** into this field, and press `<¢>`. The cursor moves into the field labelled **protocol**.

This field holds the protocol with which your system will exchange files with **mwcbbs**. The Taylor UUCP package supports several protocols. **mwcbbs** in turn recognizes protocols **a**, **g**, and **i**. Please refer to the Lexicon article **sys** for more information on available protocols, and the strengths and weaknesses of each. For the purposes of this example, type **g** (for the **g** protocol), then press `<¢>`.

The cursor is now in the field labelled **commands**, which lists the commands that the remote system may execute on your system. Because **mwcbbs** will not be calling you, just press `<¢>`. **uuinstall** will write a default list of commands into this field, then move the cursor to the next field.

The last two fields, which respectively are labelled **read-path** and **write-path**, limit the directories that the remote system can, respectively, write into or read from. They point to the **remote-send** and **remote-receive** instructions within a **sys** entry. Press `<¢>` for each field; **uuinstall** will write into each field the default directory, which is **/usr/spool/uucppublic**.

With the template completed, your entry should look like this:

```
                         Sys File Entry Screen


      system:     [mwcbbs]
      time:       [Sa,Su,Wk1800-0600]
      speed:      [2400]
      port:       [MWCBBS]
      phone:      [17085590412]
      chat:       ["" \n in:--in: nuucp word: public word: serialno]
      myname:     [bbsuser]
      protocol:   [g]
      commands:   [rmail rnews uucp uux]
      read-path:  [/usr/spool/uucppublic]
      write-path: [/usr/spool/uucppublic]


               Do you wish to save this entry? (y/n)
```

If you are comfortable with the information you entered, press **y** to have it written to the file **/usr/lib/uucp/sys**.

This concludes our examples of configuring UUCP to call remote systems. Due to the extreme flexibility of the Taylor UUCP package, it is not feasible to review all possible configurations available to you. Each of the configuration files, **/usr/lib/uucp/sys**, **/usr/lib/uucp/port**, and **/usr/lib/uucp/dial**, can include more commands than are reviewed here. Many of these are reviewed in the Lexicon entries for each of the files. A complete set of ASCII text Taylor UUCP documentation, as provided with the distribution available from several internet sites, is included in the file **/usr/src/alien/uudoc104.tar.Z**. The complete source code for the Taylor UUCP package as distributed with COHERENT is available from **mwcbbs**.

Modifying an Existing Entry

The above examples show how to use **uuinstall** to enter a new entry into files **port**, **dial**, and **sys**. You can also use **uuinstall** to delete, view, or modify existing entries in these files. Of these, the most useful is the feature for modifying an existing entry.

Let's say that we want to modify our entry for dialer **MWCBBS** to include a fourth **chat-fail** instruction, for the modem message **ERROR**. To do so, do the following:

- Invoke **uuinstall** from the shell, as described above.
- When **uuinstall** displays its main menu, type **d**, to edit the file **/usr/lib/uucp/dialer**.
- When **uuinstall** displays its action screen, type **m**, to modify this file.
- **uuinstall** then displays the names of all of the entries in this file. Use the arrow keys on your terminal to move the cursor to the entry that you wish to enter, in this case **MWCBBS**; then press `(¢)`.
- **uuinstall** extracts the entry for **MWCBBS** from **/usr/lib/uucp/dial**, writes it into a temporary file, then invokes the MicroEMACS editor for that temporary file. You can use the usual MicroEMACS commands to edit this entry. In this case, add the line

  ```
  chat-fail ERROR
  ```

  into the entry.
- When you have finished editing the entry, type **<ctrl-Z>**.
- **uuinstall** will prompt you and ask if you wish to save your changes. Type **y** if you do, **n** if you do not.

Note that if you do save your changes, you can always use **uuinstall** to remodify the entry.

Configuring UUCP for Dial-in Access

The above examples show how to configure your UUCP system so that it can dial out to another remote system. Configuring UUCP so that other systems can dial into your system is, for the most part, like configuring it to dial out; but this task does present some special problems that you must consider. The following example shows how to set up UUCP so that remote system **dalek** can dial into your system.

Giving a Remote UUCP Site a Login

At this point, you are now the systems administrator of your COHERENT system who must tell someone else how to set up her UUCP to log into your system. We've shown you the flip side of this by showing you how to access **mwcbbs**: now the job is yours.

When a UUCP site calls your system, it must log in as would any ordinary user would. Once it has logged in, however, it runs the command **/usr/lib/uucp/uucico** rather than a shell, which a normal user would run. This portion of what you must set up is configured in the file **/etc/passwd**.

You can create a UUCP login by running the command **newusr**; then edit the last field of the **/etc/passwd** entry for the login you just established to run the command **/usr/lib/uucp/uucico** instead of a **/bin/sh** or **/usr/bin/ksh**.

You could also create a UUCP login by manually editing **/etc/passwd** and copy the entry for user **uucp**, but change the user name of **uucp** to something else.

You must also define the home directory if using **newusr**. Because this is a UUCP account, the home directory appears under the directory **/usr/spool/uucp**. For example, if you wanted site **dalek** to call you, you might establish an **/etc/passwd** entry that looks like:

```
dalek:password:6:6:Coherent-Coherent \
       copy:/usr/spool/uucp:/usr/lib/uucp/uucico
```

Please note that *password* in the entry for **dalek** represents the encrypted password you assigned to site **dalek**. Give the password to the system administrator of site **dalek** so that she may properly configure her chat script to log into your system.

If we were to stop right here, **dalek** could call your system, log in, and begin a UUCP session. Unfortunately, since we've yet to configure the UUCP files themselves to know about **dalek**, your site would quickly terminate the call when **dalek** identified itself to your system after completing its chat script.

Configuring a Spooling Directory for Remote UUCP Access

Each UUCP site that calls your system must have a *spooling directory* in **/usr/spool/uucp**. While logged in as **root**, go to the directory **/usr/spool/uucp** and run the command:

```
/usr/lib/uucp/uumkdir dalek
```

Configuring UUCP Files

To control what **dalek** does on your system, you should describe it in file **/usr/lib/uucp/sys**. COHERENT includes a dummy entry in this file that you can easily modify for site **dalek**. You should make an entry that looks like this:

```
system dalek
time Never
commands rmail rnews uucp uux
remote-send /usr/spool/uucppublic !/usr/spool/uucppublic/bobfiles
remote-receive /usr/spool/uucppublic !/usr/spool/uucppublic/private
```

This entry names **dalek** as a system UUCP recognizes. Your system will never call **dalek**, because the **time** command (which gives the legal dates and times during which the remote system can be contacted) states **Never**.

The **commands** instruction names the commands that you permit **dalek** to execute on your system. This line overrides any permissions that may be available to **dalek** on the system level: that is, this line can stop **dalek** from executing commands that it otherwise would have permission to execute (e.g., **ls** or **cat**), but it cannot enable **dalek** to execute commands that it normally would not be permitted to run (e.g., **su**, **shutdown**, or **reboot**).

The lines **remote-send** and **remote-receive** name, respectively, the directories whose contents **dalek** can read, and the directories into which it can write files. Once again, these lines can stop **dalek** from accessing directories that it otherwise would be allowed to access (e.g., **tmp**), but they cannot give **dalek** permission to manipulate directories that it normally would not be able to access (e.g., **bin**).

In brief, all you have done is identified **dalek** to your system, named the commands it can execute on your system, and limited the directories it can access. It's that easy!

One Last, Loose Thread

With the spooling directory created, we are almost done. Run this command:

```
/usr/lib/uucp/uutouch dalek
```

It will place a dummy command in **dalek**'s spooling directory. More important, it returns an error if it finds some errors in the UUCP configuration for **dalek**.

Unfortunately, we cannot give you a test system that will call your system to test your UUCP configuration. You will have to use this section as a guide to configure for another UUCP site to call yours.

Requesting Files From a Remote UUCP System

To request a file from a remote UUCP system, you must know where that file is on the remote system. The file **howto.start** can be found in the directory **/usr/spool/uucppublic/mwcnews** on **mwcbbs**. This file introduces **mwcbbs**, its features and intended uses, and how to request files from it.

With this bit of knowledge, we can now request the file with the command **uucp**.

uucp is very simple. Invoked it with a specific site to call, and file to upload or download. For example, the command:

```
uucp mwcbbs!/usr/spool/uucppublic/mwcnews/howto.start /tmp
```

tells your machine to call **mwcbbs**, download the file

```
/usr/spool/uucppublic/mwcnews/howto.start
```

and put it into directory **/tmp** on your system. The call will take place seconds after you enter the command, unless you tell **uucp** to spool the request. For more information on this and other arguments, see the Lexicon entry for **uucp**.

Please note that the entry for **mwcbbs** in **/usr/lib/uucp/sys** must specify that **mwcbbs** can write to **/tmp** as part of the **remote-receive** instruction.

To send a file to **mwcbbs**, use the command:

uucp *filename* mwcbbs!/usr/spool/uucppublic/uploads/

This command uploads a copy of *filename* to the directory **/usr/spool/uucppublic/uploads** on **mwcbbs**. Again, the call takes place within seconds, unless you tell **uucp** to spool the request.

At this point, we have completed our **uucp** configuration to "talk" to **mwcbbs**, and we have requested our first file. You can tell **uucp** to download other files from **mwcbbs**; only the file names and path names will change.

Sending Files to a Remote UUCP System

Suppose, for example that site **santa** has been described to your UUCP system, and everyone has permission to read from your current directory. Suppose, too, that you have permission to write into directory **/usr/spool/reports/parents**. To send the files **good.kids** and **bad.kids** to **santa**, type the following command:

```
uucp good.kids bad.kids santa!/usr/spool/reports/parents
```

The **uucp** command compels UUCP to copy one or more files from your site to a remote site. UUCP queues both files automatically and sends them at the next scheduled time.

Note, too, the use of the '!' in the above command. The '!' separates a site name from another site name, from a directory name, or from a user ID. In the above example, the '!' indicates that directory **/usr/spool/reports/parents** can be found at site **santa**. One feature of a UUCP network is that any member can send files to any other member. That does not mean that every member must have full permissions with every other member; rather, for the sake of efficiency it is possible to route files through one or more intermediate computers, to allow batch transmissions of files. For example, to send the file **visibility** to user **blitzen** via machines **santa** and **reindeer**, use the following command:

```
uucp visibility santa!reindeer!blitzen!/usr/spool/weather/usa
```

In this example, the string **santa!reindeer!blitzen!/usr/spool/weather** indicates that directory **/usr/spool/weather** can be contacted at site **blitzen**, which in turn can be contacted via site **reindeer**, which in turn can be contacted via site **santa**. This scenario assumes that site **reindeer** has permission to write into directory **/usr/spool/weather** on site **blitzen**, and that site **santa** has permission to upload files to site **reindeer**. (And, of course, that you have permission to upload files to site **santa**.) If any of these are not true, the transaction will fail.

UUCP Administration

Once you have written and debugged the descriptions of your ports, dialers, and systems, administering UUCP consists mainly of reviewing the log files periodically to ensure that all connections are being made, and all programs executed correctly. The command **uulog** will assist you in this. When you type the command

```
uulog widget
```

uulog will open all of the log files associated with site **widget**, and display them for you. Given that the log files for given site are kept in four different directories, this can be a great convenience.

Logfiles are organized as follows:

/usr/spool/uucp/.Log/uucico/*sitename*
/usr/spool/uucp/.Log/uucp/*sitename*
/usr/spool/uucp/.Log/uux/*sitename*
/usr/spool/uucp/.Log/uuxqt/*sitename*

As you can see, one logfile for each site is kept in a directory named after a given UUCP command. UUCP records every transaction; so by reading these files, you can see whether your UUCP commands are succeeding.

If you are having trouble with your UUCP connections, send files through UUCP and observe how they fail. You may need to use **uuinstall** a few times to tweak your description of the remote site. If all else fails, contact Mark Williams Company.

If all is going well, you should run **/usr/lib/uucp/uumvlog** every day. This keeps the log files from getting out of hand. The previous section on setting the polling time describes how to do this.

The main task of the UUCP administrator is to monitor the UUCP log files to see that hardware is functioning correctly, and that files are transferred correctly. For example, failure to connect with a remote site after several attempts may mean that the remote site is having problems with its modem, or that it is scheduling outgoing calls for when you were scheduled to call in. Likewise, failure to receive scheduled calls from several sites may indicate equipment failure on your end.

Finally, the UUCP administrator must monitor the use of disk space on the system. Old mail and messages, multiple copies of files, and files automatically input by various subscription and network services can eat up disk space rapidly; you must prune extraneous material ruthlessly.

Networks

UUCP becomes truly useful when you are hooked into a network of machines that exchange information. Through UUCP, you can gain access to the Internet, through which you can exchange news and mail with others users around the world.

This section briefly describes the services you can obtain from a network, and the networks now available.

Services

Many different services are available from networks: domain-name service (DNS), mail exchanger service, and connectivity.

DNS is the registration of an Internet-style domain, e.g., **mwc.com** or **baqaqi.chi.il.us**. This usually divides into two subservices — registration in an existing domain, and registration of your own domain. For this service you need two sites on the Internet willing to either register you in their nameserver or run a nameserver for your domain. A *nameserver* gives other people's machines information about your registered machine or registered domain. In particular, the nameserver publishes an MX record, which tells machines how to get mail to you. (There are other kinds of records, but MX records are the important ones for UUCP sites.)

MX service is mail forwarding. The MX record published by the nameserver must point at a machine directly on the Internet. This machine will the be responsible for figuring out how to actually get the mail to you.

Connectivity usually means a UUCP connection. Because almost everybody in a major city is connected to everyone else, a UUCP connection to anybody effectively translates to an indirect connection to the Internet.

Available Networks

UUNET is a company in Falls Church, Virginia, that provides a large number of networking services, including all of those mentioned above.

The **UUCP network** is an extremely informal group of machines defined only by the fact that they connect to each other via the UUCP protocol. It is one of the largest networks in the world and has no central control.

UseNet is a network of machines defined by the fact that they exchange UseNet news with each other. UseNet is also an anarchy — no central organization runs it. It includes machines in the UUCP network and the Internet, as well as hundreds of other networks. It is the largest network in the world.

The **Internet** is a group of high-speed networks which all communicate with *Internet Protocol*, i.e., TCP/IP. The networks that comprise the Internet are mostly academic and research networks run by large central organizations, such as the National Science Foundation or the Australian Academic Research Project. The center of the Internet is the NIC (Network Information Center), whose address is **nic.ddn.mil**.

The **internet** (lower-case 'i') is defined by connectivity under mail. It is technically larger than UseNet, though less is said about it because it is so weakly defined.

For information on networks, what is available on them, and how to connect to them, we strongly recommend the book *The Whole Internet: User's Guide and Catalog*, cited above. For a copy, check your local bookstore, or telephone the publisher, O'Reilly & Company, at 1-800-998-9938.

Debugging UUCP Problems

When you have a problem with UUCP — and in particular, a problem with telephoning another UUCP system — you must have a clear picture of what is occurring, and what is not occurring. For instance, if you try to call **mwcbbs** and UUCP fails, you must determine what is working properly before the failure takes place.

The following subsections describe common problems with UUCP, and gives some hints on how to solve them. Please review them *carefully* before you telephone Mark Williams Company to ask for help. If you do not, we will ask that you do review them and call back only if you still cannot solve the problem.

Define the Problem Exactly

UUCP problems can take many forms. Define the problem *exactly*. This process may actually help you solve the problem. Statements like "I'm having a problem using UUCP" or "UUCP doesn't work" do not describe problems relating to UUCP. You need to know *exactly* what does or does not happen when you try to connect with another site. Please review it before you call Mark Williams Technical Support.

Before you do anything else, trying running **uucp** and **uucico** with the option **-x**. This option tells these programs to log what they do; the logs are written into the subdirectories of directory **/usr/spool/uucp/.Admin**. Often, these log messages will point directly to the problem.

A subtle error within a UUCP configuration file can cause no end of grief when you try to debug a UUCP problem. To help spot these potential problems, run the command **/usr/lib/uucp/uuchk**. This command generates a full report on your configuration files. It will spot and report on syntax errors in your configuration files. You will, of course, have to fix by hand whatever **uuchk** finds to be in error.

The following sections discuss commonly encountered problems.

Enabling and Disabling Ports

On some COHERENT systems, the permissions for the programs **/etc/enable** and **/etc/disable** are set to:

```
-r-x------   root     root
```

That is, these commands can be executed only by user **root**. This is to close a security hole; otherwise, a person who breaks into your system can disable any port she wants — including the console.

If you have only one modem, and you both initiate and receive UUCP sessions on that modem, you may run into problems with enabling and disabling that port. For the communications program **uucico** to be able to enable and disable the port on its own, **/etc/enable** and **/etc/disable** must have the permissions:

```
-r-s--s--x   root     root
```

These permissions permit **uucico** to dial out on an enabled port on its own; but it reopens the security hole. *Caveat utilitor.*

For information on how to change permissions, see the Lexicon entry for the command **chmod**.

Stale Requests and Multiple Requests

From time to time, you may accidentally issue the same **uucp** request more than once.

Note that if **uucp** fails because you failed to connect with the remote site, one action you should *not* take is to repeat the **uucp** command. If you do this, **uucp** will simply queue another request for the same file or files that you requested with the previous **uucp** command. When connection is finally made, multiple **uucp** requests will be executed, thus downloading multiple copies of the same file or files. Depending upon the size of the file or files, this could be an expensive mistake.

To remove stale requests, log in as user **uucp** or user **root**; then **cd** to directory **/usr/spool/uucp/***sitename* and remove the extraneous requests. (You can also do this to remove mail files about which you have had second thoughts.)

Note that a **uucp** generates one file, which has the prefix **C**. A mail message generates two files: one with the prefix **C**, which tells the remote system what to do with the mail message; the other has the prefix **D**, which holds the text of the message. Read the existing files to make sure that you are removing the correct files.

Problems With Lock Files

If **uucico** wishes to dial on a modem but somebody else is already using it, you will see the message

```
ERROR: All matching ports in use
```

in the log file for the site **uucico** is attempting to call. The solution simply is to wait until the port clears.

If **uucico** is already communicating with a given remote system, or if a lock file exists for a given remote system, you will see the message

```
ERROR: System already locked
```

in the log file for the site **uucico** is attempting to call.

Sometimes lock files are not cleared properly, and so tie up the system long after they should have been removed. Such files are called *stale* lock files. The solution is to use the command **uurmlock** to remove stale files.

Note that in some instances, permission problems may stop **uurmlock** from removing lock files. In this case, log in as the superuser **root** and execute the command:

```
rm /usr/spool/uucp/LCK*
```

Enabling Ports, /etc/ttys Problems

uucico reads a port's status in file **/etc/ttys**, then restores that status after it finishes its work. If you had disabled a port by hand, it remains disabled after **uucico** has worked with it — which means, of course, that no remote system can dial into your system via that port. To re-enable a port, use the command **/etc/enable**.

Note, too, that file **/etc/ttys** is sensitive to the order in which devices are named within it. The port into which you have plugged your modem must have an entry for both the remote device (e.g., **/dev/com1r**) and the local device (e.g., **/dev/com1l**). Note that the entry for the raw device *must* precede the entry for the local device. If it does not, **uucico** will not be able to dial out properly.

Note, too, that device **/dev/console** *must* be the last entry in **/etc/ttys**, or your system will not be able to dial out via a serial port.

Permission Problems

Incorrect permissions on files and directories will harm UUCP's behavior.

uucico runs as a user named **uucp** and therefore does not have any special permission or privileges that give it free access to every file or directory on your system. For example, consider what UUCP does when it attempts to contact a remote system:

1. The daemon **uucico** checks the directory **/usr/spool/uucp** to see if any lock files are present; these lock files indicate whether someone has already logged into the port from which **uucico** wishes to dial out.
2. If **uucico** finds a lock file for the port it wants to use, it checks the file **/usr/lib/uucp/sys** for information on an alternate way to contact the remote system. If **uucico** finds an alternate way to contact the remote system, it tries that way. If it does not, then it quits.

3. When **uucico** finds a port that is available, it then check file **/etc/ttys** to see if the port is already enabled for remote logins.
4. If the port is enabled, **uucico** disables the port and makes its call.

So far, so good. If, however, **uucico** does not have read and write permission on the port device from which it is attempting to make its call, its attempt to make the call will fail.

For another example of how directory-level permission affect the behavior of UUCP, consider how UUCP transfers files. When UUCP transfers files, it stores those files as temporary files in the directory **/usr/spool/uucp/***sitename*, where *sitename* is the name of the system with which UUCP is communicating. All files and directories under **/usr/spool/uucp** must be owned by user **uucp** and group **uucp**, or UUCP cannot transfer files correctly.

UUCP can also write temporary files into directory **/usr/spool/uucp/.Temp/***sitename*. This directory must also be owned by user and group **uucp**.

The permissions on the serial port from which you will dial out can affect the behavior of UUCP. UUCP must have permission to read and write to that port. The device specified by the **line** entry in **/usr/lib/uucp/port** should have permissions of **666** (see the Lexicon entry for the command **chmod**).

uucp should own all of its spooling directories. The *spooling directory* is the directory into which UUCP writes stores data and command files for the site being contacted. The spool directory for a given remote site resides under **/usr/spool/uucp** and is named after the remote site. For example, your system will use directory **/usr/spool/uucp/mwcbbs** to store files being exchanged with **mwcbbs**. Likewise, **mwcbbs** has the directory */usr/spool/uucp/yoursystem*, where UUCP stores files to be exchanged with *yoursystem*.

UUCP Cannot Find Its Own Files

If the command **/usr/bin/uucp** says it can not get its own name when you invoke it, then you give yourself a UUCP site name of no more than seven characters in the file **/etc/uucpname**.

The command **/bin/mail** command may also return a similar message. The cure is the same.

As noted above, a UUCP command may also fail to execute because permissions are set up incorrectly on the UUCP executables. UUCP commands frequently invoke one another. For instance, **uucico** invokes **uuxqt** after it has communicated with a remote system. **uuxqt** processes all files uploaded from the remote system. **uuxqt** may, in turn, invoke other commands — for example, it can invoke **smail** to deliver mail to a user on your system or forward mail to a user on another remote system. If the permissions on a UUCP executable are incorrect, it may find that, when it tries to complete a task, it does not have permission to write to a given directory or a log. A list of UUCP permissions appears in the Lexicon entry for **UUCP**. Make sure that the permission on your file conform to what is given there.

Modem Configuration

The commonest source of error lies in modem configuration. Each modem has its idiosyncrasies, and command languages differ from device to device; however, in general your modem should be configured as follows:

- Echo off.
- No result codes.
- Carrier detect (DCD) is true.
- Terminal ready (DTR) is true.
- DCD follows DTR.

If you are working with a high-speed modem (9600 baud or faster), you should also configure it to do the following:

- Lock modem to computer baud rate at 19,200 if your modem supports it;, if not, lock it to 9600 baud.
- Set modem for RTS/CTS handshaking.
- Use a flow-control device for the port's description in file **/usr/lib/uucp/port**. See the Lexicon entry **asy** for details on how to do this.

Telebit modems present special problems for configuration. They are designed to be used with UUCP, and in general they are fast and robust; but because they do more than ordinary modems, they require more extensive setup to work correctly. The Lexicon entry for **modem** gives detailed information on how to set up a Trailblazer modem so that it works properly with the COHERENT implementation of UUCP.

If your modem supports data compression, it may not be ideal to use this feature with every remote site. For instance, attempting to compress files that already are compressed (as are the files on the Mark Williams bulletin board), only adds to the data stream and *reduces* overall throughput. Before you turn on data compression, make sure that the files you are downloading are *not* compressed.

The Modem Does Not Respond

When you try to call a system via the commands **/usr/bin/uucp** or **/usr/lib/uucp/uucico** and the modem does not respond (i.e., the lights on the modem do not flicker), look at file **/usr/lib/uucp/port**. Check the permissions on the serial port used to dial out on, as specified therein.

In some cases, you will see the error message

```
Retry time not reached
```

Taylor UUCP puts a horizon on callouts: if a call to a given site fails, UUCP will wait a predetermined amount of time before it tries again. If you are repeatedly invoking **uucp** or **uucico** from the command line, you may see this error. To get around this limitation, use the command-line option **-f**; for example:

```
uucico -s systemname -f
```

This problem can also arise if a previous connection to a site failed. In this case, UUCP writes a bad-connection report into file

/usr/spool/uucp/.Status/*systemname*

where *systemname* names the system you are trying to contact. UUCP does this to keep your system from wasting time contacting a system whose connection is defective, even if you use the **-S** option with **uucico**. Remove this file and UUCP will resume dialing out. Note that this will not clear up the problem that triggered the original bad-connection report, and the connection may fail for other reasons.

The Modem Responds But Does Not Dial

In some cases, the modem responds (i.e., its lights flicker) but it does not dial out. This can have any of several causes.

First, the modem's register settings may be incorrect. Review them. Check the above example for some simple examples of how to set modem registers, and check the documentation that came with your modem.

You may be trying to access the modem through the remote COM port, e.g., **/dev/com1r**. In this instance, the system awaits a carrier signal, as you cannot open a remote COM port without; therefore, no communication ever begins. The solution is to use the local device, e.g., **/dev/com1l**. This way, the system will not wait for the carrier to come up on the modem, and dialing will begin.

The Modem Dials But No Connection Made

Sometimes a modem will dial out but no connection is made. This is typically caused by plugging the telephone line into the wrong port on the modem.

Check the log file for the site you are calling. It will usually give a message that indicates what the problem really is. If calling **mwcbbs**, use the command:

```
uulog mwcbbs
```

The Modem Dials, Carrier Is Established, Nothing Else Happens

The first suspect is the modem's register settings. The modem register settings that we discussed above generally work well for **uucp** to dial out to another system, *if* your modem is Hayes-compatible. If it is not, or if it is an off-brand that only *claims* to be Hayes-compatible, check your modem's documentation and make sure that the register settings are correct.

To get a picture of what is happening, run the command **/usr/lib/uucp/uucico** with the option **-xchat**. If calling **mwcbbs**, use the command:

```
/usr/lib/uucp/uucico -Smwcbbs -xchat
```

This tells your system to call **mwcbbs** and to write debugging information into its log file **/usr/spool/uucp/.Admin/audit.local**, which you can review later. This is very useful in determining if there is a problem in a chat script.

uulog Shows Lost Packets

If your UUCP communication sessions terminate prematurely, your system may be losing characters on the serial ports. An indication of this problem is the appearance of the message **Lost Packets** in your UUCP logs. If your system exhibits these symptoms during transfers on 4800-bps or higher-speed lines, *we strongly urge you to replace your existing 8250- or 16450-based UARTs with those based upon the 16550A design, such as the National Semiconductor NS16550AFN.* These newer UARTs are pin-compatible with the older UARTs. COHERENT automatically senses and enables them when it boots.

uulog Shows Incorrect Response

This points to one of four problems:

1. Your site is sending an improper site name to the remote system (in other words, the remote site doesn't know about your system).
2. The remote site does not have a spooling directory for your site.
3. Your site does not have a spooling directory for the remote site
4. **/usr/lib/uucp/sys** contains an error or incorrect chat script.

Files Refuse To Be Sent or Cannot Be Received

Check the instructions **local-send**, **local-receive**, **remote-send**, and **remote-receive** in the remote system's entry in file **/usr/lib/uucp/sys**. This can result in the remote system being denied permission to load files onto your system, or read files from it. Make sure these instructions are set correctly, and point to directories in which user **uucp** has read and write permission.

File Transfers Fail With imsg Statements

One problem is frequently encountered when COHERENT systems attempt to UUCP with Sun workstations: connections are made correctly, but when file transfers fail with numerous iterations of the debug message "imsg" until the script times out. This is caused by differences in parity: the COHERENT chat scripts by default use no parity, while those on the other system do. The solution is to change the chat script on the Sun system to set no parity when it talks with COHERENT system.

Files are Being Lost

If a configuration problem exists on your side of a UUCP connection, files could be lost. This may be true if **uucico** or **uuxqt** are running with incorrect permissions. Taylor UUCP notes problems of this nature in its log files. If it can preserve a file that would otherwise be lost, **uucico** saves the file in directory **/usr/spool/uucp/.Preserve**, and logs the fact that it has saved it.

Non-COHERENT UUCP Site Problems

It is important to understand that COHERENT's UUCP is designed to be compatible with other implementations of UUCP, but may not use the same configuration files. This can complicate the debugging of problems when you attempt to establish communication with a system that uses a different implementation of UUCP.

We will supply whatever assistance we can, but if it is determined that the non-COHERENT site is part of the problem, it is up to you to find how that non-COHERENT site has configured itself. You may even need to set up a conference call among yourself, the remote site's administrator, and MWC Technical Support.

Where to Go From Here

As we mentioned earlier, COHERENT does not now implement the entire Taylor UUCP package. Full sources for Taylor UUCP are available from **mwcbbs**, and from various sites on the Internet.

This tutorial only touches upon the esentials of configuring UUCP for communicating with other systems. Taylor UUCP is much more conformable than this tutorial may have led you to believe.

For a fuller description of the Taylor UUCP configuration files, see the Lexicon entries for **dial**, **port**, and **sys**. For further information, check the Lexicon entry for each UUCP command, as well as the overview article **UUCP**. This article will also point you to related articles in the Lexicon, such as the ones for **modem** and **RS-232**.