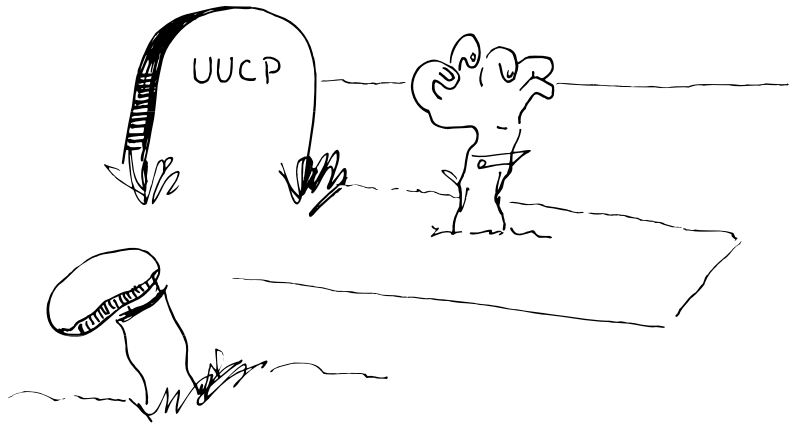
30 UUCP



30.1 INTRODUCTION

UUCP (UNIX-to-UNIX copy) is a set of programs and protocols that allow computers to communicate over serial connections, networks, or ordinary telephone lines. It supports two operations: file copying and remote command execution.

The history of UUCP, like that of Internet networking, dates back to the steamy dawn of UNIX computing. Over time, UUCP has been adapted for several generations of now-obsolete communications hardware. By now, UUCP contains more historical grotesquerie than useful code. However, it's still the most widely-used way for two UNIX systems to communicate using modems.

See Chapter 19 for more information about SLIP and PPP.

The functions of UUCP are a subset of those offered by Internet networking, and interest in UUCP has dwindled as it has become easier and cheaper to set up Internet connections. Even for computers whose connection to the outside world is through a modem, dial-up Internetworking systems such as SLIP and PPP may offer a better way of connecting two computers. UUCP is in an odd state where it is too obsolete to bother improving, and yet not quite obsolete enough to actually go away. We expect that there will be continued use of UUCP throughout the 90s.

There are, broadly speaking, two sorts of UUCP. The more common these days is HoneyDanBer UUCP, which is named after its authors: Peter Honeyman, David A. Nowitz, and Brian E. Redman. HoneyDanBer

UUCP is usually associated with ATT-ish versions of UNIX, although a number of BSD systems have also appropriated parts of it. The other kind of UUCP is closely related to the original V7 UUCP; it's associated with BSD and older ATT systems. The differences between the two are mostly superficial.

The two UUCP's have, over time, become inbred. Several vendors have formulated hybrid systems that include the features of both. It has also been common for vendors to switch flavors of UUCP between releases. For these reasons, it is hard to make general statements about what kind of UUCP you should expect to find on what systems.

In the first edition of this book, we discussed UUCP in relatively thorough detail. But due to the waning importance of UUCP, the proliferation of nonstandard systems, and the publication of several excellent UUCP books, we now cover only the features of UUCP that seem useful and relevant to modern computing. If you want to learn more about the UUCP arcana, the book *Managing UUCP and Usenet* by Tim O'Reilly and Grace Todino is a good reference. A full citation is given on page 704.

30.2 UUCP VS. SLIP AND PPP

See Chapter 19 for more information about SLIP and PPP.

UUCP is a batch system. If your modem connection is down more often than it is up, UUCP may be preferable to SLIP or PPP for some applications. UUCP is primarily used to transport electronic mail and Usenet news; if that's all you want your communication link to do, UUCP is both simpler and more secure than the IP-based alternatives.

UUCP was designed with slow serial lines in mind and is considerably more efficient at pumping information across a pair of modems than IP. If you make long-distance modem calls or share one phone line between voice and data, UUCP might suit your needs well.

If you're paying to connect to a service provider, UUCP may also be dramatically cheaper than SLIP, depending on your level of use. The price of SLIP connections is falling, but it still has a long way to go before it's within the reach of everyone.

SLIP and PPP offer dramatically more functionality than UUCP. It is theoretically possible for UUCP to be of some use other than mail and news, but in reality this just doesn't happen. For a taste of the goodies that come with Internet access, see Chapter 20.

You don't necessarily need to choose between SLIP and UUCP. With a little extra configuration work, you can run them both on the same modem. On the other hand, if you're going to go to the trouble and expense of setting up a SLIP or PPP connection, you might as well use it for everything and forget about UUCP.

30.3 AN OVERVIEW OF UUCP

In order to contact another site using UUCP, you must make prior arrangements with that site's administrator. You must tell your system how to contact the other site, and the other site must be configured to permit this contact. Once configuration is complete, the two computers can call one another without human supervision. This setup is usually called a UUCP "connection," even though the two computers are actually connected only on occasion. The sites you communicate with are called UUCP "neighbors."

UUCP operations are initiated with the **uux** and **uucp** commands. **uux** executes command on a remote system and **uucp** copies files. **uux** and **uucp** both create a file that describes the operation to be performed, which is then saved away in UUCP's work queue (**/var/spool/uucp** on most systems). Actual transmission of the jobs is handled by a separate process, **uucico**.[1] You, the site administrator, specify how often **uucico** is to be run.

UUCP does not directly support forwarding of files or command sequences; you may only communicate with machines to which you have established direct connections. However, transactions may involve more than one directly-connected host. For example, you may copy a file directly from one neighbor to another.

Since both mail and news require third-party forwarding, these systems provide their own forwarding facilities. Each time a mail message or news article makes a hop from one machine to another, it re-enters the mail or news system for further routing.

Both mail and news make use of remote command execution rather than file copying. Mail messages and news articles are sent as standard input to special mail- and news-receiving programs that try to determine their ultimate destinations. These programs accept the materials for local use or feed them back to UUCP for retransmission to another site (often, they do both).

30.4 UUCP ADDRESSES

Addressing systems are generally either route-based or destination-based. "Go straight through two lights, then hang a right at the House of Foam," is a route-based address; it tells you how to get somewhere by giving you a complete list of instructions. "900 Memory Lane" is a destination-based address. It only tells you *where* to go; you have to use a map or directory to figure out *how* to get there. UUCP is one of the last outposts of route-based addressing.

1. UNIX-to-UNIX copy in, copy out. Any similarity to "garbage in, garbage out" is incidental.

Commands in the UUCP system use the syntax *sitename*!*target* to identify remote files and commands. Mail and news have adopted a similar syntax for the specification of indirect UUCP paths. The basic and extended syntaxes are similar enough that we consider them together under the generic heading of "UUCP addresses."

A UUCP address consists of a list of names separated by exclamation points (or as UNIX hipsters call them, "bangs"). The last name (the target) is a user name, filename, or command name, depending upon whether the address is a mail address, file copy address, or remote command execution request. All other names leading up to the target are UUCP site names. The list of sites specifies a sequence of UUCP hops that can be followed to get to the appropriate machine. If the address is to be used with a native UUCP command, there can usually only be one site specified.

For example, the mail address

```
foo!bar!roger
```

would route mail to somebody named Roger on the host "bar" via the host "foo."

In the past, the UUCP world was a vast sprawl of randomly-connected sites. The use of long paths for mail and news was common, and all kinds of special mapping systems and path-computing algorithms were used to figure out efficient routes from one place to another.

Of late, Internet connectivity has become so common that this thicket of UUCP paths is largely gone. Most UUCP connections are now supplied by commercial vendors that are connected directly to the Internet. These vendors usually provide one-hop news service, and they help with email routing by translating between the Internet and UUCP domains. In effect, the UUCP connection is a sort of private link between the vendor and the client site. From the perspective of the outside world, the client site appears to be on the Internet. Consequently, indirect UUCP paths are seldom seen anymore.

30.5 USER-LEVEL UUCP

Two programs can enqueue UUCP requests:

- **uucp** – copy files between hosts
- **uux** – execute commands on a remote host

The following sections describe these commands in detail. Although we speak of the commands as though they were meant for human users, it is more common for them to be invoked by other programs and scripts.

uucp: Copy Files

uucp is similar to **cp**, except that the locations of files are specified using UUCP syntax. Each path begins with a system name, followed by an exclamation point and a (possibly remote) filename. If the system name is null, the path is assumed to refer to a file on the local system and the exclamation point is optional. The filename must be specified in one of the following formats:

- A full pathname (e.g., **/usr/dict/words**)
- A pathname beginning with *~user/* (e.g., **~evi/proposal**)
- A pathname beginning with **~/** (e.g., **~/explain**)
- A relative pathname (e.g., **bin/bustbladder**)

The use of tildes ("~", and actually, you must call them "twiddles" if you want to be cool) to specify pathnames should be familiar from the shell. "**~**" denotes the user's home directory, and *~user* is the home directory of *user*. Tildes are interpreted on the specified system, so these expressions may mean something different from what you expect. For reasons that will become clear later, the "current user" used to evaluate expressions such as *~/file* isn't necessarily you; this syntax often proves too treacherous for real-life use.

The full pathname syntax means about what you would expect: the name is simply looked up on the appropriate system. The relative pathname syntax is trickier because **uucp** always inserts the absolute pathname of the local directory from which it is run in front of the relative pathname. For example, if you **cd** to **/staff/garth** and execute

```
uucp akelei!files/exabyte exabyte
```

uucp will try to copy the file **/staff/garth/files/exabyte** from the system akelei into **/staff/garth/exabyte** on the local machine, probably not what you want.[2]

The **uucp** command accepts a number of flags that modify its behavior in insignificant ways. You can have it send you mail when the copy is complete, force it to make a copy of local files rather than use the actual files themselves (in case you want to delete the files immediately), allow it to create directories as needed for storage of the transferred files, etc. It does *not* accept the **-r** flag to copy a directory hierarchy recursively.

One thing that you must keep in mind when using any UUCP command is that the operations you request will ultimately be performed by a

2. Various shells interpret "!" and "~" as metacharacters, so they must be backslashed on the command line if they are to reach **uucp** unmodified. We have omitted this detail for clarity in this chapter's examples. If you get an error message like "event not found," you have probably omitted a backslash.

uucico process with UID and GID different from your own. Files you want to manipulate must be world-readable or world-writable.

For example, if you want to copy a file from a remote system into your home directory, you must give write permission on your home directory to everyone—this is *not* recommended. To copy files from the local system to a remote system, you must give read permission on them to the world. Files that UUCP creates are usually owned by the pseudo-user uucp, not by you, no matter where they are located. If you have **uucp** copy a file into your home directory, you will probably be able to read it but not modify it. You will have to copy the file to a different filename and then delete the original (which you can do because you have write permission on your home directory, even if not the file).

Because of all these permission and ownership headaches, there is a special world-writable directory, usually **/var/spool/uucppublic**, available for use with UUCP. It is analogous to the system-wide **/tmp** directory in that anybody can put files there, but it is intended for use primarily when performing UUCP operations. You can have **uucp** put transferred files here for you to pick up later.

Since **/var/spool/uucppublic** is used for temporary storage of files that are on their way into or out of the system, it has a tendency to accumulate unwanted junk files. As with **/tmp**, you should run a cleanup script out of **cron** to delete these files. See *Cleaning the Filesystem* on page 176 for more information about how to do this.

uux: Execute Commands on Other Systems

uux is used to initiate remote command execution. **uux** accepts shell-style command strings in which each file or command name is a network path, specified in the same syntax used by **uucp**. **uux** understands the shell metacharacters “>”, “<”, “|”, and “;”. Since the command string must appear to **uux** as a single argument, it should be quoted on the command line.

For example, the command

```
uux '!diff hal!/usr/jack/rabbit moon!/res/hare > !diffs'
```

invokes the local command **diff** on the files **/usr/jack/rabbit** from the machine hal and **/res/hare** from the machine moon, sending the output of the command to the file **diffs** on the local machine.

30.6 UUCP Data Transport

Inter-machine communication is managed by the **uucico** program. There are a variety of programs in the UUCP system for queuing requests and querying the state of connections, but only **uucico** can

make actual transactions. When two systems have a UUCP "conversation," a copy of **uucico** runs on each machine.

uucico can operate in two modes: master and slave. At any given time, one of the two **uucico** processes plays each role. The two **uucico**'s switch roles occasionally so that large outgoing or incoming jobs do not block data flowing in the other direction.

In master mode, **uucico** attempts to deal with requests that have been queued on the local system. Satisfying these requests may involve sending a file to the slave site, requesting a file from the slave site, or requesting execution of a program at the slave site. The designation of master and slave is independent of the current sender or receiver of information; if the master has requested a file from the slave, it will be both the master and the receiver. In slave mode, **uucico** listens to the orders issued by the remote site and attempts to carry them out.

To initiate a conversation, a **uucico** process on one machine must contact another machine and start up a remote **uucico** process to communicate with. This is usually done by making a special login account on the remote machine that uses **uucico** as its login shell. This way, the calling **uucico** need only log in to the remote machine in order to establish a **uucico**-to-**uucico** connection.

As soon as the two **uucico** processes have been connected, they negotiate a communication protocol to use during the subsequent conversation. The standard protocol used over telephone lines is called the "g" protocol, and every version of UUCP understands it. Other protocols exist, but they are mostly intended for use on reliable circuits. The "g" protocol is fully error-correcting, so transferred data are guaranteed to arrive at their destination intact.[3]

30.7 SETTING UP UUCP

Setting up UUCP is not difficult, but some of the steps can be time-consuming. Two sorts of tasks are involved: tasks that are performed once when the system is first set up, and tasks that you must repeat each time you acquire a new UUCP neighbor.

We will go through the steps in the order you need to complete them when setting up your first UUCP connection. You may be able to parallelize some steps; read through the entire procedure before starting.

- Identify your UUCP software.
- Verify software configuration.
- Connect modems.
- Describe your modems to UUCP.

3. Ha!

- Enter telephone prefix codes.
- Prepare an access control specification.
- Find a UUCP neighbor.
- Make a login for the new neighbor.
- Make an **L.sys** or **Systems** file entry for the neighbor.
- Debug the connection.
- Configure your mail system to use UUCP.
- Set up automatic calling using **cron**.

Identifying your UUCP Software

There are three important directories in the UUCP system:

- **/usr/lib/uucp** – for storage of executables and config files
- **/var/spool/uucp** – for storage of jobs and temporary files
- **/var/spool/uucppublic** – for use by users

Most of UUCP's user-level commands are kept in **/usr/bin**.

You may find that your system has bonus UUCP directories. On some systems, **/usr/lib/uucp** has been moved to **/etc/uucp**. You might also find that the UUCP spool directory is **/usr/spool/uucp** rather than **/var/spool/uucp**.

In this chapter, we'll refer to UUCP directories by the most commonly-used pathnames. If your system is set up differently, you'll need to remember to map each path that we mention to its local equivalent.

Take a look at the contents of your **/usr/lib/uucp** directory. Do you see files named **Devices**, **Dialcodes**, **Dialers**, **Permissions**, or **Systems**? If so, your UUCP is HoneyDanBer-ish. If you see files named **L-devices**, **L-dialcodes**, **L.cmds**, **L.sys**, or **USERFILE**, your UUCP is BSD-ish. If you see none of these files, or a combination of both sets, your system is nonstandard (or, in technical terminology, "weird").

On some systems, UUCP software is considered optional and must be specifically loaded from the distribution CD-ROM (or tapes) when you install the system. Your manuals will explain, perhaps.

If the **/usr/lib/uucp** directory exists but contains only the commands **uucico**, **uuclean**, and **uuxqt**, then your system is probably set up without any example configuration files to look at.

A brief summary of the files usually found in **/usr/lib/uucp** is shown in Table 30.1 on the next page. There isn't exactly a one-to-one mapping from BSD to HoneyDanBer, but the two systems are fairly close. HoneyDanBer also provides several commands that have no BSD equivalents (not all UUCP commands are shown in the table).

Table 30.1 Files in the /usr/lib/uucp directory

BSD	HDB	Purpose
L-devices	**Devices**	Specifies types of available modems
	Dialers	Modem dialing instructions
L-dialcodes	**Dialcodes**	Phone number database
L.aliases		Lists hosts that change their names
L.cmds	**Permissions**	Lists commands available via **uuxqt**
USERFILE	**Permissions**	Filesystem access specifications
L.sys	**Systems**	List of UUCP neighbors, login scripts
uucico	**uucico**	Manages inter-machine communication
uuclean	**uuclean**	Cleans up spool directory
uuxqt	**uuxqt**	Remote command execution server

Verifying your Software Configuration

Everything in **/usr/lib/uucp** should be owned by the user uucp, and the programs stored there should be setuid to this account. In addition, **/var/spool/uucp** and **/var/spool/uucppublic** should be owned by uucp and should have group owner daemon. Check these ownerships carefully, as several vendors ship with bogus permissions.

In **/usr/lib/uucp**, no file should be writable by anyone but its owner (the uucp account), and the files **L-devices**, **L-dialcodes**, **L.cmds**, **L.sys**, **Devices**, **Dialcodes**, and **Systems** (whichever are present) should be readable only by the owner.

The **/var/spool/uucp** directory should have the permissions:

```
drwxr-xr-x 13 uucp daemon     512 Jul 15 02:32 uucp
```

Inside **/var/spool/uucp**, all subdirectories should have mode 775. The BSD log files should have the permissions

```
-rw-r--r--  1 uucp daemon      71 Jul 15 23:20 ERRLOG
-rw-rw-r--  1 uucp daemon  662326 Jul 16 19:25 LOGFILE
-rw-rw-r--  1 uucp daemon  301847 Jul 16 19:25 SYSLOG
```

and the HoneyDanBer permissions should be

```
drwxr-xr-x  2 uucp daemon     512 Feb  8  1990 .Admin/
drwxr-xr-x  2 uucp daemon     512 Feb  8  1990 .Corrupt/
drwxr-xr-x  6 uucp daemon     512 Feb  8  1990 .Log/
drwxr-xr-x  2 uucp daemon     512 Feb  8  1990 .Old/
drwxr-xr-x  2 uucp daemon     512 Feb  8  1990 .Sequence/
drwxr-xr-x  2 uucp daemon     512 Feb  8  1990 .Status/
drwxr-xr-x  2 uucp daemon     512 Feb  8  1990 .Workspace/
drwxr-xr-x  2 uucp daemon     512 Feb  8  1990 .Xqtdir/
```

/var/spool/uucppublic should exist, should be owned by user uucp and group daemon, and should have mode 777.

See Chapter 6 for more information about creating user accounts.

If your system does not have the correct permissions or ownerships, you may need to add the uucp login before you can continue. Use an asterisk as uucp's password, since no one will be logging in to the account.[4] uucp's home directory should be **/var/spool/uucppublic**, and its shell should be **/usr/lib/uucp/uucico**. The UID should be within the range you reserve for pseudo-users (usually less than 100).

Connecting Modems

UUCP uses modems in the same way that other modem-using programs do, so everything should be set up as described in Chapter 8, *Serial Devices*. Device files should be owned by user uucp and group daemon, and should have mode 664; these ownerships and permissions are also those correct for use with **tip** and **cu**. Either one of these programs can be used to verify that the modem is connected correctly and that you can dial out to other sites.

If you plan to use the same modem as both a dial-in and a dial-out line for UUCP, there are some additional tasks that you must perform. Refer to Chapter 8.

Describing your Modems to UUCP

UUCP only pays attention to its own configuration files. Even though you may have had to edit **/etc/ttys**, **/etc/remote**, **/etc/inittab**, or other system-wide configuration files to set up your modems, you still need to give UUCP its own copy of the information.

There are a variety of standards in the modem world. Different modems can't necessarily be controlled the same way, nor can they always communicate with each other. These incompatibilities give rise to two separate problems. The first problem is that you can't necessarily use the same modem to communicate with all of your UUCP neighbors. UUCP solves this problem by allowing you to specify explicitly what kind of modem should be used for each neighbor.

The second problem is that UUCP itself must be able to control your modems in order to place calls, set communication speeds, and detect problems. Different versions of UUCP solve this problem in different ways. HoneyDanBer provides the **Dialers** file, where instructions for controlling modems are encoded in a standard format. Some UUCPs have a file called **/etc/acucap** or **/usr/lib/uucp/modemcap** that serves the same purpose; it's patterned after **/etc/termcap**. In BSD UUCPs, there is a standard set of modem types that are supported; if support for your modem isn't compiled into the code, you're out of luck.

4. Some sites use "uucp" as the generic login for all clients. This works, but we recommend giving each system its own login. This helps to control access to your system and lets you disable one site's login without disabling everyone else's. See page 692.

But fear not. Almost any modem you are liable to encounter will conform to the Hayes command syntax, and it is highly unlikely that you will need to perform any special configuration. If you're not sure exactly what kind of modem you are dealing with, try your system's "Hayes-compatible" setting first. This will work in 95% of cases.

Modem information is stored in **/usr/lib/uucp/L-devices** on BSD systems and in **/usr/lib/uucp/Devices** on HoneyDanBer systems. The formats are slightly different, but both files record more or less the same information: the type of modem being used, the device file through which it can be accessed, and the modem's speed.

In most versions of UUCP, the UUCP configuration files (**L-devices** and **Devices** included) share a common syntax for comments and continuation lines. Lines beginning with a pound sign ("#") are ignored, and lines ending with a backslash are joined with the following line before interpretation. Check to be sure your system supports continuation lines before using this feature. Also, be sure to start each configuration line in the first column, since some versions of UUCP are confused by initial spaces.

Each line in **L-devices** (or **Devices**) represents one piece of equipment that can be used to connect to other computers. The format for modem entries is much the same in both files; we will describe the BSD **L-devices** format and note differences for HoneyDanBer.

Each line in **L-devices** looks like:

```
type device dialer speed brand chat
```

In your system's documentation, these fields are probably referred to by different names: *caller*, *device*, *call unit*, *class*, *dialer*, and *chat*. Some of the fields in **L-devices** are used in different ways depending on what type of device is being described, and so the field names in standard documentation are more vague. We'll only be discussing modems here, so we've chosen explicit names for clarity.

Fields are separated by whitespace and have the following meanings:

type The *type* field indicates what sort of device the line describes. For modems, this field should contain the letters ACU, which stand for "Automatic Calling Unit." Some other codes you might see here (in BSD) are DIR for a dedicated serial line and TCP for an Internet link. TCP links are provided mainly to ease the transition from UUCP to Internet; you simply change ACU to TCP when you become an Internet site, and data addressed to UUCP neighbors will be delivered over the Internet.[5]

5. As long as your neighbors are running **uucpd**.

device The *device* field specifies the name of the device file through which the connection is to be made. For a modem this is something like **/dev/cua0** (often a link to **/dev/ttya** or whichever serial port your modem is connected to).

dialer This field is obsolete. Put a dash or the word `unused` here as a place holder.[6]

speed The *speed* field tells how fast a modem can communicate, in bits per second. In BSD, modems that support several speeds (and most do) must have multiple entries in **L-devices**, each with a different speed.

If your UUCP is HoneyDanBer-ish, you can directly specify a range of speeds in the *speed* field; for example, `300-9600`. You can also use the word `Any` to specify that a modem is fast enough to use for any site.

brand This field identifies the particular brand and model of modem. For modern modems, this field usually contains `hayes`. (`hayestone` and `hayespulse` can also be used in BSD to specify touch-tone or pulse dialing).

Some old releases of SunOS use a modification of the *type* field instead of a separate *brand* field. On these systems, you'd use a type of `ACUHAYES` instead of type `ACU` and brand `hayes`.

On HoneyDanBer systems and others with user-configurable brand definitions, the value of the *brand* field identifies an entry in **Dialers**, **acucap**, or **modemcap**.

chat The *chat* is a miniature script used to gain access to the communications device. Usually you don't have to put anything in this field, but it can be useful if the modem lies on the other side of a data switch. It can also be used to send a command string to the modem to put it in Hayes-compatible mode, if that is not the default.

The format of the chat script is identical to that of the login script used in the **L.sys** or **Systems** file, but since scripting is rarely used in **L-devices**, we will delay a discussion scripts until page 696. Some versions of BSD-style UUCP do not allow a chat script. In HoneyDanBer, you must use the **Dialers** file to specify the script for a device.

6. In the past, the device that dialed the telephone was often different from the device that communicated over the connection once it was established. To use UUCP in these situations, two device files had to be specified. But these days, modems have dialers built in.

As you can see, the only really interesting tidbits in the **L-devices** file are the device file and speed of the modem. But sometimes, a modem has other important properties that UUCP must be made aware of. For example, you might own only one or two modems that speak a certain high-speed protocol. To connect to a neighbor with this high-speed protocol, you need a way to identify modems so that UUCP knows they are appropriate for use with that neighbor.

This kind of information can be sneaked into **L-devices** or **Devices** by adding a non-numeric prefix to the front of the *speed* field. For example, if a modem understands the MNP-5 error correction protocol at 19,200 bits per second, you might put `MNP19200` in the speed field. In other UUCP configuration files, you can specify that an `MNP19200` modem is needed to talk to a particular site. UUCP just matches up the letters, so you may pick your own prefix conventions.

Suppose your site has a Hayes modem (**/dev/cua0**) that runs at 1,200 and 2,400 baud and a Telebit modem (**/dev/cua1**) that understands advanced protocols such as V.32bis, V.42bis, and MNP-5 at speeds up to 19,200 bps. **L-devices** might look like this:

```
ACU     /dev/cua0     unused     1200      hayestone
ACU     /dev/cua0     unused     2400      hayestone
ACU     /dev/cua1     unused     T19200    hayestone
```

Entering Telephone Prefix Codes

All versions of UUCP have a macro facility that allows you to define text names for phone numbers and prefixes of phone numbers. BSD-ish systems store this information in **/usr/lib/uucp/L-dialcodes**, and HoneyDanBer puts it in **/usr/lib/uucp/Dialcodes**. The format is the same: each line lists a name and a phone prefix, separated by whitespace. For example,

```
# Phone number for Ubob (Bob Watson's consulting company)
#
ubob_phone 555-2368
```

defines ubob_phone to mean the phone number 555-2368. In the **L.sys** or **Systems** file, where UUCP neighbors are defined, you could list the phone number for this site as ubob_phone instead of 555-2368.

This facility allows you to put the phone number for a site in a publicly-readable location. **L.sys** and **Systems** contain account names and passwords for remote sites, so they must be readable only by the UUCP agents and root. Phone numbers can sometimes help users to debug a connection when no system administrator is available, so it is often nice to put them in a separate file.

Conversely, you may want to use the dialcodes file to hide long-distance billing codes. For example, if `genacct` is defined to access a long distance service and bill the call to the general expenses account, then something like `genacct1-303-555-2368` can be used from **`L.sys`** or **`Systems`** to specify both the phone number and the billing.[7]

Preparing an Access Control Specification

See Chapter 23, Security, for a more general discussion of security issues.

It is not a good idea to give your neighbors free run of your system via UUCP. All versions of UUCP allow you to specify which files your neighbors may access via the **`uucp`** command and which commands they may execute via **`uux`**. You should plan and implement a UUCP security strategy before you establish any UUCP connections.

BSD and HoneyDanBer handle UUCP security in completely different ways. BSD uses two files in **`/usr/lib/uucp`**, **`USERFILE`** and **`L.cmds`** (sometimes **`L-cmds`**), to control access to files and commands, respectively. HoneyDanBer uses the file **`/usr/lib/uucp/Permissions`** to specify both kinds of security.

We will treat the two systems separately in the discussion below. But before we launch into the specifics, let's discuss some general principles of UUCP security.

The main thing to keep in mind is that even though a remote site might have no more access to your site than a local user, this is usually enough to compromise the security of your installation. Even if system files can't be directly written to, they can be copied away for remote analysis and scrutinized for weaknesses. And on most systems, some user files are sure to be available for casual browsing.

It is safe to permit unrestricted access to **`/var/spool/uucppublic`**. But think carefully before allowing access to any other part of the system. If you must make additional files available, be as restrictive as you can; don't allow access to **`/var`** just so that a remote site can get to the **`/var/spool/news`** directory, for example.

See Chapter 24 for more information about Usenet news.

All your precautions will be nullified unless remote command execution is also carefully restricted. Unless you're involved in a special situation, any access to commands other than **`rmail`** (for receiving email), and **`rnews`** (for receiving news) is highly suspect.

Both flavors of UUCP allow permissions to be set on a per-site or per-user basis (except for **`L.cmds`** in BSD, which applies to all sites). A tricky feature of UUCP security is that access depends on several different con-

7. Most modems will ignore the dashes, but you can insert special punctuation marks if your modem understands them. For example, a comma signifies a pause in the Hayes command language. Refer to Chapter 8, *Serial Devices*, for more information.

ditions, not just one fact such as the identity of the remote system. In various situations, the role played by **uucico** (master or slave), the identity of the requesting user, the login name under which a remote site logged in, and the initiator of the telephone call can all make a difference in the computation of permissions.

Permissions that seem to say clearly, "Remote site wasteheat can never access anything outside **/var/spool/uucppublic**" often don't say that at all. You must be conversant with the subtleties of these configuration files in order to make them mean what you want them to mean.

Because of all this nastiness, you may find that the following sections attempt to teach you more about UUCP security than you really care to know. If you just want to use a secure, boilerplate configuration, here's what to do. Under BSD, set up the **USERFILE** like this:

```
unused,   /var/spool/uucppublic
,unused   /var/spool/uucppublic
```

and **L.cmds** like this:

```
PATH=/bin:/usr/bin:/usr/ucb:/usr/local/lib/news
rmail
rnews,Error
```

On a HoneyDanBer system, set up the **Permissions** file like this:

```
MACHINE=OTHER COMMANDS=rnews:rmail
```

BSD Security: USERFILE and L.cmds

USERFILE restricts the files that are available for access, while **L.cmds** restricts the programs that may be run via **uux**. In addition to restricting the access allowed to remote systems, these files also restrict the access that users have to the local system when using UUCP commands. For example, suppose that your system name is "jove" and that someone on one of your UUCP neighbors executes

```
uux '!cat jove!/etc/passwd !myfile | jove!mail evi'
```

Permission to execute the **cat** command on the remote system (meaning the system that originated the command) must be granted by the **/usr/lib/uucp/L.cmds** file on the remote system. Likewise, permission to access **myfile** on the remote system must be granted by the remote system's **USERFILE**. Assuming that the remote system permits this command, its success on jove depends upon whether or not jove's **USERFILE** allows access to the **/etc/passwd** file and whether or not jove's **L.cmds** file allows access to the **mail** command. The argument **evi** in this example is neither a file nor a command, so it is not subject to any security checks. Note that it does not matter whether or not jove allows use of the **cat** command.

Format of the USERFILE

Each line in the **USERFILE** is of the form

```
loginname,systemname [c] pathname ...
```

loginname and *systemname* must be separated by only a comma. The other fields may have spaces or tabs between them.

When file access permissions are to be determined, the **USERFILE** is read sequentially until a line that applies to the current situation is found. For local **uucp** or **uux** processes, or when **uucico** is run in master mode, the file is searched for a line that has a *loginname* field identical to the login name of the current user. When **uucico** runs in slave mode, it is the *systemname* field that is examined instead; a line is selected when this field matches the actual name of the remote site (*not* the site's login name, e.g., "moto" rather than "Umoto"[8]).

If no matching lines are found, the first line with an empty field in the appropriate spot is used. If there is more than one matching line, the first matching line is used.[9] Either of the two first fields may be empty on any line. In some versions of BSD, both fields may be empty.

The character `c` (callback) that follows the *loginname* and *systemname* fields is to be either entered literally or left out. It has meaning only in the case where the program examining the **USERFILE** is a slave-mode **uucico** process. If the `c` is present, the connection between the local and remote **uucico**s is broken and the local **uucico** attempts to call the remote system back immediately, using its own idea of the remote system's telephone number. This provides a degree of security against impostors, yet ensures that either side of the connection may initiate a conversation.

The pathname list consists of directories or prefixes that filenames must begin with to be accessible by the remote system. For example, the directory name **/var/spool/uucppublic** in this field would allow access to any file within that directory or one of its subdirectories.

To illustrate the subtleties of the **USERFILE**, imagine that there are two sites that jove talks to: one called "goodsite" and one called "badsite." Assume that we'd like to allow goodsite to access files anywhere underneath the **/usr** or **/etc** directories, but we want badsite's access restricted to **/var/spool/uucppublic**. Since goodsite has access to sensitive information, we want to be sure that it is not impersonated and we'll put it on callback status.

8. By convention, UUCP logins begin with a capital "U" to distinguish them from real users. See page 692 for more information about the naming of UUCP logins.
9. Actually, there are some versions of UUCP that use the last matching line.

Imagine further that there is a user "badguy" on the local system who we suspect of subversive activities. We'd like to prevent him from shipping out the system's files, but we want everyone else to have free run of the system. Here's what the **USERFILE** should look like:

```
Ugoodsit,goodsite   c /etc /usr
Ubadsite,badsite    /var/spool/uucppublic
badguy,unused       /dev/null
,unused
unused,             /var/spool/uucppublic
```

The line that gives permissions for badsite is not strictly necessary, since those permissions match the defaults specified for other systems on the fifth line. However, it is wise to include such a line in case you decide to liberalize the default access permissions. You might forget to rescind badsite's default permissions at that time.

Login and system names specified as `unused` are given to prevent matching of null fields. It is important to provide the login names of the remote sites as well as their system names, since some remotely-executed programs may attempt to forward files or execution requests by executing the **uucp** or **uux** commands.

This example setup is dangerous in several ways. To begin with, you shouldn't give permissions on **/etc** or **/usr** to external sites. And unfortunately, new contacts will by default have access to all files on the system when remote command executions they request cause **uucp** or **uux** to be executed, because of line four. But without line four, local users couldn't access their own files when using **uucp** or **uux**.

Format of the L.cmds File

/usr/lib/uucp/L.cmds contains a list of commands accessible via remote command execution, and, optionally, a list of directories in which to search for these commands. The default directories are **/bin**, **/usr/bin**, and **/usr/ucb**. If you intend to run news, you should add the directory that contains news commands (**/usr/lib/news/bin** or **/usr/local/bin**) by adding a configuration line of the form

```
PATH=/bin:/usr/bin:/usr/ucb:/usr/local/bin
```

to the **L.cmds** file.

Command names in **L.cmds** are listed one per line, and may have the optional suffixes "`,Error`" and "`,No`", indicating that acknowledgment messages are to be sent to the initiator of the command request only in the case of an error, or never, respectively.

Typically, the **L.cmds** file allows execution of the commands **rmail** and **rnews**. **rmail** is the mail-receiving program used in conjunction with

UUCP, and **rnews** is the standard receiving program for Usenet news. **rnews** should be specified as `rnews,Error`. If there are other commands that you want to include in your **L.cmds** file, you should be sure that they do not have any kind of built-in shell escape facility, and the shells themselves should be off limits.

HoneyDanBer Security: Permissions

/usr/lib/uucp/Permissions is essentially no different in purpose and scope from its BSD equivalents, but it uses a flexible file format that provides for a few more bells and whistles. HoneyDanBer UUCP is much more secure than BSD in its default configuration.

Be careful with the syntax of this file, as small errors can cause unintelligible permission errors in other parts of the UUCP system. Luckily, HoneyDanBer provides the **uucheck** command which, with the **-v** option, can be used to explain how the contents of the Permissions file will be interpreted.

Statements in the **Permissions** file are sequences of *name=value* clauses separated by whitespace. Long lines can be broken using the backslash convention, and multiple values for a name may be separated with colons. The properties that may appear as names are listed in Table 30.2. The "When" column indicates whether a property applies when the local system is the caller (the initiator of the conversation), the callee, or both.

Table 30.2 Properties in /usr/lib/uucp/Permissions

Property	When	Value
MACHINE	Caller	Machines this statement applies to
LOGNAME	Callee	Login names this statement applies to
VALIDATE	Callee	List of system names OK for this login
READ	Both	Directories the remote site can read files within
WRITE	Both	Directories where files can be deposited
NOREAD	Both	Exceptions to READ
NOWRITE	Both	Exceptions to WRITE
REQUEST	Both	Can remote site ask for your files? (yes/no)
SENDFILES	Callee	OK to perform locally-requested tasks? (yes/call)
COMMANDS	–	List of commands the remote site may execute
CALLBACK	Callee	Call the remote site back immediately? (yes/no)
MYNAME	Both	System name alias

In order to determine whether a particular operation should be permitted, **uucico** searches the **Permissions** file for a line that is applicable to the current situation. When the local site is the caller, **uucico**

searches for a line with a MACHINE clause that matches the system name of the system that was called. When the local site is the callee, **uucico** looks for a LOGNAME clause containing the account name that the remote site used to log in.

There are two exceptions to this rule. First, when **uucico** wants to verify that a remote site's command execution request is legitimate by checking the value of COMMANDS, it looks for a matching MACHINE entry, even if the remote site initiated the call. Second, if no matching MACHINE or LOGNAME clauses are found during a permission search, a statement that specifies MACHINE=OTHER, if one exists, will be used.

The remaining properties specify various permission details. In the explanations below, we'll refer to the following two statements:

```
LOGNAME=Urhino:Ubob MACHINE=rhino REQUEST=no \
   SENDFILES=yes READ=/var/spool NOREAD=/var/spool/mail \
   WRITE=/tmp:/var/spool/uucppublic COMMANDS=rnews:rmail \
   CALLBACK=yes
MACHINE=akelei MYNAME=titan REQUEST=yes
```

A single statement may contain both MACHINE and LOGNAME clauses. Thus, properties described in the first statement apply to the site "rhino" whether or not rhino initiates a conversation (assuming that rhino logs in with the account name Urhino).

READ and WRITE list directories within which the remote site should be granted read and write permission. Reading and writing are always defined from the local site's perspective, regardless of who initiates a conversation. Normal UNIX file permissions still apply; even if UUCP allows a remote site to read files in **/usr/foo**, they might not be readable by the UNIX account that the remote site uses to log in. Both READ and WRITE default to **/var/spool/uucppublic**.

NOREAD and NOWRITE specify exceptions to READ and WRITE. It's often clearer to use these exception clauses to mask out certain directories than to explicitly list the directories' siblings in a READ or WRITE. In the first of the examples above, rhino and bob can read files from anywhere in **/var/spool** except **/var/spool/mail**.

REQUEST and SENDFILES control the transfer of files from the local site to the remote site. It is assumed that files may always be transferred in the other direction. REQUEST tells whether it's OK to transfer files at the remote site's request, and SENDFILES tells whether it's OK to transfer them at the local site's request.

There is a subtle difference between these two properties: if REQUEST is turned off, remote requests to transfer files will fail and be discarded. But if SENDFILES is turned off, it simply means "not now"; the files can

be transferred later when the local site is the initiator of a subsequent conversation. That's why the opposite of `yes` is `no` for `REQUEST` and `call` for `SENDFILES`. The defaults are `no` and `call`. (In the example above, there's no way for rhino to ever get a file from the local site without someone at the local site sending it.)

`COMMANDS` lists the commands that a remote site may execute. The default value is compiled into UUCP and is implementation-dependent.

`CALLBACK` is only checked when a remote site dials in. If its value is `yes`, the conversation is aborted and the local site attempts to call the remote site. This provides a measure of extra security, since an impostor would have to take over the remote site's phone as well as its UUCP information. If `CALLBACK` is turned on, only `MACHINE` entries relating to particular host will be significant, since the local site will always be the caller by the time that the **`uucico`** processes begin to transfer files.

`VALIDATE` is a not-very-reliable way to check that a caller is who it claims to be. It checks the login name against the listed system names; if the caller's system name doesn't match, the connection is terminated.

The `MYNAME` property allows you to masquerade as a site with a different system name. The value should be set to the system name you would like to adopt. This can be useful when changing your system name; your UUCP neighbors won't all update their configurations immediately, and people in the outside world may persist in routing UUCP traffic using the old name.

Armed with this exciting information, we can now decode the example configuration shown earlier. From the first statement, we see that the system rhino can never request files from us, but that we are willing to send files queued as the result of local commands any time we talk to rhino. Any files under **`/var/spool`** can be sent, except for those in or beneath **`/var/spool/mail`**. Rhino can only write files in the directories **`/var/spool/uucppublic`** and **`/tmp`**. Rhino is on callback status, and may only execute the **`rnews`** and **`rmail`** commands. All of these conditions also apply to the system "bob" when bob calls us; when we call bob, the default settings are used.

The second statement says that when we call "akelei," we should pretend to be a site called "titan," and that we should let akelei request files from **`/var/spool/uucppublic`** (the default value of `READ`).

Finding a UUCP Neighbor

Before you can go much further with your UUCP configuration, you'll need a UUCP neighbor to talk to. It may take time to arrange a connection, so it's best to plan ahead.

In the past, obtaining a UUCP connection required begging skills, since local universities and businesses were often the only potential connection points. But these days there are companies that offer UUCP connections as a commercial service. The fees for a connection are usually reasonable, and it's now considered somewhat rude to cadge free UUCP connections from universities.

Even if you can obtain a free UUCP connection from another site, you may still be better off using a commercial supplier. Suppliers are usually well-connected to networks other than UUCP (including Internet), and it's therefore faster to get materials to their proper destinations. If your site is only accessible through an indirect chain of UUCP sites, your risk of experiencing problems is increased.

See Chapter 16 for more information about DNS.

Another nice feature of using a commercial UUCP provider is that the provider can use DNS magic to give the world the impression that you are on the Internet. bill@whitehouse.gov is a more respectable address than uunet!goatboy!bill. Silly, but true.

Most service companies charge a flat monthly fee for UUCP service. Usually, you must also reimburse them for telephone charges incurred on your behalf. News is usually available for an additional charge.

Two large providers are UUNET and PSI; both operate world-wide. Send email to info@uunet.uu.net or info@psi.com for more information.

Once you've arranged a connection of some sort, each site must give the other the following four pieces of information:

- The machine name of the local system
- The remote system's login name and password
- The phone number of the local system
- The name and phone number of the system administrator

You must also reach agreement on the calling schedule and the modem speeds and protocols to be used. You might want to ask if there are any unusual features of the remote site's login procedure.

Making a Login for a New Neighbor

A UUCP login need not be a full-fledged user account, so you don't have to perform all the steps outlined in Chapter 6, *Adding New Users*. It is usually sufficient to add a line to the **`/etc/passwd`** file or to the equivalent network administrative database.

We suggest that the login name for a remote system be the name of the system prefaced with a capital "U" and truncated to eight characters. This convention makes UUCP logins recognizable on sight, allows you to figure out the login for a particular site without maintaining an explicit

table, and avoids problems with systems that are confused by long login names. For example, the login name for a machine called "watergate" would be "Uwaterga".

UUCP passwords should consist of random letters and digits, since no human needs to remember or type them.

All UUCP logins should share a single UID. This single-UID scheme avoids cluttering the pseudo-user UID range with UUCP logins and allows files created in **/var/spool/uucppublic** by one login to be read and rewritten by another. The shared UID should *not* be the same as the UID of the generic "uucp" login.

Once you've made the **/etc/passwd** entry for a new neighbor, use the **passwd** command (or **yppasswd**, if appropriate) to set a password for the remote system.

Making an L.sys or Systems Entry for a Neighbor

L.sys (BSD) and **Systems** (HoneyDanBer) are central to UUCP. These files, which are practically identical, contain the names, phone numbers, and passwords of all systems with which you communicate. They normally live in **/usr/lib/uucp**.

In some versions of UUCP, it is possible to receive calls from systems that are not listed in **L.sys** or **Systems**; however, it's never possible to place a call to another system without first entering the system's information in these files.

Because **L.sys** and **Systems** contain instructions for logging in to your UUCP neighbors, you must keep their contents confidential. Never give any permissions on these files to the world, and make sure that any backup copies are kept secure.

The format of an entry is

```
site times type speed phone chat
```

The fields have the following meanings:

site This field specifies the remote computer's host name (*not* the remote computer's login name).

times This field specifies the times at which a call may be initiated to the specified host. It doesn't place any restrictions on the times at which calls may be received from that host, however. See *Specifying when to call*, below, for instructions on filling out this field.

type This field specifies the type of connection to be used. It is identical to the *type* field of the **L-devices** or **Devices** file, and should be ACU for a modem.

speed This is the line speed to be used when communicating with the remote site. The speed may be prefaced with an arbitrary non-numeric code to be matched inside the **L-devices** or **Devices** file. For example, if you make **L-devices** entries for your Telebit Trailblazer modems that have the line speed specified as, say, T9600 rather than just 9600, you can then specify that calls to a given system are to be made with a Trailblazer by putting T9600 in the speed field of **L.sys**.

phone The phone number to call when making the connection. If the number includes a non-numeric prefix, it is looked up in **L-dialcodes** (BSD) or **Dialcodes** (ATT) and substituted before dialing occurs. It is possible to specify the entire phone number in the dialcodes file and simply reference it here without any additional numbers.

Some versions of UUCP interpret dashes in the telephone number as pauses and equal signs as requests to wait for a secondary dial tone (usually implemented as an extra-long pause). Other versions of UUCP pass the characters along to the modem; Hayes-compatible modems interpret dashes and commas as pauses. The characters "#" and "*" may be used like numbers for tone dialing systems.

chat The chat script is a program that specifies how to log in to the remote site. It usually includes instructions for entering the remote user name and password. See *Logging in to the remote system* on page 696 for details.

Specifying when to call

The *times* field of the **L.sys** or **Systems** file is used to restrict outgoing calls to particular times of day or particular days of the week. Different flavors of UUCP vary a little bit in the time formats they accept, but the general format of the field is

```
dayhhmm-hhmm/grade;timeout
```

All segments are optional except for *day*. On many systems, you can combine more than one specification by separating the instances with a comma or a vertical bar. On other systems, you simply duplicate configuration lines, changing only the *times* field. In either case, the meaning of multiple specifications is OR.

The *day* field should contain one of the codes listed in Table 30.3. This field limits the applicability of the subsequent hour range to certain days of the week, except in the case of the keywords Any, NonPeak,

`Night`, and `Evening`, which are comprehensive specifications that do not accept an additional hour range.

Table 30.3 Day keywords and their meanings

Keyword	Meaning
`Any`	Any day, any time
`Wk`	Weekdays
`Su`, `Mo`, `Tu`, ...	Sunday, Monday, Tuesday, ...
`Evening`[a]	5 p.m.-8 a.m. M-F, All day Sat. and Sun.
`Night`[a]	11 p.m.-8 a.m. M-F, All day Sat, Sun. from mid-5 p.m.
`NonPeak`[a]	6 p.m.-7 a.m. M-F, All day Sat. and Sun.
`Never`	Never call this site (can still receive calls, though)

a. BSD only.

The range of hours *`hhmm-hhmm`* further qualifies the *`day`* keyword. It is always optional, and if it is left out, any time of day is assumed to be OK. The range may span midnight.

The grade of a UUCP job is signified by one character from 0 to 9, A to Z, or a to z. Grade 0 is the highest priority, and grade z is the lowest. Mail is usually handled at grade C, news at grade d, and file copying at grade n. The grade of a job may be given as an argument to **uucp** and **uux**. Many systems do not support the grade flag.

If a grade is specified in the *`grade`* field, no jobs of lower priority than that specified may be transferred during the stated time. This feature is most commonly used to keep Usenet news off the wire during business hours, while permitting email to be transferred normally. On most systems, grade specifications apply only to sending; low-grade jobs can be received at any time.

`timeout`, also optional, tells the number of minutes to wait between successive (unsuccessful) attempts to contact the remote host. The default varies from system to system. An hour is a common default, as is a series of progressively longer delays ranging from five minutes up. Most versions of UUCP abandon the attempt to contact a remote system after a certain number of unsuccessful retries. Often, the permissible number of retries is constant, so a too-small value of *`timeout`* will cause the connection to be abandoned earlier than is reasonable. Retries are made subject to the usual time slot restrictions.

For example, the time specification

```
Wk0900-1330/C;5,Th0000-2200/d,Sa,Su
```

says that on any weekday, mail and higher grade jobs may be sent between the hours of 9:00 a.m. and 1:30 p.m., and that if a connection

attempt fails, to wait only five minutes before trying again. In addition, anything of news grade or higher may be sent any time on Thursday, except for the hours of 10:00 p.m. to midnight. On Saturday and Sunday, jobs of any grade may be transferred at any time.

A more realistic example is

```
Any/C,Evening
```

which says that calls may be placed at any time, but that jobs of lower priority than mail must wait until after 5:00 p.m. to be transferred.

Logging in to the remote system

The *chat* field of **L.sys** and **Systems** is a miniature script used to negotiate the remote site's login procedure. It consists of a number of expect/send pairs, where an "expect" is some string of characters to be expected from the remote host, and a "send" is a string of characters to be sent once the previously expected string appears, followed by a carriage return. There are often a number of special features that are not covered here; see your manuals for a thorough description.

A string consisting of two double quotes indicates the null string; a null send string is still followed by a carriage return. Several special escapes are described in Table 30.4.

Table 30.4 Chat script escape sequences

Escape	Alternate	Meaning
\b	BREAK	Line break of 3/10 second
\b*n*	BREAK*n*	Line break of *n*/10 second
\c		Suppress the CR at the end of send
\d	PAUSE	Delay 1 sec for \d, 3 for PAUSE (send only)
	PAUSE*n*	Delay *n* seconds
\r	CR	Carriage return
\n	NL	Newline
XXX		The character with octal value *XXX*
	EOT	End-of-transmission

The expect/send strings are normally separated from each other with spaces. There is a special case in which a dash ("-") is used as the separator; statements of this type are in the form

```
expect-send-expect-...
```

where the *send* field is sent only if the first *expect* is not received within 45 seconds, and after which it is the second *expect* field that is waited for. There may be an arbitrary number of *expect* and *send* strings joined together with dashes.

For most versions of UNIX, the chat script need not be complex. A script like the following will suffice in 99% of cases:

```
"" "" ogin:--ogin: Ujove ssword: s3jHd2
```

Here, "Ujove" is the login name and "s3jHd2" is the password. The first exchange has a null expect field, so the send is output immediately. There are no characters in the send field, so this exchange has the effect of always sending a carriage return to the remote system.

The next exchange waits for the string "ogin", presumably a substring of the system's login prompt. The first character has been left out because the case of the first letter may vary, and because noisy telephone lines tend to affect the first characters of a line more often than characters in the middle.

If the login prompt is not received in 45 seconds, another carriage return is sent and the script waits again for the same login prompt. If this second expect times out, the login script is aborted, since there are no more dash-separated clauses.

Once the login prompt has been obtained, the user name is output, the password prompt is waited for, and the login password is sent. It's expected that the login shell for the remote account will be **uucico**, so no further navigation is required.

If your UUCP neighbor doesn't supply you with a chat script, you can use the **tip** command to call the remote system interactively and see how its login procedure works. If you have the opportunity, you should call your own system and formulate a chat script that you can give out to new neighbors.

Debugging the Connection

The best way to debug your first connection is to get on the telephone with the other site's administrator and send a few jobs back and forth with **uucico** debugging turned on. To initiate a call with debugging, the syntax is usually something like

```
/usr/lib/uucp/uucico -r1 -ssystem -x9
```

where *system* is the name of the system to call. Since you are running this from the shell, the debugging output will appear right on your screen. The **-x9** flag asks for debugging output; the number following the **x** indicates the verbosity. Debugging level 9 is fairly verbose, 5 is good for an initial audit, and numbers higher than 9 will give more information than is useful.

If there is a configuration problem that prevents the call from being completed, **uucico** may create a lock file in **/var/spool/uucp** that

records the time at which the call was attempted. Depending on how you've configured the link, an attempt to retry the call might fail because **uucico** thinks that insufficient time has elapsed since the last attempt. You must remove the file **/var/spool/uucp/STST/.***system* in BSD or **/var/spool/uucp/.Status/***system* in HoneyDanBer to get **uucico** to try the call again.

It is more difficult to monitor an incoming call. You will probably have to content yourself with looking at output from **ps** and **w** to trace the general flow of the conversation and to be sure that the remote system was able to log in without problems. If you are running HoneyDanBer and a remote site initiates a conversation with debugging turned on, some diagnostic information may be put in a file called **AUDIT** in the UUCP spool directory.

Configuring your Mail System to Use UUCP

Most versions of **sendmail** do not handle UUCP mail directly. Instead, they give UUCP-addressed mail to a special UUCP mailer (usually called **rmail**) for forwarding. The mail system does the address parsing and rewriting, and the UUCP mailer is responsible for examining the final result and forwarding it to the appropriate machine. On most systems, the UUCP mailer also receives incoming UUCP mail from other machines and submits it to the local mail system.

Chapter 21, *Electronic Mail*, describes the proper incantations for configuring **sendmail**. However, since UUCP addresses are stylistically different from other kinds of mail addresses, you may find it difficult to add UUCP to an existing installation. Unfortunately, some vendors' mail-handling tools make the process even harder than it needs to be.

These versions of ***sendmail*** *are included on the CD-ROM.*

We recommend installing either the IDA **sendmail** package or version eight of Berkeley **sendmail**. Both of these packages contain versions of **sendmail**, **/bin/mail**, and **rmail** that are designed to work together and make the UUCP configuration process easy.

See Chapter 16 for more information about DNS.

If you want email addresses for your users to look like Internet-style addresses rather than UUCP addresses, you need to have a UUCP neighbor on the Internet who can front for you. The Internet Domain Name System (DNS) allows the fronting machine to publish a mail exchanger (MX) record for you in much the same way that it publishes its own name. This will cause mail sent to you at the Internet-style address to be routed through your neighbor for forwarding via UUCP. Commercial UUCP vendors provide this service as a matter of course.

Testing UUCP mail is generally fairly easy. You can send mail to yourself that must travel through a remote host and back again to get to you, proving that you can both send and receive mail correctly. In the follow-

ing examples, we'll assume that "lair" is the name of your neighbor and that your local system's name is "jove."

The first thing you should try is

```
mail -v lair!jove!yourname
```

This will let you watch **sendmail** chew on the address. A successful session looks something like this:

```
% mail -v lair!jove!garth
Subject: Hi garth
<Control-D>
Cc:
Null message body; hope that's ok

lair!jove!garth... Connecting to lair.uucp...
lair!jove!garth... Sent
```

If instead of this, **sendmail** tells you that it's "totally stumped," the problem lies in your **/etc/sendmail.cf** file. You need to review your UUCP mailing rules. If you get messages saying that the host lair is unknown, that probably comes from the UUCP system itself and requires that you go back to the configuration files in **/usr/lib/uucp** to try to figure out what is wrong.

You can use **uustat -slair** to verify that a job has been queued for the remote site. If everything looks good, force a call to lair with:

```
/usr/lib/uucp/uucico -r1 -slair -x5
```

It may take a minute or two for the message to get turned around, so you may have to force another call to pick up the incoming message. Once the second session has completed, wait another minute and then check your mailbox.

If there's no mail, you should check the contents of the UUCP log files in **/var/spool/uucp** (described starting on page 701). If the mail went out but didn't come back, the problem most probably lies with the address rewriting rules in your **sendmail.cf** file. Refer to Chapter 21, *Electronic Mail*, for help with debugging address rewriting rules.

If you configure **sendmail** to read the names of UUCP sites directly out of **L.sys** or **Systems**, be sure to restart **sendmail** when you add a new UUCP neighbor so that **sendmail** will become aware of it.[10]

Setting Up Automatic Calling using cron

Once you know that your UUCP system is functional, you need to set it up for automatic operation. The times you specify in **L.sys** or **Systems**

10. Actually, you'll need to restart **sendmail** no matter how you tell it about your neighbors.

tell when it's *permissible* to make a call, but the contents of this file don't cause calls to actually be made. You use the **cron** daemon for that.

The **cron** daemon executes programs at preset times. It is a whole topic in itself, and more information about how to use it is given in Chapter 10, *Periodic Processes*. Here, we will just present some boilerplate configurations that can be used directly or with only a little modification.

The timing of incoming calls is controlled by your UUCP neighbors. If you find that incoming calls arrive at inconvenient times, the only thing you can do is place your neighbors on callback status using **USERFILE** or **Permissions**. But actually, it's easier to just call the administrators of the offending sites and ask them to modify their calling schedules.

You do have control over the times when you initiate calls to other systems. The usual way to initiate calls is to run **uucico** in master mode (the **-r1** flag does this). With no other arguments, **uucico** scans its spool directories to discover for which systems work has been queued, and then makes calls to each of those systems, provided that the current time is within the range specified for each system in the **L.sys** or **Systems** file. If there is no work for a site, no call is placed.

uucp and **uux** will, by default, attempt to start up **uucico** whenever a job is queued for a remote system, unless the **-r** flag is included on the command line. Mail and news always use this option, so you have more control over their schedules than you do over the transfer times of operations initiated by users.

The simplest way to set up UUCP is to have **cron** run **uucico** in master mode every hour or so. The longest amount of time any job will spend sitting in your system's UUCP queue (assuming it is eligible to be transmitted) is the amount of time between executions of **uucico**.

On an ATT system, the crontab line for an hourly check will look something like this:

```
43  *  *  *  *   /usr/lib/uucp/uucico -r1
```

This line should be added to the configuration for the uucp account with **crontab -e uucp**. The BSD syntax is slightly different.

Calling only when there is outbound data for a site is fine as long as your neighbors are set up to call you when they have data for you. But if you call a site that doesn't call you, you have to be sure that a connection is made every once in a while regardless of whether or not you have work for it. There is no way for you to know whether or not there are jobs waiting on the other end without forcing the call.

It is only slightly harder to do a poll than it is to do a queue scan; the problem is that a separate command line is required for each system.

The **-s***system* flag to **uucico** will cause the named system to be called regardless of whether or not there is work. This flag should be supplemented with **-r1** to ensure that **uucico** is started in master mode.

The easiest way to handle polling is to write a shell script and install it in your UUCP directory. If you do not poll many systems, you can hardwire their names right into the script; otherwise, you should have a script that accepts as arguments the names of systems to call. You should run this script out of **cron** just as you run **uucico** directly for non-polled sites. Some versions of UUCP provide such a polling command ready-made.

As a trivial example, the following script simply polls its arguments:

```
#!/bin/csh -f
# uucp.poll sitename1 sitename2 ... sitenameN
foreach site ($argv)
  /usr/lib/uucp/uucico -r1 -s${site}
end
```

If you were to use this script as part of a polling regime, you'd have a crontab line for each set of sites to be polled at once. For example:

```
00  7     *  *  *   /usr/local/etc/uucp.poll bog pika
23  7,16  *  *  *   /usr/local/etc/uucp.poll tiger luna
```

30.8 THE UUCP LOG FILES

See Chapter 12 for more information about the care and feeding of log files.

Most of the files in **/var/spool/uucp** are internal to the UUCP system and won't need to be configured or inspected. There are a couple of files and directories that may interest you, however. The information in these files tends to be extremely verbose, so you may wish to run some cleanup scripts out of **cron** to delete them every week or to compress them and move them somewhere else.

LOGFILE and .Log

In BSD-ish UUCPs, **LOGFILE** contains general information about UUCP activity on the local site and actions performed when connected with remote sites, whether or not the calls were initiated locally. Every time a UUCP remote command execution request is queued locally, a line of the following form is entered in the **LOGFILE**:

```
evi lair (7/19-21:52-5938) XQT QUE'D (rmail allspice!bob)
```

which shows the requestor (evi) the remote site (lair), the date, and the command to be executed.

In addition, information about the various calls that were attempted or received, whether or not they succeeded, and what went on during the conversations are also recorded here.

In ATT-ish UUCP, the **LOGFILE** information for each neighbor is separated into the files **/var/spool/uucp/.Log/uucico/***neighbor* and **/var/spool/uucp/.Log/uuxqt/***neighbor*. Most releases provide a command called **uudemon.cleanup** that gathers the logs at the end of each day and summarizes them into **/var/spool/uucp/.Old**.

SYSLOG and .Admin/xferstats

SYSLOG is used in BSD, and **.Admin/xferstats** in ATT. The format is the same. This file shows the size in bytes of each UUCP transaction, the name of the requestor, the site with which the transaction occurred, the time of the transaction, and the time needed to perform the transaction. For example, the line

```
news akelei (7/19-20:55) (553748130.51) sent data 8347
  bytes 38.85 secs
```

indicates that 8,347 bytes of news were transferred in 39 seconds.

ERRLOG and .Admin/errors

ERRLOG (**.Admin/errors** in HoneyDanBer) is used to record various errors that occur during the operation of the UUCP system. This file is the first place you should look when attempting to track down UUCP problems. A line from this file looks like:

```
ASSERT ERROR (uux) pid: 16140 (10/4-19:15) CAN'T OPEN
  D.sigiX00W0 (0)
```

Here, **uux** was unable to open a file that it expected to have access to. This may have been caused by someone removing the file by hand.

30.9 MISCELLANEOUS UUCP SUPPORT PROGRAMS AND FILES

There are several utilities supplied with UUCP for examining the job queue, finding the status of connections, and reading the logs in an organized way. These programs aren't really essential to the operation of UUCP, but they can be helpful in tracking down problems and in monitoring traffic.

uuclean: Sanitize the Spool Directory

uuclean is used to remove outdated cruft from the spooling area and to perform general housekeeping for the UUCP system. It can be used to remove all files over a certain age, and can select the files to be deleted on the basis of their names, which is useful if only files of a certain type (e.g., lock files) are to be removed.

uuclean is generally run out of **cron**, just like **uucico**. Normally, **uuclean** is run every 24 hours, and only the **-m** flag is supplied; this

causes **uuclean** to do a general cleanup and to send mail to any users whose stale files were deleted.

uuq and uustat: Monitor the UUCP Queue

These two commands are similar, and most systems provide one or the other. The commands allow you to monitor the UUCP work queue, and they support a number of options to select only jobs having certain qualities, such as being destined for a particular system.

uuq and **uustat** can also be used by the administrator to delete jobs from the UUCP queue; to do this, the command must first be run in normal or verbose mode to provide the job names for each job, and then run again with the **-d** (for **uuq**) or **-k** (for **uustat**) flag to indicate which job is to be deleted.

uusnap: Show Status of Connections

This violent-sounding command displays the status of all UUCP connections that have work waiting for them on the local system. It shows the number of files and commands waiting to be handled, and the current status of the connection. If the status is blank, it means that the system has not yet been called; otherwise, it will say that the remote system is currently being talked to, or will show any problems or comments relevant to previous connection attempts.

uuname: Show UUCP Neighbors

Without any arguments, **uuname** shows the names of all known UUCP neighbors. With a **-l** flag, it shows the UUCP name of the local host.

To obtain the list of neighbors, **uuname** simply looks at the first field of each line in the **L.sys** or **Systems** file, hardly an amazing feat. However, it does handle comments and continuation lines, so it's useful for writing shell scripts. **uuname** is also useful because ordinary users do not have read permission on **L.sys** or **Systems**, so they need an intermediary, setuid program to search this file for them.

uupoll: Force a Call to a UUCP Neighbor

This command is used to force a telephone call to a remote system. Rather than using **uucico** with polling flags, as we've described in this chapter, it queues a job for the system and then attempts to start **uucico** normally. **uucico** sees that there is work and initiates a call.

uulog: The Command of a Thousand Faces

This command does different things under different versions of UUCP. Some versions of the UUCP system actually keep log information in sep-

arate files in the spool directory; this information is not compiled into **/var/spool/uucp/LOGFILE** until **uulog** is run. On these systems, **uulog** should be run at least once per day.

On other systems, log entries are written directly to the log files and the **uulog** command is used to search through these logs for information about particular systems. Consult the manuals for your system to find out what your **uulog** does.

30.10 SPECIFICS FOR VARIOUS OPERATING SYSTEMS

Except for BSDI, our example systems all use HoneyDanBer UUCP.

Installation of UUCP is optional in Solaris. Refer to your system installation manuals for specific instructions. **/etc/uucp** is used instead of **/usr/lib/uucp**; the spool directory is **/var/spool/uucp**, and the public directory is **/var/spool/uucppublic**.

HP-UX is a plain vanilla HoneyDanBer system. Configuration files are located in **/usr/lib/uucp**, and spool and public directories are in **/usr/spool/uucp** and **/usr/spool/uucppublic**.

IRIX is similar to HP-UX. The **/usr/lib/uucp/Devconfig** file is used for configuration of protocol stacks.

SunOS UUCP is essentially the same as that for Solaris. Installation is optional. Refer to your system's installation guide.

Installation is optional. The system is mostly vanilla HoneyDanBer. A **uucpsetup** command is provided to assist with configuration.

BSDI uses a version of BSD UUCP that has been enhanced to permit local customization. Configuration files are kept in **/etc/uucp**; the **/etc/uucp/CONFIG** file is used to declare the location of other directories. See the man page for **uuparams** for information about how to set up this file. The other directories default to **/var/spool/uucp** and **/var/spool/uucppublic**.

30.11 RECOMMENDED SUPPLEMENTAL READING

O'REILLY, TIM and GRACE TODINO. *Managing UUCP and Usenet*. Sebastopol: O'Reilly & Associates, 1990.

TODINO, GRACE and DALE DOUGHERTY. *Using UUCP and Usenet*. Sebastopol: O'Reilly & Associates, 1990.

KROL, ED. *The Whole Internet*. Sebastopol: O'Reilly & Associates, 1992.